АНДРЕЙ ШКРЫЛЬ

PHP — ЭТО ПРОСТО

ПРОГРАММИРУЕМ ДЛЯ WEB-САЙТА

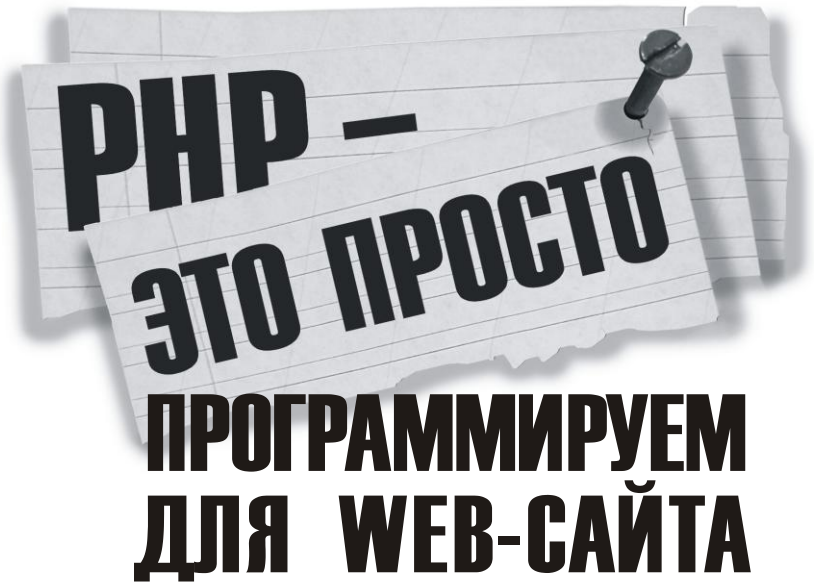
ЧТО НУЖНО ЗНАТЬ О PHP

РАБОТА С PHP, APACHE, MySQL И phpMyAdmin

ПРАКТИЧЕСКИЕ ПРИМЕРЫ И ПОЛЕЗНЫЕ СОВЕТЫ

ГОТОВЫЕ РЕШЕНИЯ Mambo, phpBB, FCKEditor И CPanel

Андрей Шкрыль



Санкт-Петербург

«БХВ-Петербург»

2006

УДК 681.3.06
ББК 32.973.26-018.1
Ш66

Шкрыль А. А.

Ш66 PHP — это просто. Программируем для Web-сайта. — СПб.: БХВ-Петербург, 2006. — 368 с.: ил.

ISBN 5-94157-905-5

Рассмотрены практические вопросы программирования на языке PHP и создания полноценных интерактивных Web-сайтов. На реальных примерах показаны особенности работы с APACHE, MySQL, phpMyAdmin и с популярными готовыми решениями — Mambo, phpBB, FCKEditor и CPanel. Материал сопровождается множеством иллюстраций, схем и полезных советов: начиная с использования общедоступных интернет-сервисов, таких как курс валют, и заканчивая настройкой Web-сервера APACHE.

Для начинающих программистов

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Екатерина Капалыгина*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Зинаида Дмитриева*
Художник	*Елена Беляева*
Дизайн обложки	*Инны Тачиной*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.06.06.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 29,67.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 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 5-94157-905-5

Оглавление

Благодарности **1**

Введение **3**

Глава 1. История PHP **5**

Глава 2. Что нужно знать о PHP **7**
2.1. Как работает PHP 7
2.2. Преимущества PHP 16

Глава 3. Среда разработки PHP-программ **17**
3.1. PHP Expert Editor 18
3.2. PHP Designer 2006 20
3.3. NOTEPAD++ 24

Глава 4. Джентльменский набор Web-разработчика **29**
4.1. Установка Денвера 31
4.2. Работаем с Денвером 35
4.3. Денвер изнутри 39
4.4. Создаем свой сайт 39
4.5. Конфигурационные файлы 40
4.6. Информация о PHP 41

Глава 5. Наши первые программы на PHP **45**
5.1. Программируем — начнем с простого 45
5.2. Переменные 54
5.3. Практикуемся в работе с переменными 57
5.4. Функции 62
5.5. Константы 65
5.6. Массвивы 67
5.7. Переменные окружения 71

Глава 6. Счетчик посещений 75
6.1. Разработка программы 75
6.2. Счетчики, которые не надо разрабатывать 88
6.3. Информеры 91
6.3.1. Информеры от Rambler 91
6.3.2. Курс валют 93
6.3.3. Цены на российские автомобили 95

Глава 7. Все, что нужно знать о формах 97
7.1. Назначение форм 97
7.2. Создание формы 98
7.3. Простые элементы формы: поле ввода и кнопка 101
7.4. Немного практики 105
7.5. Методы отправки данных формы 108
7.5.1. Метод GET 109
7.5.2. Метод POST 110
7.5.3. Что лучше: GET или POST? 111
7.6. Остальные элементы формы 112
7.6.1. Поле для ввода пароля 112
7.6.2. Переключатель (Radio button) 113
7.6.3. Флаг (CheckBox) 114
7.6.4. Список 115
7.6.5. Поле ввода многострочного текста (TextArea) 118
7.6.6. Скрытое поле 119

Глава 8. Что скрывает браузер 121

Глава 9. Сплошная практика 127
9.1. Форма обратной связи 127
9.2. Гостевая книга 135
9.2.1. Приступаем к работе 136
9.2.2. Дорабатываем гостевую книгу 151
9.2.3. Цензура не дремлет 155
9.2.4. Управляем с удобством — админка 160
9.2.5. Использование сессий 168
9.3. Голосование 175
9.3.1. Приступаем к работе 175
9.3.2. Использование cookie 183
9.3.3. Админка для системы голосования 190
9.4. Загрузка файлов 198
9.4.1. Основы 198
9.4.2. Познаем тонкости 208
9.4.3. Полноценный скрипт для закачки файла 209
9.5. Определяем быстродействие скрипта 212

Глава 10. Базы данных и работа с MySQL 221
10.1. Основные понятия 221
10.2. phpMyAdmin — первое знакомство 224
10.3. Разрабатываем структуру будущей базы данных 224
10.4. Создаем БД или работаем в phpMyAdmin 229
10.5. Разрабатываем план кодирования 241
10.6. SQL-запросы 242
10.7. Кодирование 245
10.7.1. Подключаемся к базе 246
10.7.2. Модуль авторизации 248
10.7.3. Модуль logout.php 258
10.7.4. Основной файл форума index.php 258
10.7.5. Модуль вывода информации show.php 260
10.7.6. Модуль действий action.php 276
10.8. В заключение главы 283

Глава 11. FCKEditor 285
11.1. Установка 285
11.2. Первое знакомство 285
11.3. Простой пример 287
11.4. Настраиваем панели инструментов 289
11.5. Получаем данные из редактора 291
11.6. Настраиваем инструмент по закачке файлов 293

Глава 12. phpBB 297
12.1. Установка 297
12.2. Работаем с форумом 300

Глава 13. Знакомство с Mambo 305
13.1. Установка 306
13.2. Основные принципы работы с Mambo 311
13.3. Разрабатываем свой сайт 313
13.4. Устанавливаем модуль для Mambo 329
13.5. Устанавливаем компонент для Mambo 330
13.6. Устанавливаем шаблоны для Mambo 334
13.7. В заключение 334

Глава 14. Закачиваем сайт на хостинг 335
14.1. Бесплатный хостинг от Holm.ru 335
14.2. cPanel 340

Заключение 347

Приложение 1. Настройка PHP 349

Приложение 2. Список сайтов, связанных с PHP 353

Предметный указатель 355

Хочу сказать спасибо своим родителям, которые привили мне любовь к получению новых знаний и упорство в достижении целей, а также поддерживали меня в сложных жизненных ситуациях. Хочу поблагодарить брата Антона и сестру Ольгу за их веру в меня.

Выражаю большую признательность своей жене Наташе за ее поддержку, помощь в поиске новых идей, а также за ее терпение.

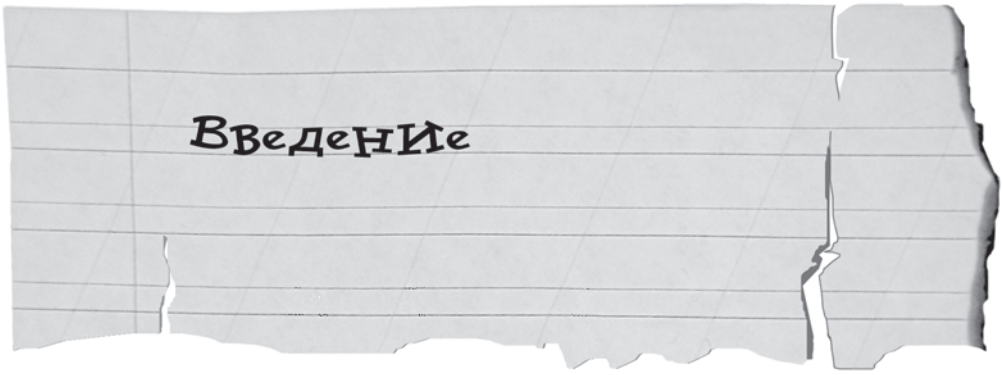
Огромная благодарность замечательным и талантливым людям, профессионалам своего дела Шишигину Игорю и Фленову Михаилу, за их моральную поддержку и консультации в сложных для меня вопросах. Я очень рад и горд знакомству с ними.

Хочу поблагодарить Колмакова Андрея aka Slider и Бурдачева Сергея aka Mef, которые помогали мне своими советами и делились опытом.

Особая признательность Кропотову Максиму, творчески одаренному и грамотному руководителю, который предлагал и продолжает предлагать участие в интереснейших Web-проектах. Работать с ним в одной команде доставляет истинное удовольствие.

Спасибо Климову Виктору за приглашение участвовать в проекте по реализации общегородской биллинговой системы.

И, конечно, огромная признательность коллективу издательства "БХВ-Петербург", благодаря которому вы держите в руках эту книгу.



Интернет уже давно прочно вошел в нашу жизнь. В настоящее время огромное число людей не мыслят своей жизни без глобальной сети.

Вспоминается аналогия с телевидением: телевизор был когда-то в новинку и люди специально приходили в гости друг к другу, чтобы посмотреть на это чудо человеческой мысли. Никто не мог тогда подумать, что телевидение станет цветным, и появится множество самых разнообразных телевизионных программ на любой выбор. Такой же новинкой был когда-то и Интернет — в момент начала своего пути. Сейчас даже дети знают, что это такое. Когда нам надо срочно найти какую-то информацию — мы идем в Интернет, мы хотим пообщаться с родственниками или друзьями, которые живут за множество километров от нас, — мы идем в Интернет. Нам одиноко и мы ищем вторую половинку — мы идем в Интернет. Мы голодны и хотим есть — пожалуйста, заходите в Интернет — пару щелчков мышью и через несколько минут обед у нас на столе. У нас есть хобби, или мы любим обсуждать темы, связанные с нашей профессией, — мы идем в Интернет и общаемся на разнообразных форумах, где мы можем познакомиться с новыми людьми, поднять давно волнующую нас тему.

Я думаю, многие пытались делать простейшие странички в Интернете с помощью языка гиперразметки — HTML. Главный недостаток этого способа заключается в том, что странички получаются статическими, т. е. диалог с пользователем сводится к выбору им определенных разделов и получению информации, другими словами, он сводится к нулю. Лет 10 назад такой Web-сайт мог бы пользоваться популярностью. Но время не стоит на месте. Сейчас ситуация изменилась и современный сайт — это интерактивный сайт с широким спектром сервиса: форум, интернет-магазин, голосование. Перечисленные примеры это всего лишь малая часть того, что можно встретить в сети. Но меняется не только мода, меняются и технологии, с помощью которых можно следовать за ней. Это утверждение полностью оправдывает язык PHP, который совершил настоящий прорыв в области Web-программирования. Несмотря на свою простоту, он обладает огромным потенциалом, позволяющим делать фактически все, что угодно. Самое главное преимущество PHP заключается в том, что его может освоить практически каждый человек, хотя бы немного знакомый с компьютерами, например, по курсу школьной

информатики. Таким образом, перед вами открываются поистине безграничные возможности самореализации в глобальной сети.

В данной книге я использовал подход с точки зрения начинающего программиста (если бы это была художественная книга, то можно было бы сказать, что она написана от первого лица). Книга изобилует различными схемами, которые позволяют читателю проще осваивать технические тонкости, связанные с разработкой скриптов. Основное внимание уделено не количеству, а качеству излагаемого материала, а также тем фундаментальным вещам, без которых программирование на PHP будет сложным занятием. Девиз книги: отбросьте сомнения и страх, у вас все получится, скоро и без особых усилий вы сможете создать свой собственный интерактивный сайт.

Приятного прочтения.

Глава 1
История PHP

В 1995 году Расмус Лердорф разработал PHP/FI (Personal Home Page/Forms Interpreter, Персональная домашняя страница/Интерпретатор Форм). Расмус создал данный инструмент, чтобы узнать, кто же читает его резюме, которое было размещено в Интернете. Также Расмус предоставлял возможность просматривать собираемую статистику посещений, которая была реализована с помощью разработанного им средства.

В 1997 году Энди Гутманс и Зив Сураски в поисках удобного средства для создания динамических Web-страниц обнаружили PHP/FI. Они оценили идею Расмуса, но их не устроила скорость работы и функциональные возможности данного инструмента. Поэтому они переписали его с нуля, это было возможно, т. к. исходный код PHP/FI был доступен абсолютно всем. Так появился PHP 3. Язык получился удобным, обладал лучшей функциональностью и скоростью работы, по сравнению с его предком. Изменения коснулись и названия языка, теперь оно стало — PHP: Hypertext Preprocessor (PHP: Препроцессор гипертекста), что было очень удачным решением со стороны разработчиков.

PHP был и остается языком с открытым кодом, и в его усовершенствовании может принять участие каждый. Благодаря этому, PHP смог быстро развиваться и совершенствоваться, что и произошло в результате.

После выхода 3-й версии PHP стал завоевывать внимание многих разработчиков в области Web-программирования, и с каждым годом эта тенденция все увеличивалась, к тому же нашлось достаточное количество людей, которые выразили желание принять участие в совершенствовании PHP. Сегодня PHP достойно занимает лидирующие позиции среди средств разработки скриптов для Web. В настоящее время существует уже 5-я версия этого замечательного средства. Какие же изменения приобрел PHP на сегодняшний день? Все очень просто, с каждой новой версии язык становится все мощнее, в настоящий момент он обладает огромным количеством функций, которые позволяют решать множество проблем, возникающих при разработке Web-скриптов. В то же время он остается одним из самых простых языков, даже если вы никогда не программировали на C++ или Pascal, вы все равно сможете успешно овладеть PHP в достаточно короткие сроки. И для этого вам понадобятся всего лишь основы знаний HTML.

Также стоит отметить, что для PHP не нужна специальная среда, он является кроссплатформенным, т. е. может работать и под Windows, и под UNIX, и под Linux. Эта особенность также сыграла не последнюю роль в популяризации PHP.

Еще одним большим плюсом является то, что в Интернете для PHP существует большая библиотека готовых скриптов. Как небольших в виде счетчика посещений Web-страницы, так и целых систем, которые позволяют создать сайт с нуля при минимальном написании кода. Таким образом, даже если вы знаете только основы PHP, вы достаточно успешно сможете построить полноценный интерактивный сайт, воспользовавшись рядом готовых решений. Конечно, это не самый оптимальный вариант, но об этом мы поговорим подробнее на страницах книги.

Необходимо отметить, что документация по PHP является бесплатной, и она всегда доступна по адресу **http://www.php.net/download-docs.php** (имеется и русскоязычная версия).

Глава 2
Что нужно знать о PHP

Пожалуй, PHP — один из немногих языков, который удобнее изучать, реализуя конкретные примеры. Именно так мы и будем поступать на протяжении всей книги. На мой взгляд, это один из самых удобных и практичных способов освоить PHP, потому что данный язык тем и хорош, что позволяет достаточно быстро решать поставленные перед ним задачи, даже если человек, который этим занимается, обладает не высокой квалификацией.

Как я уже говорил ранее, при изучении PHP достаточно обладать хотя бы базовыми знаниями HTML. Если их у вас нет, это тоже не проблема, т. к. достаточно зайти в Интернет и ввести в любом поисковике фразу "Учебник по HTML". Могу сказать точно, вам будет из чего выбрать.

Замечание

В мире Web много различной терминологии, что поначалу может запутать человека, который попадает в этот мир впервые. Если вы предполагаете писать только на PHP, вы все равно столкнетесь со многими терминами, напрямую к PHP не относящимися, но являющимися неотъемлемой частью мира Web. Поэтому если на страницах книги вы столкнулись с каким-то новым, неизвестным для вас понятием, то советую воспользоваться любым поисковым сервисом, введя в него фразу "Толковый словарь Web-терминов", что позволит более быстро и точно понять материал.

2.1. Как работает PHP

Для того чтобы понять, каким образом работает PHP, предлагаю обратиться к самому простейшему примеру, когда пользователь запрашивает обычную HTML-страничку (рис. 2.1).

То есть сначала пользователь запрашивает определенный HTML-документ, предположим, он хочет получить файл NEWS.HTML, чтобы просмотреть последние новости. Для этого, как правило, используется специальная программа под названием Web-браузер (например, Internet Explorer, Opera, Mozilla и др.), наш пользователь не исключение, он тоже использует эту программу. Соответственно он запрашивает файл

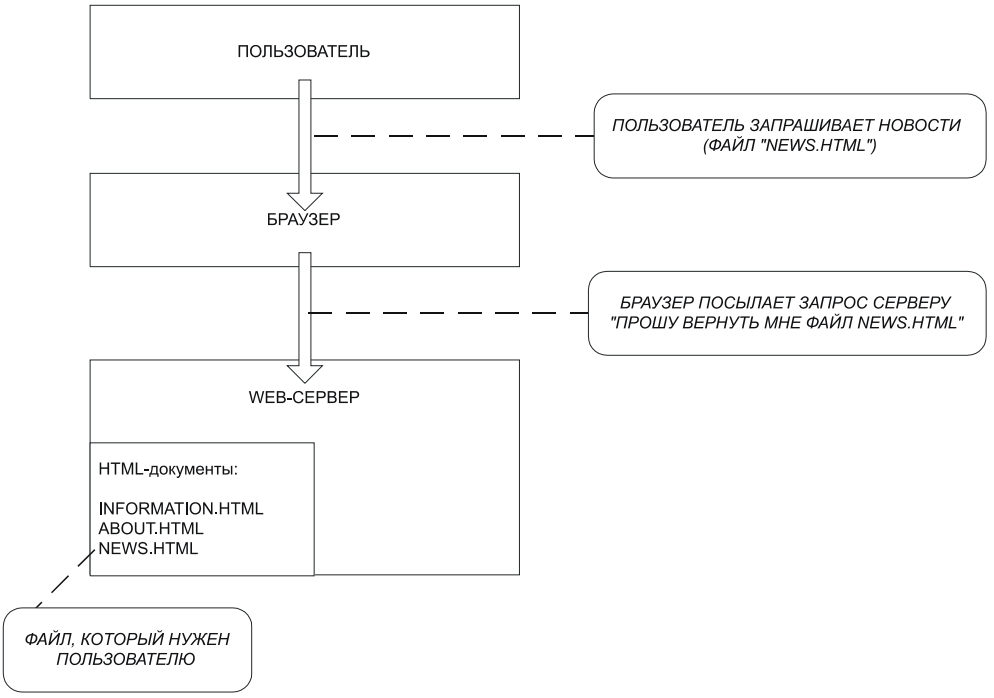


Рис. 2.1. Схема обработки запроса пользователя по получению файла NEWS.HTML

NEWS.HTML у определенного сервера, назовем его просто СЕРВЕР, с помощью своего Web-браузера.

Web-браузер, получив команду от пользователя, посылает запрос на сервер, в нашем случае это будет приблизительно следующее: "ПРОШУ ВЕРНУТЬ МНЕ ФАЙЛ NEWS.HTML".

На рис. 2.2 в виде схемы изображен процесс передачи сервером браузеру запрошенного файла и соответственно получение той информации пользователем, которую он хотел увидеть.

Сервер, получив запрос от браузера, начинает его обрабатывать. Сначала он проверяет — есть ли у него такой файл, если да (мы рассматриваем именно этот случай), то передает его браузеру, если нет, то возвращает *ошибку 404*, которая свидетельствует о том, что файл не найден. На рис. 2.3 вы можете увидеть окно **Internet Explorer**, когда при попытке обратиться к файлу обнаружилось, что запрашиваемого файла на сервере нет.

Браузер, получив файл, на основании HTML-тегов, которые содержатся в нем, формирует готовый документ, удобный для просмотра пользователя.

Здесь очень удобно провести аналогию с книгой. У книги есть определенные правила оформления, такие как наличие оглавления, названия книги и издательства, которое выпустило книгу, имя автора, нумерация страниц, что позволяет читателю удобно работать с каждой книгой. Так же и HTML (язык гиперразметки документов) позволяет задавать структуру документа на основании общих принципов,

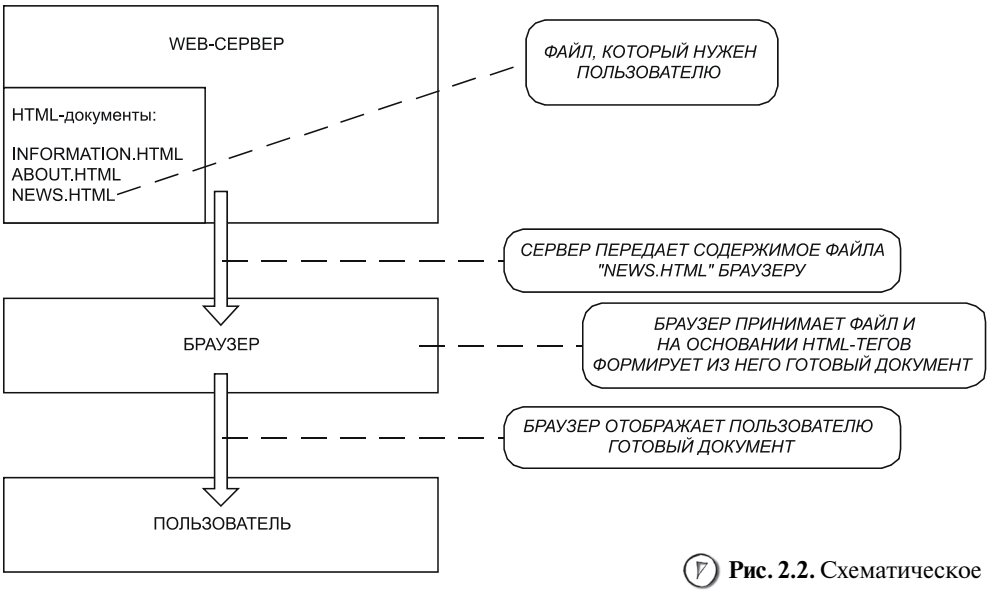


Рис. 2.2. Схематическое представление процесса, когда после запроса пользователя сервер возвращает нужный ему файл

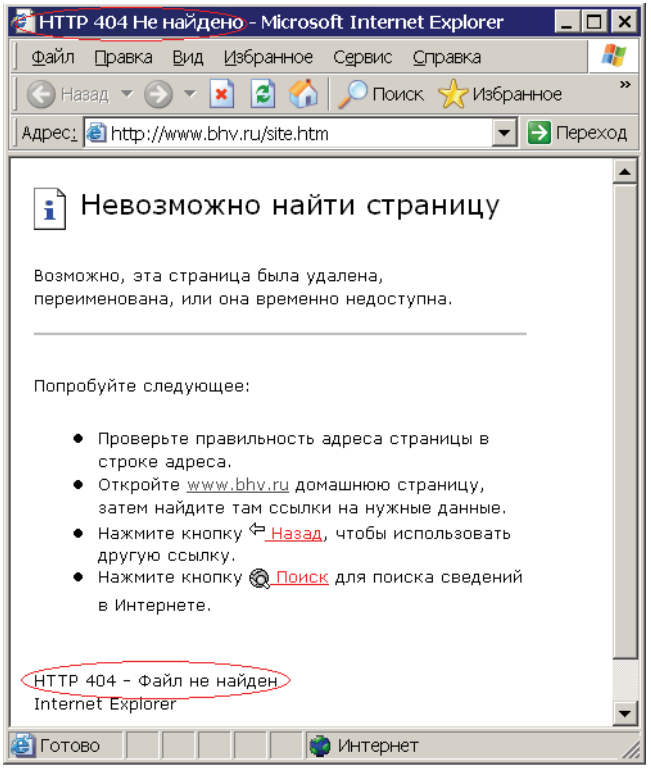


Рис. 2.3. Ошибка 404 возникает, когда запрашиваемый файл не найден. Фрагменты, отображающие код ошибки, обведены овалами

положенных в его основу. Если правила оформления книги важны для удобства работы читателя с книгой, то HTML-разметка в первую очередь рассчитана на браузер, который в соответствии с тегами формирует готовый документ, который уже в свою очередь получается удобным для просмотра пользователем. Сам по себе HTML-файл со всеми своими тегами будет очень не удобен для работы с ним, а вот когда браузер в соответствии с этими тегами оформляет документ, то мы получаем информацию в удобном для работы виде. На рис. 2.4 изображен HTML-файл, открытый в Блокноте, на рис. 2.5 HTML-файл, открытый в браузере Internet Explorer (на самом деле открытый в браузере и обработанный согласно тегам, размещенным в файле).

Теперь рассмотрим описанный ранее процесс еще раз, но предположим, что пользователь запрашивает файл, который является программой, написанной на PHP.

PHP является языком серверных скриптов (или сценариев), который можно встраивать в обычный HTML-документ. В *главе 1* уже упоминалось, что полное название PHP — это "PHP: Hypertext Preprocessor" (PHP: Препроцессор гипертекста). Так вот именно этот препроцессор в лице файла php.exe и обрабатывает программы (скрипты или сценарии), написанные на PHP. Предлагаю обратить внимание на рис. 2.6 и 2.7.

Замечание

Вы можете получить аналогичный рис. 2.4 результат, если в Internet Explorer откроете любой сайт, а затем выберете пункт меню ***Вид | Просмотр HTML-кода****.*

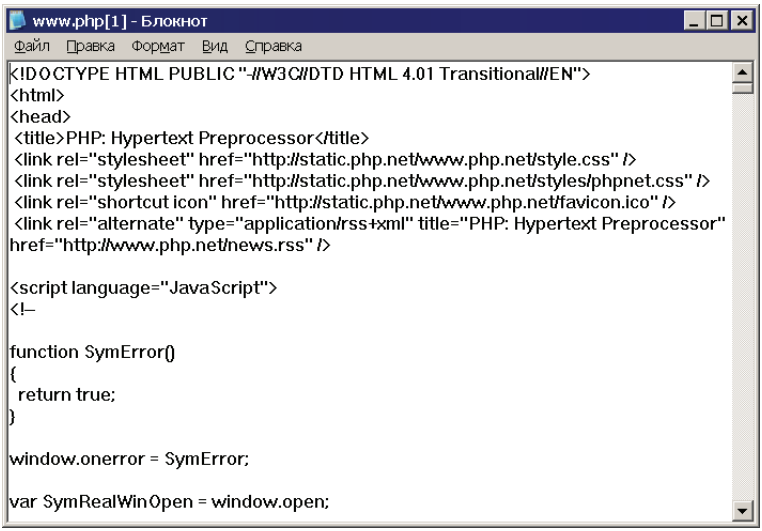


Рис. 2.4. HTML-файл, открытый в Блокноте (главная страница сайта **www.php.net**)

Рис. 2.5. HTML-файл, открытый в браузере Internet Explorer. По сравнению с предыдущим рисунком, он подвергся обработке браузером и поэтому представлен в удобном для восприятия виде

На рис. 2.6 представлена ситуация, аналогичная запросу HTML-файла, которую мы рассматривали ранее (см. рис. 2.1), главное отличие заключается в том, что сейчас пользователь запрашивает не HTML-файл, а PHP-программу и то, что на сервере теперь хранятся эти самые PHP-программы. Давайте теперь посмотрим, что происходит, после того как сервер получил запрос от пользователя (рис. 2.7).

То есть когда сервер получает запрос, он также проверяет, есть ли такой файл? Если да, то далее он обращается к файлу настроек Web-сервера, где сказано, что все файлы с расширением PHP являются программами и их нужно обрабатывать препроцессором гипертекста, прежде чем возвращать пользователю. Расширение PHP является самым распространенным, но есть также и другие (альтернативные) варианты расширений для программ, написанных на PHP:

- PHP2 — программа, написанная на PHP второй версии (например, PROGRAM.PHP2);
- PHP3 — программа, написанная на PHP третьей версии (например, PROGRAM.PHP3);
- PHP4 — программа, написанная на PHP четвертой версии (например, PROGRAM.PHP4);

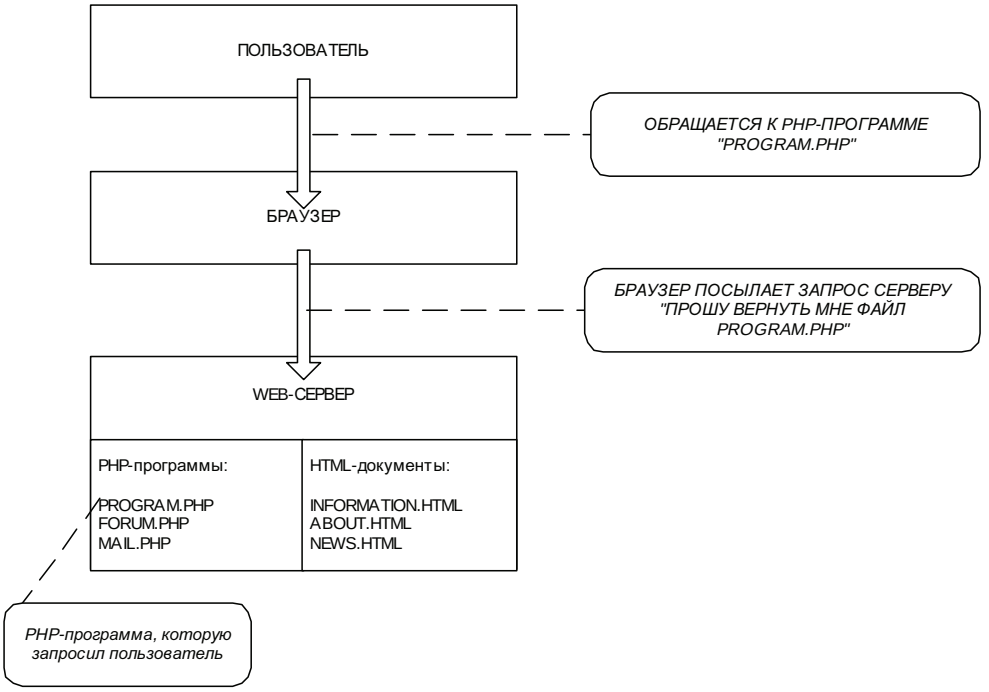


Рис. 2.6. Схематическое изображение процесса обращения пользователя к PHP-программе

Вообще вы можете настроить свой Web-сервер так, чтобы файлы с самыми разнообразными расширениями, например TXT, обрабатывались как PHP-программы — только в этом нет особого смысла.

- PHP5 — программа, написанная на PHP пятой версии (например, PROGRAM.PHP5);
- PHTML и PHTM — менее распространенный вариант (например, PROGRAM.PHTML или PROGRAM.PHTM).

Таким образом, файл PROGRAM.PHP передается препроцессору, на выходе которого получается обычный HTML-файл, он передается браузеру и пользователь получает статический HTML-документ — то есть результат выполнения PHP-программы.

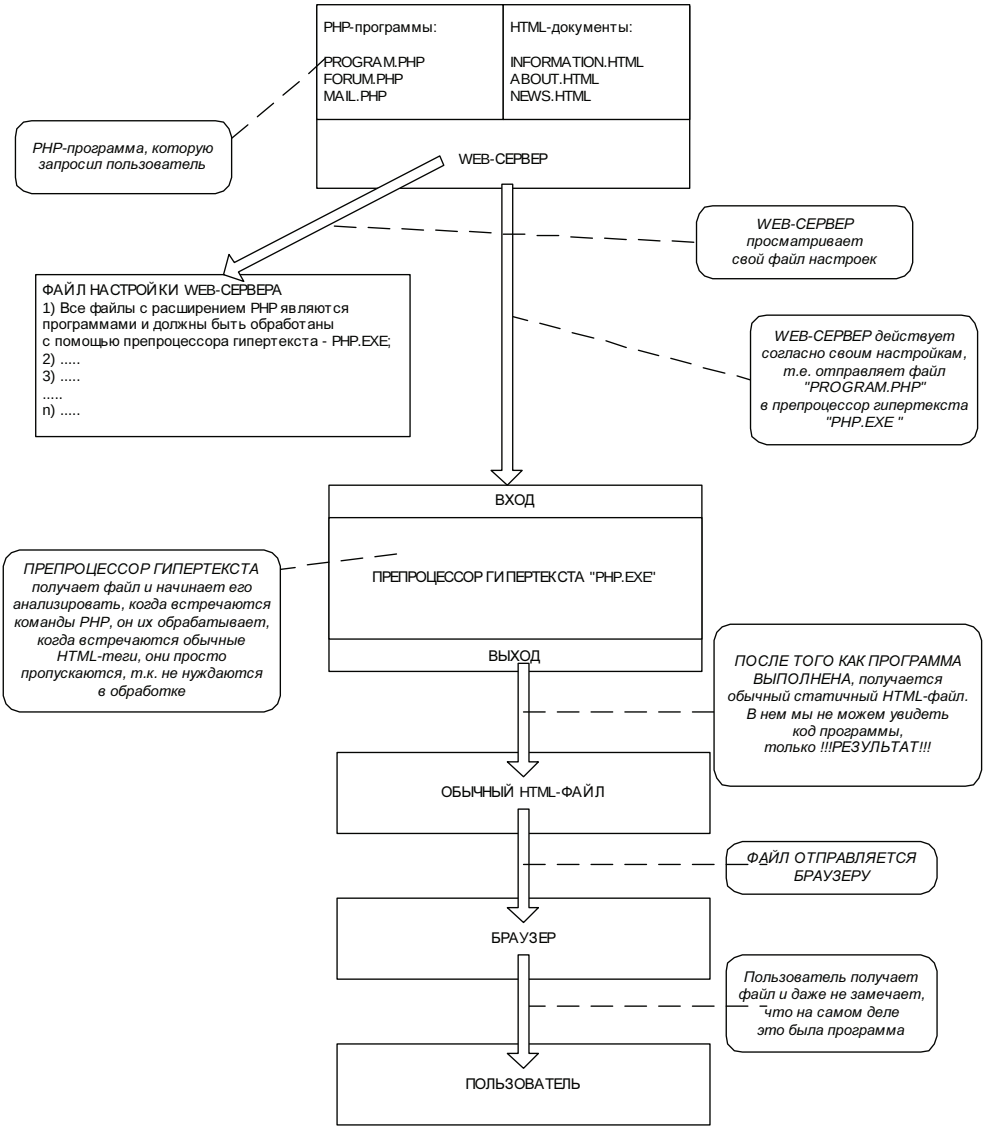
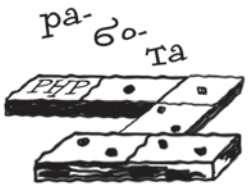


Рис. 2.7. Схематическое изображение процесса получения пользователем результата выполнения PHP-программы



Рассмотрим работу препроцессора. Обратите внимание на рис. 2.8, на котором изображен процесс обработки препроцессором программы, написанной на языке PHP.

Как же препроцессор узнает о том, что это именно код на языке PHP, который нужно обработать, а не обычные теги HTML. Условие того, что файл имеет расширение PHP, является не достаточным. Все очень просто — код на PHP оформляется специальными тегами, вариантов оформления несколько:

- `<?php...?>`
- `<?...?>`
- `<%...%>`
- `<script language="php">…</script>`

Рассмотрим работу препроцессора более подробно:

(1) После того как на вход препроцессора поступает файл, он начинает "просматривать" (или обрабатывать) его содержимое.

(2) Как только ему встречается открывающий тег, обозначающий код PHP, тут препроцессор переключается в активный режим, т. е. начинается выполнение команд, которые заключены в эти специальные теги. Некоторые команды после обработки будут заменены на HTML-теги, некоторые (например, осуществляющие подключение к базе данных) будут просто выполнены.

(3) Как только встречается закрывающий тег, обозначающий окончание кода на PHP, препроцессор опять переключается в режим просмотра кода, пока ему не встретится очередной участок кода на PHP или пока не будет достигнут конец файла.

Как частный случай, можно рассмотреть пример, когда файл полностью состоит из команд PHP, тогда препроцессор от начала до конца обрабатывает файл. Но такой подход применяется очень редко, чаще всего PHP-команды совместно используются с HTML-тегами, а можно сказать наоборот, HTML-теги используются с PHP-командами.

Положительным моментом обработки PHP-программы препроцессором является то, что пользователь не видит исходный код программы, он только получает результат ее выполнения.

Я специально упростил модель обработки PHP-программы и назвал элемент, который ее обрабатывает, препроцессором. Во многих изданиях PHP называют интерпретатором или компилятором, но на самом деле все немного сложней. Он полностью не является ни тем, ни другим. Это может достаточно сильно запутать человека, который только знакомится с PHP. Таким образом, намного удобнее называть его препроцессором гипертекста, который обрабатывает PHP-программы и на выходе имеет обычный HTML-документ.

Рекомендуется использовать `<?php...?>`, этот способ наиболее распространен и предпочтителен. Далее я буду предполагать, что вы используете первый вариант оформления.

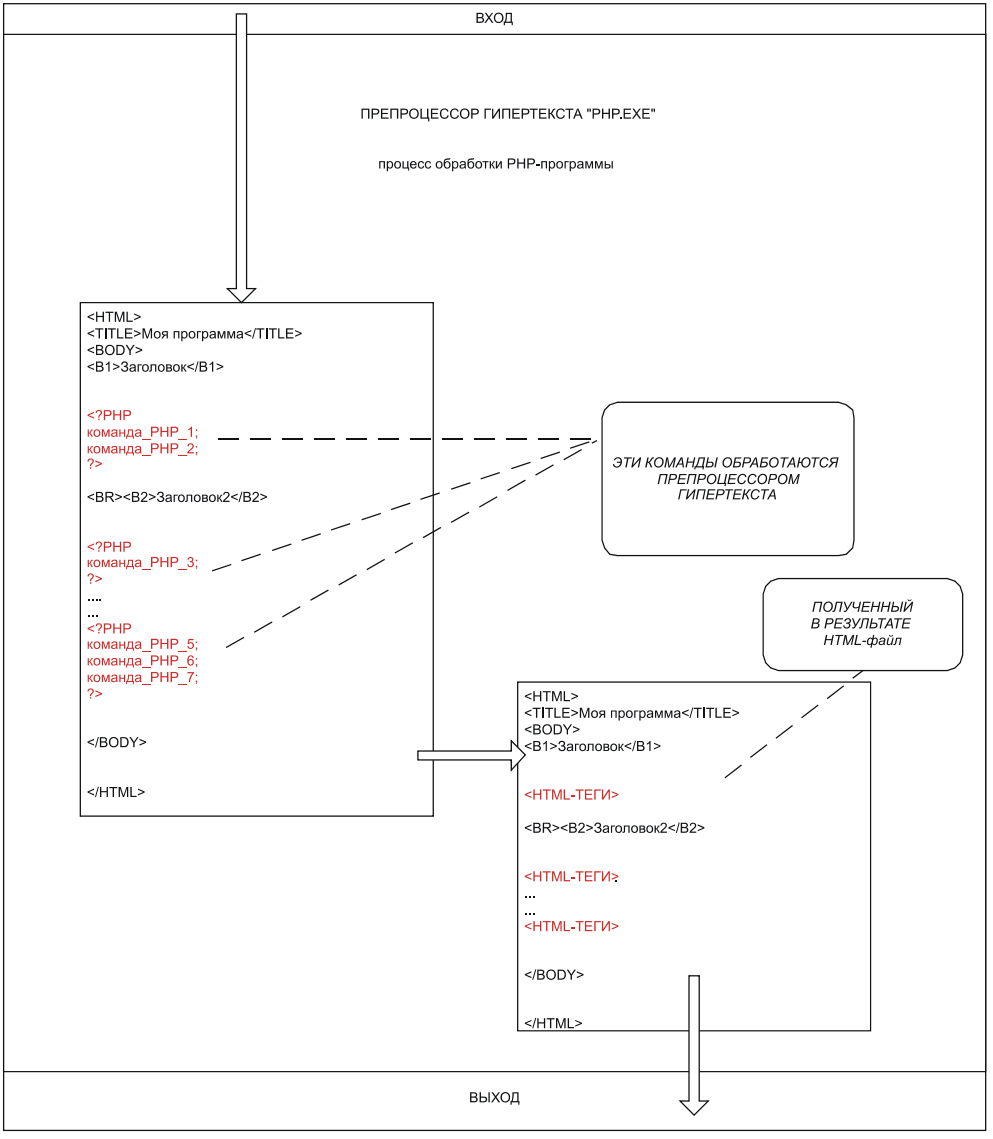


Рис. 2.8. Схема обработки PHP-программы препроцессором гипертекста

2.2. Преимущества PHP

- Фрагменты PHP-кода можно внедрять прямо в HTML-документ и наоборот. Главное при этом не забыть установить для файла расширение PHP.
- Так как PHP специально разработан для Web, многие вещи реализуются намного проще, чем если бы вы то же самое выполняли с помощью других языков.
- PHP очень прост в освоении и позволяет достаточно быстро решать поставленные перед ним задачи. То есть уже в течение нескольких минут вы сможете написать свою первую программу, а через несколько часов вы сможете сделать уже что-то более-менее серьезное, например, разработать собственную гостевую книгу.
- Если вы знакомы хотя бы с одним языком программирования, то освоить PHP вам будет очень легко. Если нет, то это тоже будет сделать несложно, просто на это уйдет немного больше времени.
- PHP бесплатен.
- PHP — кроссплатформенный язык, что не привязывает его к определенной операционной системе или определенному Web-серверу. В результате мы получаем огромную свободу в выборе программного обеспечения при работе с PHP.
- Совместимость программ, разработанных в ранних версиях PHP, с его более поздними версиями.
- PHP очень гибок. Он не зависит от браузера, т. к. программа выполняется на сервере, а результат возвращается в виде HTML-документа.
- Простота работы с базами данных: PHP может работать с различными СУБД, например, MySQL, MS SQL Server, ORACLE, InterBase, Firebird и др. Причем работа может быть организована с использованием различных технологий доступа к данным.
- PHP является объектно-ориентированным языком, что позволяет вам воспользоваться всеми преимуществами этой технологии.
- PHP располагает огромным количеством стандартных функций, которые позволят вам решить множество задач за минимальное количество времени.
- PHP многогранен, вам предоставляется множество готовых инструментов для вашего творчества, например, вы запросто можете работать с файлами формата PDF, электронной почтой или даже разработать интернет-магазин, который будет использовать средства электронной коммерции, такие как WebMoney или PayPal.
- PHP может работать с XML.
- В настоящее время в Интернете есть огромное количество готовых решений, написанных на PHP, начиная от простых счетчиков на вашу страницу и заканчивая системами управления и построения сайтов, которые позволяют до максимума упростить процесс создания и сопровождения сайта, поэтому, даже не будучи гуру в PHP, вы сможете создать качественный интерактивный сайт.
- Наконец, вы можете принять участие в совершенствовании PHP, т. к. его код открыт для всех.

Глава 3
Среда разработки PHP-программ

Любую программу удобно разрабатывать в специальной среде, конечно, можно писать и в Блокноте, но недостатки этого выбора сразу налицо: во-первых, синтаксис языка не подсвечивается, во-вторых, нет ни удобной справки, ни подсказок, ни возможностей автоматически вставлять наиболее распространенные команды или фрагменты кода, ни многих других, которые не только помогают сократить время разработки программы, но и сделать этот процесс максимально удобным и комфортным.

В настоящее время существует огромное количество редакторов, в которых вы можете создавать и редактировать файлы PHP. В эту главу вошли три программы, которыми можно пользоваться сразу после элементарного процесса установки, и самое главное они бесплатные, правда, одну из них — PHP Expert Editor — нужно зарегистрировать, но это абсолютно бесплатно, лишь сделать пару дополнительных действий. Данная глава не претендует на исчерпывающее руководство по их использованию, она призвана помочь читателю комфортно разрабатывать свои первые программы на PHP.

Очень сложно дать ответ на вопрос, какой из редакторов лучше всего использовать, т. к. это личное предпочтение каждого. Наподобие того, как каждому человеку нравятся разные машины, так же и редакторы: у различных людей разные предпочтения. Для того чтобы убедиться в этом, достаточно зайти на любой более менее серьезный форум, посвященный PHP, и посмотреть топики, посвященные обсуждению редакторов. Такая тема встречается практически на каждом форуме, если же у вас не получается найти ее,

Блокнот рекомендуется использовать лишь в крайнем случае, например, когда нужно внести изменения в код очень срочно, а других средств под рукой нет.

Часто можно встретить синоним термина "редактор" — IDE (Integrated Development Environment, интегрированная среда разработки).

то всегда можно создать свою собственную. Вы удивитесь, насколько разные предпочтения у людей.

К тому же вы можете посетить следующие два сайта, на которых вы найдете информацию по самым разнообразным редакторам:

- **http://www.thelinuxconsultancy.co.uk/phpeditors.php**
- **http://www.php-editors.com/review/**

3.1. PHP Expert Editor

Разработчик: Ankord Development Group.

Официальный сайт: http://www.ankord.com/ (русскоязычный вариант доступен по адресу **http://www.ankord.com/ru**).

Размер дистрибутива: 3,11 Мбайт.

Платформа: Windows 98/ME/NT/2000/XP.

Минимальные требования к компьютеру: процессор — Pentium100, оперативная память — 64 Мбайт, свободного пространства на винчестере — 4 Мбайт.

Дополнительная информация: программа нуждается в регистрации (незарегистрированную копию можно использовать в течение 30 дней). Для регистрации достаточно зайти по адресу **http://www.ankord.com/ru/phpxedit_reg.html** и заполнить регистрационную форму, в течение нескольких дней вы получите по электронной почте регистрационный ключ.

PHP Expert Editor — удобный в использовании PHP-редактор или IDE (Integrated Development Environment, интегрированная среда разработки), созданный специально для PHP-мастеров. Будет удобен как начинающим, так и профессиональным PHP-программистам. Имеет встроенный HTTP-сервер и отладчик для запуска и отладки PHP-скриптов (вы можете использовать также внешний HTTP-сервер, на мой взгляд, это даже удобнее). Предоставляет возможность проверки синтаксиса PHP, имеет встроенный браузер, FTP-клиент, файл-эксплорер, настраиваемые шаблоны кода, три режима подсветки кода (PHP & HTML, HTML only, PHP only), существует подсветка Java Script и CSS-файлов, функции быстрой навигации в PHP-коде. Поддерживает следующие форматы файлов: Windows, UNIX, Mac.

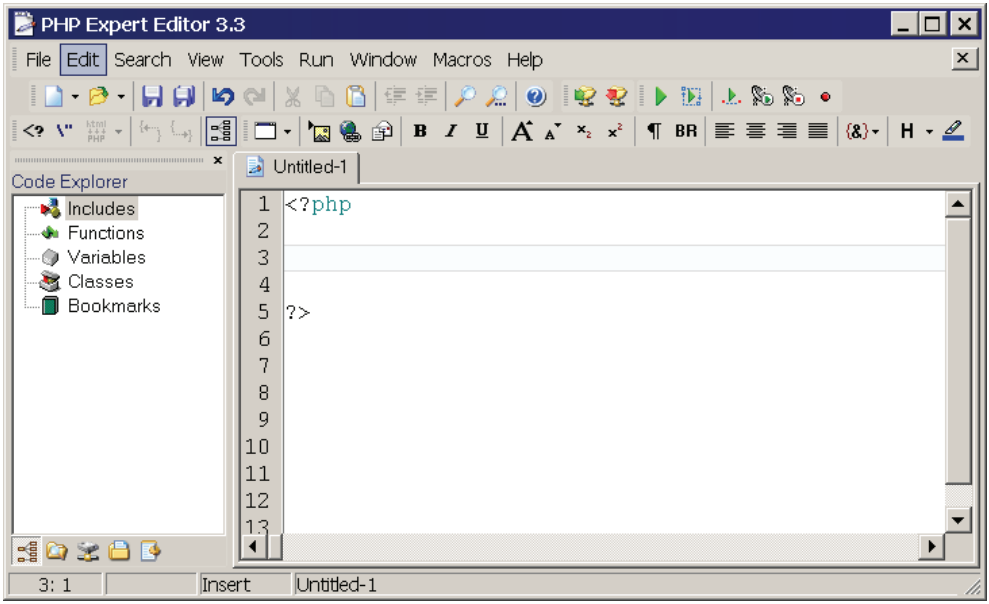


Рис. 3.1. Окно **PHP Expert Editor 3.3**

Рис. 3.2. Окно **About PHP Expert Editor**

К дополнительным возможностям можно отнести быструю вставку всех функций PHP с отображением подсказки и их параметров; подсветку парных символов (скобок, кавычек); два стиля интерфейса — Classic и Office XP.

На рис. 3.1 и 3.2 вы можете увидеть окно программы **PHP Expert Editor 3.3** и окно **About PHP Expert Editor**.

Несколько советов, которые помогут при работе с PHP Expert Editor.

- *Очень удобно использовать **Code Templates** (Шаблоны кода). Есть набор предопределенных шаблонов, но вы также можете создавать свои собственные. Предположим, вам надо будет вставить условную конструкцию* **IF-Else**, *вы просто нажимаете <Ctrl>+<J> и выбираете ее из появившегося окошка. Для того чтобы создать свой собственный шаблон, нужно выбрать пункт меню **Tools | Code Templates**.*
- *Для того чтобы вызвать окно настройки программы — **Properties**, необходимо выбрать пункт меню **View | Editor Options**.*
- *Для того чтобы включить функцию автосохранения, необходимо в окне **Properties** перейти на вкладку **Advanced** и установить флаг **Auto save every**, после этого в поле, расположенном напротив, указать в минутах значение, которое будет указывать, через какой промежуток времени будет выполняться автоматическое сохранение файла, с которым вы работаете.*
- *Для смены стиля интерфейса достаточно выбрать пункт меню **View | Interface Style**, а затем выбрать необходимый вам стиль.*

3.2. PHP Designer 2006

Разработчик: MPSOFTWARE.

Официальный сайт: http://www.mpsoftware.dk/phpdesigner.php.

Размер дистрибутива: 1,7 Мбайт.

Платформа: Windows 98/ME/NT/2000/XP.

Минимальные требования к компьютеру: оперативная память — 32 Мбайт; свободного пространства на винчестере — 3,5 Мбайт.

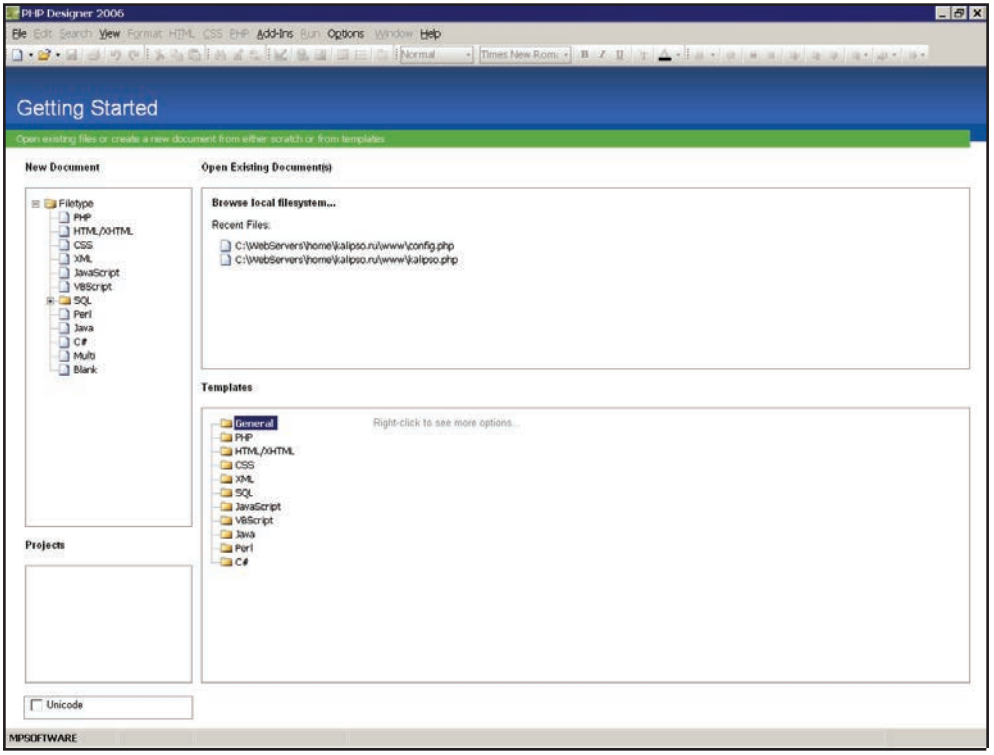
Не важно, являетесь ли вы профессионалом или новичком в PHP. Бесплатный редактор PHP Designer 2006 обеспечит вас полным набором инструментов, которые необходимы для разработки программ на PHP.

PHP Designer 2006 позволяет подсвечивать код для PHP, HTML, XHTML, CSS, Perl, C#, JavaScript, VB, Java и SQL (Ingres, InterBase, MSSQL, MySQL, Oracle, SyBase и Standard SQL), что очень удобно, т. к. вы можете использовать его не только при разработке PHP-программ, но и в других ситуациях. Например, при построении SQL-запросов к различным СУБД. Этот редактор очень универсален, так сказать "все в одном".

Он содержит библиотеку, разделенную по категориям, для большинства команд, относящихся к PHP, HTML/XHTML, CSS, SQL. Имеет встроенный FTP-клиент, браузер и файл-эксплорер. Содержит встроенную систему напоминания ToDO и менеджер проектов. Обладает настраиваемым интерфейсом и содержит более 10 встроенных тем оформления. Оболочка PHP Designer очень удобна и практична.

На рис. 3.3 и 3.4 вы можете увидеть окно программы **PHP Designer 2006** и окно **About**.

Рис. 3.3. Окно **PHP Designer 2006**



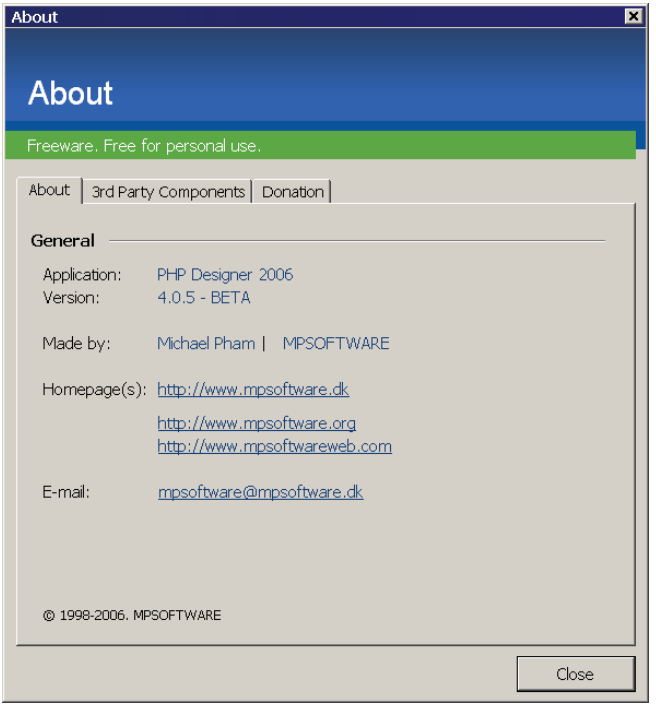


Рис. 3.4. Окно **About** программы PHP Designer 2006

При первом запуске PHP Designer 2006 вы попадете в насыщенную различными элементами среду разработки (см. рис. 3.3). Для некоторых это может быть и недостатком, т. к. поначалу может запутать, но на самом деле через какое-то время вы достаточно свободно будете чувствовать себя в этой среде.

Несколько советов, которые помогут при работе с PHP Designer 2006.

- *Для того чтобы получить доступ к библиотеке команд, нужно выбрать пункт меню **View | Panels | Code Libraries**, после этого можно выбирать команды, которые необходимо вставить в программу (рис. 3.5).*
- *В PHP Designer 2006 есть набор вспомогательных панелей, которые предоставляют доступ к дополнительным возможностям редактора. Чтобы их включить, нужно выбрать пункт меню **View | Panels** и имя необходимой панели.*
- *Режим прозрачности (Transparent mode). Если вы хотите немного повеселиться, то режим прозрачности для вас, я был удивлен, когда обнаружил эту функцию и*

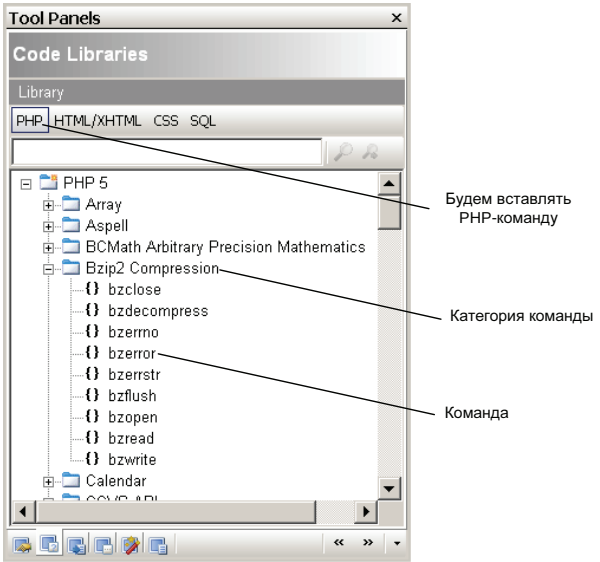
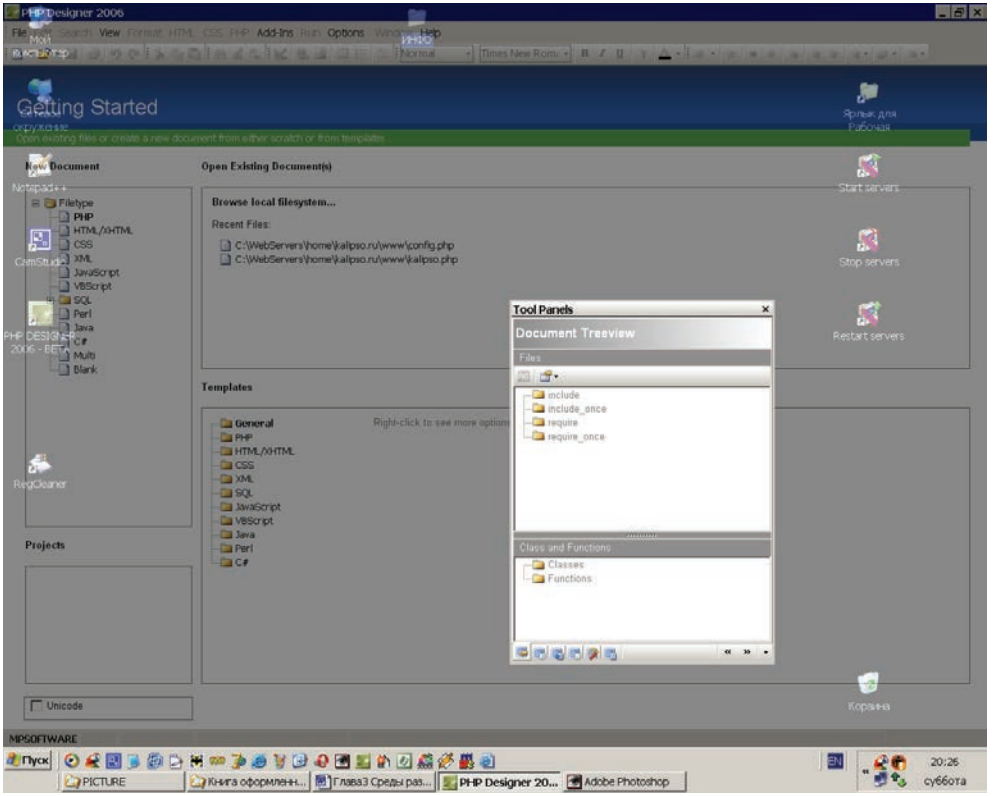


Рис. 3.5. Панель **Code Libraries**

Рис. 3.6. Режим прозрачности в действии



другого логического объяснения ей не нашел. Данный режим включается посредством пункта меню ***Options | Interface | Transparency | Transparent Mode****. Когда режим включен, то становится доступным пункт меню* ***Options | Interface | Transparency | Transparent Value****, где и задается коэффициент прозрачности, который может быть в районе от* ***255*** *до* ***0****, чем он меньше, тем более прозрачно окно* ***PHP Designer 2006****. Таким образом,* ***PHP Designer 2006*** *практически можно сделать обоями для рабочего стола, только эти обои будут работать (рис. 3.6).*

- *Для смены внешнего вида PHP Designer 2006 необходимо выбрать пункт меню* ***Options | Interface****, после чего появится выпадающий список, в котором вы можете выбрать вариант оформления на свой вкус.*
- *Очень удобен пункт меню* ***HTML | Forms****, который позволяет достаточно быстро создавать некоторые элементы вашего сайта при минимуме написания кода. Например, выбрав пункт* ***HTML | Forms | Form****, задав необходимые параметры в появившемся окне и нажав кнопку* ***OK****, будет автоматически сформирован код для создания формы.*

3.3. NOTEPAD++

Автор: Don Ho.

- **Официальный сайт: http://notepad-plus.sourceforge.net**.
- **Размер дистрибутива:** 956 Кбайт.
- **Платформа:** Windows.
- **Минимальные требования к компьютеру:** свободного пространства на диске 2,3 Мбайт.

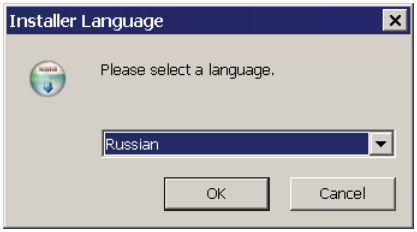


Рис. 3.7. Первое окно, которое появляется при установке программы Notepad++

Данный редактор может использоваться как альтернатива обычному блокноту, только в более продвинутом его варианте. Конечно, до самого удобного редактора программа явно не дотягивает, зато обладает самым маленьким по размеру дистрибутивом и является достаточно универсальной, т. к. позволяет не только разрабатывать программы на PHP.

Первый приятный сюрприз ждет вас при установке программы (рис. 3.7).

Как видно, данная программа поддерживает русский язык, это значит, что у вас будет дополнительное удобство при работе с ней. Очень порадовало качество выделения кода программы, используется очень удачное сочетание цвета и шрифтов для различных элементов кода. С помощью кнопок с изображением лупы с плюсом и минусом можно масштабировать текст кода, что тоже удобно. Если необходимо сделать, чтобы большая часть кода помещалась на экране, уменьшаем, если необходимы более крупные символы, увеличиваем.

Notepad++ поддерживает различные форматы и кодировки. Большое количество вариантов подсветки синтаксиса программы для совершенно различных языков (в этом отношении он смело может потягаться с PHP Designer 2006).

На рис. 3.8 и 3.9 вы можете увидеть окно **Notepad++** и окно **About**.

Несколько советов, которые помогут при работе с Notepad++.

- *Для того чтобы указать программе тип файла, с которым вы хотите работать, необходимо выбрать пункт меню* ***Форматы****.*
- *Для того чтобы выбрать вариант подсветки синтаксиса кода программы, необходимо воспользоваться пунктом меню* ***Синтаксис****.*
- *Вы можете задать цвета и шрифт подсветки для различных вариантов синтаксиса самостоятельно, для этого необходимо воспользоваться пунктом меню* ***Синтаксис | Определение стилей****.*
- *Вы можете скачать дополнительные модули* ***(Plugins)*** *с сайта и установить их.*
- *При работе с Notepad++ вы можете использовать функцию сохранения сессии. Здесь под сессией понима-*

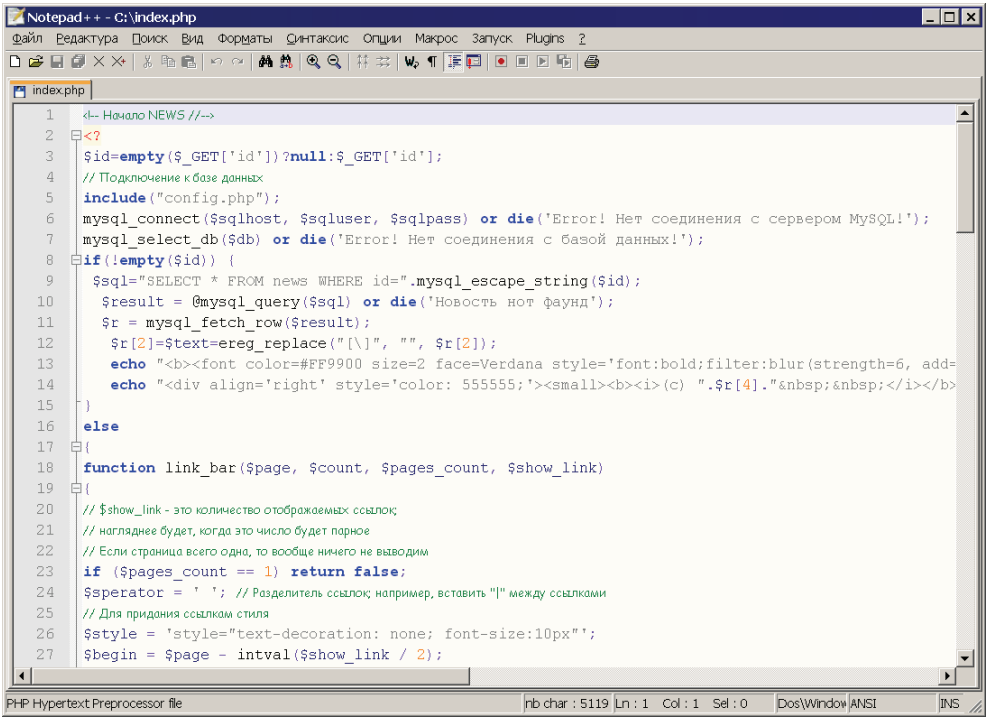


Рис. 3.8. Окно **Notepad++**

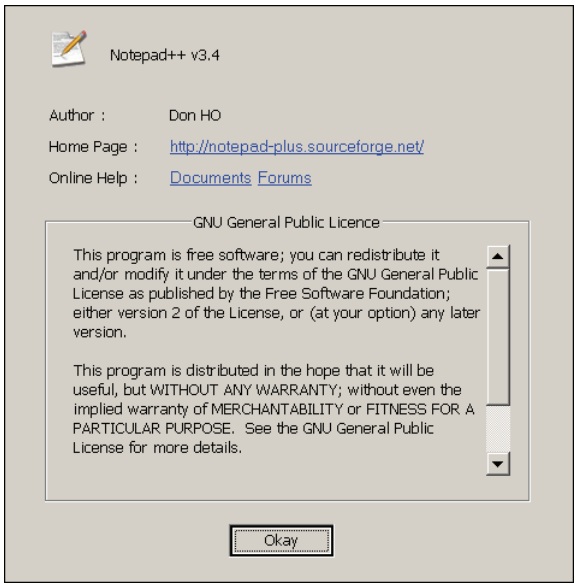


Рис. 3.9. Окно **About** программы **Notepad++**

*ется ваша работа в редакторе, т. е. вы открыли, скажем, 5 файлов, поработали с ними, потом сохраняете вашу рабочую сессию (**Файл | Save Session**) и закрываете редактор. Когда вы заново откроете редактор, вам достаточно будет выбрать пункт меню **Файл | Load Session**, и файлы, с которыми вы работали ранее, сразу загрузятся в редактор.*

*Есть возможность замены иконок кнопок в панели инструментов, для этого нужно скачать набор иконок с официального сайта (выберите раздел сайта **Download** и найдите заголовок **Theme Package**). После этого вам нужно распаковать архив и выполнить ряд действий, описанных в файле readme.txt, например если установить набор под названием **Notepad++ Theme Demo Package**, то можно будет увидеть следующий результат — рис. 3.10.*

Рис. 3.10. Набор иконок под названием **Notepad++ Theme Demo Package**

Глава 4
Джентльменский набор Web-разработчика

Как вы уже знаете, для выполнения скриптов и их отладки необходим Web-сервер, который смог бы принимать запросы пользователя и отправлять ему соответствующие ответы. Для того чтобы Web-сервер мог выполнять скрипты, написанные именно на PHP, необходимо установить PHP. Если вы захотите работать с базой данных, а мы будем работать с ней (в дальнейшем вы убедитесь, что использование баз данных упрощает реализацию многих задач), то вам придется установить СУБД (Систему Управления Базами Данных).

Одним из самых распространенных Web-серверов является Apache, а одной из самых распространенных СУБД, предназначенных для реализации Web-задач, является MySQL. Именно эти два программных продукта наиболее часто используют в связке с PHP. Дело в том, что они достаточно давно развиваются в очень тесном сотрудничестве друг с другом, направленном на улучшение положительных сторон каждого из них. Таким образом, при соответствующей настройке стабильность и скорость работы вашего сайта будет на должном уровне.

Для того чтобы приступить к установке, нужно сначала скачать (только не торопитесь пока это делать) соответствующие дистрибутивы. В табл. 4.1 приводится необходимая информация.

Таблица 4.1. *Дистрибутивы для установки Web-сервера с поддержкой PHP и MySQL*

Название программы	**Официальный сайт (откуда можно скачать дистрибутив)**	**Размер дистрибутива**
Apache (Web-сервер)	www.apache.org	4,9 Мбайт
PHP	www.php.net	8,9 Мбайт
СУБД MySQL	www.mysql.com (справка на русском языке доступна на www.mysql.com/doc/ru/index.html)	16,9 Мбайт
phpMyAdmin	www.phpmyadmin.net	3,4 Мбайт
	Итого:	34,1 Мбайт

Замечание

phpMyAdmin — популярное средство, позволяющее сделать процесс работы с базой данных очень удобным, без него вам бы пришлось осуществлять большинство действий (например, создание таблиц) в командной строке, что очень неудобно.

Замечание

Пожалуй, единственным недостатком Денвера является то, что он работает только под Windows.

Не удивительно, если некоторых испугает итоговая цифра. Я думаю, не всем захочется качать такой объем данных через Интернет (особенно по модему), к тому же все это придется настраивать вручную. То есть не в лучших традициях Windows-приложений, когда запустил установочный файл, пару раз нажал кнопку **Next**, и все готово. Придется открывать конфигурационные файлы и прописывать настройки ручками. Конечно, для опытного человека это не составит особого труда, но для новичка, который всего лишь хочет написать небольшую программу, это может показаться утомительным и занять достаточное количество времени. К тому же желание заниматься PHP может запросто пропасть.

Я сторонник того подхода, что всегда надо начинать с более простых вещей. В данном случае есть более простой выход, под названием *Денвер*. С помощью него вы можете установить Apache, PHP, MySQL, phpAdmin буквально за пару минут, при этом размер дистрибутива, который вам придется скачать, будет равен 3 Мбайт.

На рис. 4.1 вы можете увидеть логотип Денвера.

Рис. 4.1. Логотип Денвера

На официальном сайте **www.denwer.ru** он характеризуется как:

"Джентльменский набор Web-разработчика ("Д.н.w.р" читается "Денвер" — почти как название города) — набор дистрибутивов, используемый Web-разработчиками (программистами и дизайнерами) для отладки сайтов на "домашней" (локальной) Windows-машине без необходимости выхода в Интернет".

Денвер устанавливается в один-единственный каталог и вне его ничего не изменяет. Он не пишет файлы в Windows-директорию и не прописывает

никакой информации в реестре. При желании вы можете даже поставить себе сразу два Денвера, и они не будут конфликтовать.

- Никакие "сервисы" NT/2000 не "прописываются". Если вы запустили Денвер, то он работает. Если завершили — то перестает работать, не оставляя после себя следов.
- Системе не нужен деинсталлятор — достаточно просто удалить каталог.
- Все конфигурирование и настройка под конкретную машину происходит автоматически.

4.1. Установка Денвера

Для начала необходимо скачать последнюю версию дистрибутива, для этого достаточно зайти по ссылке **http://www.denwer.ru/dis/latest** и скачать предложенный файл.

После того как файл будет закачан, можно приступить к установке Денвера. Файл представляет собой самораспаковывающийся архив, поэтому выполните на нем двойной щелчок левой кнопкой мыши, после чего появится окно **Инсталлятор** (рис. 4.2).

Нажимаем **Да**, архив начнет распаковываться, этот процесс отображается в окне, изображенном на рис. 4.3.

По окончании распаковки вы увидите следующее окно (рис. 4.4).

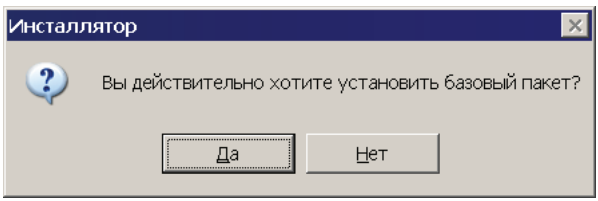


Рис. 4.2. Окно, которое появляется после запуска файла установки Денвера

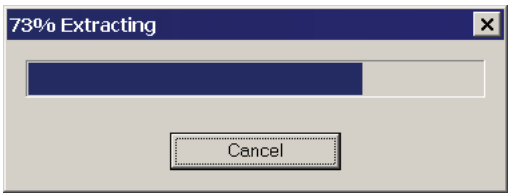


Рис. 4.3. Процесс распаковки дистрибутива

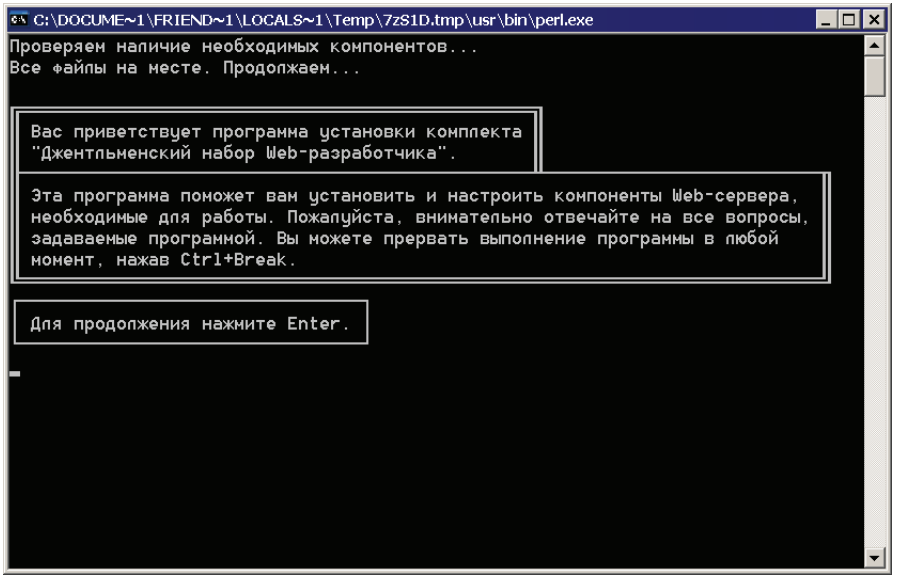


Рис. 4.4. Окно приветствия установщика Денвера

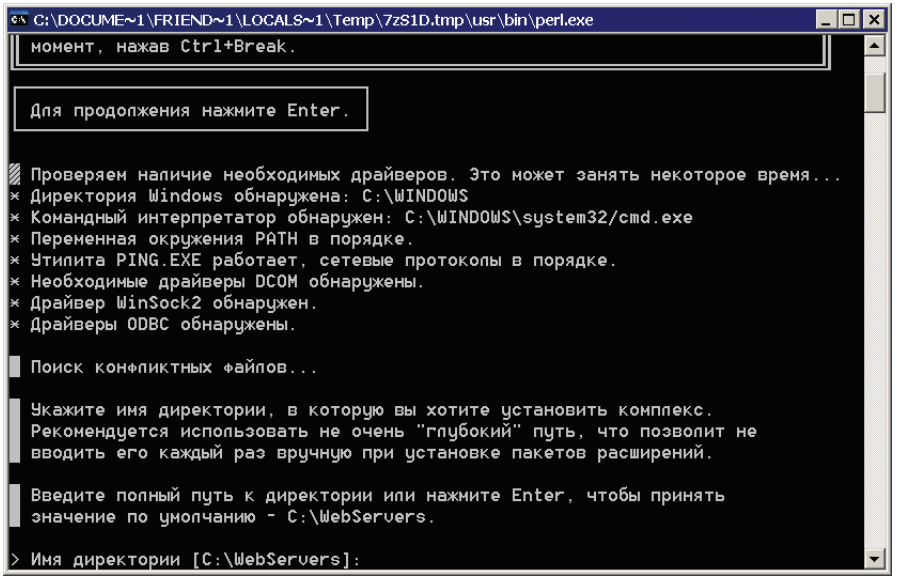


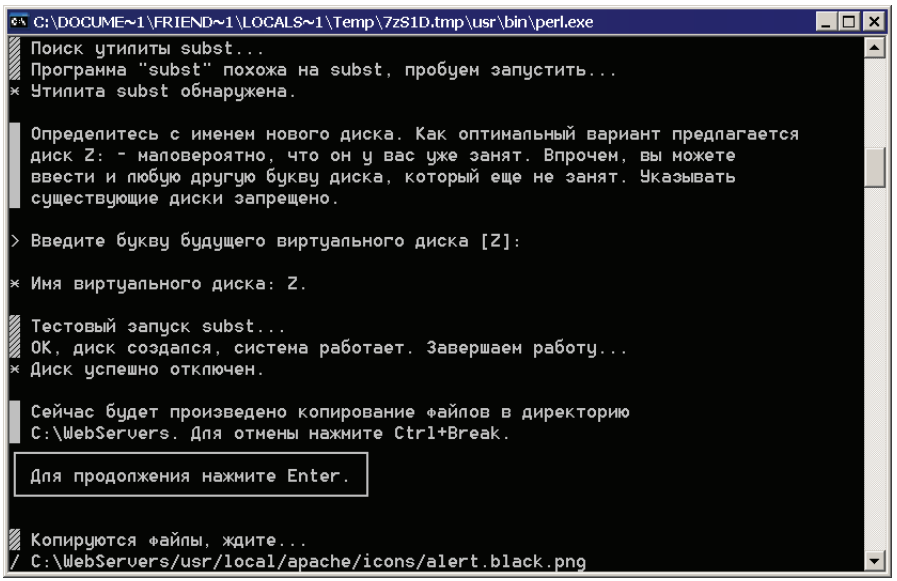
Рис. 4.5. Выбор пути, куда будет установлен Денвер

Как видите, мы погрузились во времена DOS-программ, но зато, пожертвовав красочностью Windows-приложений, у нас есть дистрибутив небольшого размера, который удобно качать по сети. Денвер написан на Perl, это вы можете заметить по заголовку окна (заголовок достаточно длинный, но в его конце можно увидеть perl.exe). Нажимаем <Enter>, как нам и предлагают. Далее видим следующее — рис. 4.5.

Здесь нам предлагается ввести путь и имя директории, в которую будет установлен Денвер, а вместе с ним и PHP, Apache, MySQL, phpMyAdmin. По умолчанию предлагается путь C:\WebServers, советую оставить его. Далее нажимаем <Enter>. После чего нам будет задан вопрос, действительно ли мы уверенны в том, что хотим установить Денвер в указанную на предыдущем шаге папку. Нажмите клавишу <Y> и затем <Enter>.

Теперь нас информируют о следующем шаге установки — выбор буквы виртуального диска, который будет появляться при запуске Денвера. Нажмите <Enter>. После чего вы можете ввести любую букву или согласиться с предложенной — **Z**, я советую сделать выбор в пользу последнего варианта. Нажмите <Enter>. Все готово к копированию файлов Денвера в указанный вами путь, необходимо еще раз нажать <Enter>, чтобы приступить к этому процессу (рис. 4.6).

Рис. 4.6. Процесс копирования файлов



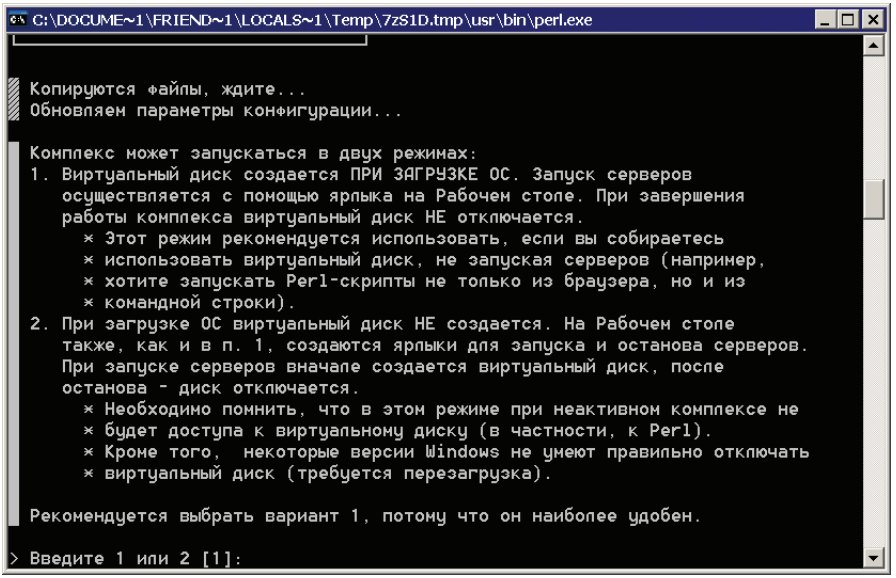


Рис. 4.7. Выбор варианта запуска Денвера

После того как копирование завершится, вам будет предложено выбрать режим запуска Денвера, возможны следующие варианты (рис. 4.7):

- *автоматический* — комплекс будет запускаться автоматически при входе в ОС;
- *ручной* — чтобы начать работу с комплексом, необходимо будет запустить его с помощью специального ярлыка, по завершении работы производится остановка комплекса также с помощью специального ярлыка.

Предлагаю выбрать второй вариант, как более удобный, т. к. в этом случае вы будете запускать комплекс только тогда, когда он вам нужен. Нажмите <2>. Далее вас спросят: "Хотите ли вы, чтобы на рабочем столе были размещены ярлыки, предназначенные для работы с комплексом?" Рекомендую согласиться и нажать <Y>. После этого процедура установки будет закончена. Нажмите <Enter>. Окно установки закроется, и вы увидите на рабочем столе три новых ярлыка (рис. 4.8):

- **Start servers** — запуск комплекса;

Рис. 4.8. Слева направо: первый ярлык для запуска комплекса, второй для остановки, третий для перезапуска

- **Stop servers** — остановка работы комплекса;
- **Restart servers** — перезапуск комплекса (т. е. сначала будет осуществлена остановка комплекса, а потом сразу его запуск).

После установки Денвера вам будут доступны: Apache, PHP, MySQL, phpMyAdmin, Perl.

Вообще PHP может быть установлен двумя разными способами:

- *как внешняя программа* — при этом каждый раз, когда пользователь запрашивает PHP-скрипт, на сервере запускается новый процесс в лице программы php-cgi.exe;
- *как модуль Web-сервера* — в этом случае PHP будет работать как одно целое вместе с Web-сервером.

У каждого из вариантов есть свои плюсы и минусы. Первый вариант позволяет добиться большей безопасности для Web-сервера, второй обеспечивает лучшее быстродействие при выполнении скриптов, а также позволяет получить доступ к дополнительным возможностям PHP, но более сложен в настройке по сравнению с первым. Установка Денвера предполагает, что PHP будет работать как модуль Web-сервера Apache.

Стоит также отметить, что если на вашем компьютере установлен Firewall, то для корректной работы Web-сервера необходимо прописать в нем соответствующие правила (или установить разрешения). Если вы используете Windows XP и у вас установлен Microsoft IIS, то рекомендуется остановить его работу, перед тем как запускать Денвер.

Последним советом, перед тем как приступить к работе с Денвером, будет такой: когда вы тестируете или отлаживаете ваши программы, т. е. у вас запущен Денвер, не рекомендуется находиться в Интернете или быть подключенным к локальной сети, т. к. иначе ваша система может быть объектом атаки. Если для вас это не приемлемо, т. е. вам нужна связь с внешним миром, тогда настройте соответствующим образом ваш Firewall.

4.2. Работаем с Денвером

Для того чтобы запустить Денвер (запустить ваш персональный Web-сервер), необходимо выполнить двойной щелчок левой кнопкой мыши на ярлыке **Start servers**. После чего вы увидите окно запуска, оно будет отображаться несколько секунд, потом исчезнет, а в трее появится пиктограмма в виде пера, обозначающая работу Web-сервера Apache на вашей системе (рис. 4.9 и 4.10).

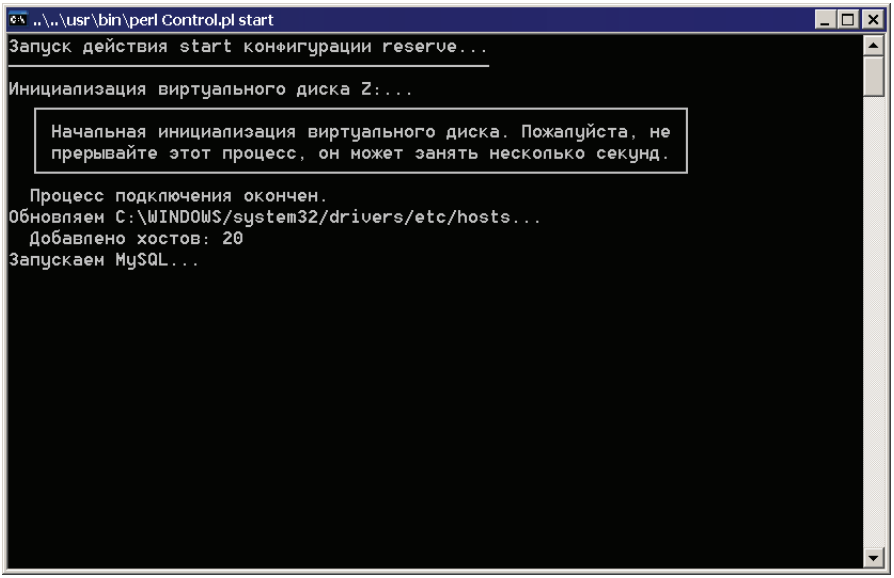


Рис. 4.9. Окно запуска Денвера

Рис. 4.10. Значок в трее, характеризующий работу Web-сервера Apache на вашем компьютере

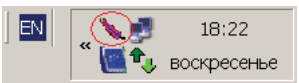
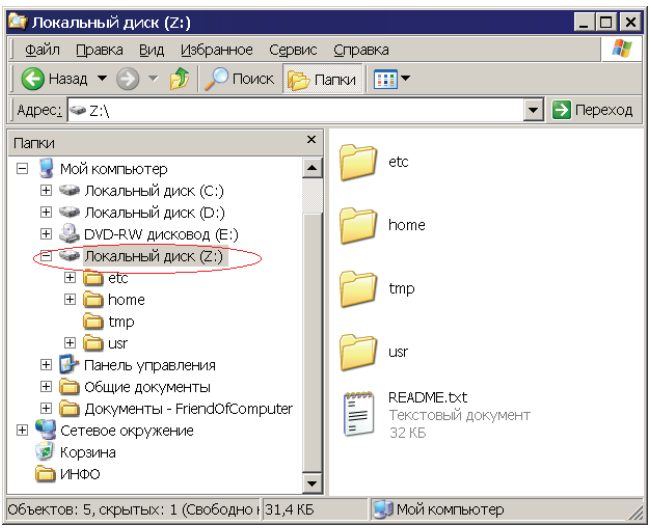


Рис. 4.11. Виртуальный диск Z:, появившийся после запуска Денвера



После запуска Денвера на вашем компьютере будет создан специальный виртуальный диск, если вы не сменили его имя при установке Денвера, то это будет диск Z: (рис. 4.11).

Когда вы хотите закончить работу с Web-сервером, например, вам нужно перезагрузить компьютер, рекомендуется воспользоваться ярлыком **Stop servers**, чтобы остановить Apache.

Ярлык **Restart servers** осуществляет перезапуск Web-сервера, это действие необходимо, когда вы совершили некоторые изменения и хотите, чтобы они вступили в силу. Под изменениями могут подразумеваться создание нового виртуального хоста (или другими словами: добавление нового сайта, например, primer.ru), внесение изменений в конфигурацию Apache, PHP или MySQL.

Ярлыки, которые мы только что рассмотрели, представляют собой систему управления запуском и завершением работы комплекса Денвер или, другими словами, вашего персонального Web-сервера.

Итак, у вас запущен Web-сервер, предлагаю открыть браузер и набрать в нем localhost, после этого нажмите <Enter>. Вы увидите следующее — рис. 4.12.

Выберите пункт **Утилиты**, вы увидите утилиты и инструменты, которые включает в себя Денвер (рис. 4.13).

Замечание

*Ярлыки **Start Servers**, **Stop Servers** и **Restart Servers** ссылаются на файлы Start.exe, Stop.exe, Restart.exe, находящиеся в папке WebServers\etc\.*

Рассмотрим их более подробно:

- **DNSearch: поиск файлов** — аналог поисковых систем в Интернете, например, таких как Rambler или Google, т. е. вы вводите искомую фразу и устанавливаете условия поиска, после чего нажимаете кнопку **Искать**;
- **Список зарегистрированных сайтов** — данный инструмент позволяет отобразить в виде списка все существующие сайты на вашем Web-сервере;

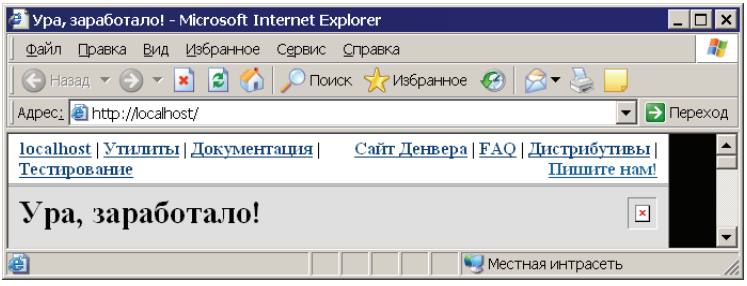


Рис. 4.12. Тестовая страничка Денвера

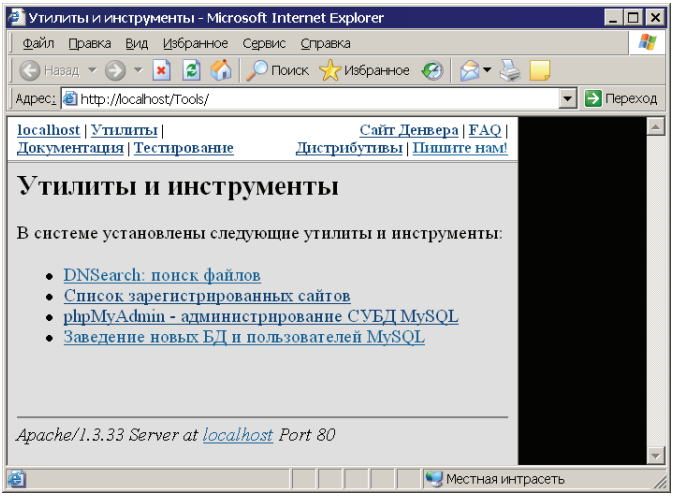


Рис. 4.13. Утилиты и инструменты Денвера

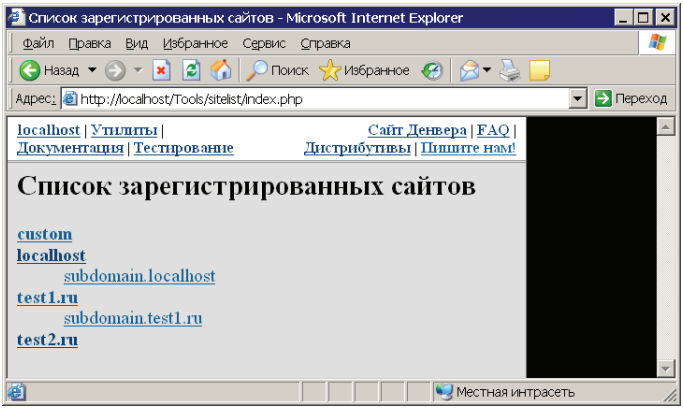


Рис. 4.14. Список зарегистрированных сайтов

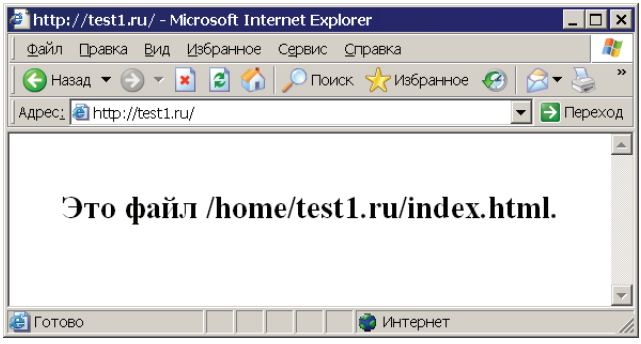


Рис. 4.15. Тестовая страничка

- **phpMyAdmin - администрирование СУБД MySQL** — это система администрирования MySQL через Web-интерфейс. Позволяет создавать базы данных, таблицы, выполнять к ним запросы и т. д. (О работе с этим инструментом мы поговорим в *главе 10.*);
- **Заведение новых БД и пользователей MySQL** — название данного инструмента говорит само за себя.

Выберите пункт **Список зарегистрированных сайтов**, появится список зарегистрированных в системе сайтов (рис. 4.14).

Выберите **test1.ru**, вы должны увидеть следующее — рис. 4.15.

Поздравляю! Вы только что протестировали работоспособность установленного комплекса Денвер.

4.3. Денвер изнутри

В официальной документации к Денверу сказано, что он "похож на небольшой UNIX". И это на самом деле верно, т. к. организация каталогов создана по подобию этой операционной системы. Как вы уже знаете, после запуска Web-сервера у вас появится в системе виртуальный диск (как правило, с именем Z:), предлагаю открыть его в проводнике и посмотреть, что на нем располагается (см. рис. 4.11). Рассмотрим назначение каждой папки:

- etc — хранит скрипты конфигурирования Денвера, здесь же находится его конфигурационный файл CONFIGURATION.txt, все его параметры обозначены русскими комментариями, в целях эксперимента можно посмотреть содержимое данного файла, но править его вручную не рекомендуется, т. к. это может негативно сказаться на работе всего комплекса;
- home — с данной папкой вы будете часто работать, т. к. в ней будут храниться ваши сайты (или виртуальные хосты);
- tmp — папка предназначена для хранения временной информации, такой как пользовательская сессия (о сессиях мы поговорим в *разд. 9.2.6*). В данной папке, вернее, в ее подкаталоге !sendmail будут храниться все письма, отправленные с помощью функции `mail()` (работу с данной функцией мы рассмотрим в *разд. 9.1*);
- usr — системная папка, в которой хранятся Apache, PHP, MySQL и некоторые другие элементы.

Файл README.txt в корне виртуального диска, который появляется после запуска комплекса, содержит краткую информацию по Денверу.

4.4. Создаем свой сайт

По умолчанию, после установки Денвера, у вас будут доступны несколько виртуальных хостов, например, если вы наберете в браузере **www.test1.ru**, то у вас отобразится тестовая страничка (см. рис. 4.15).

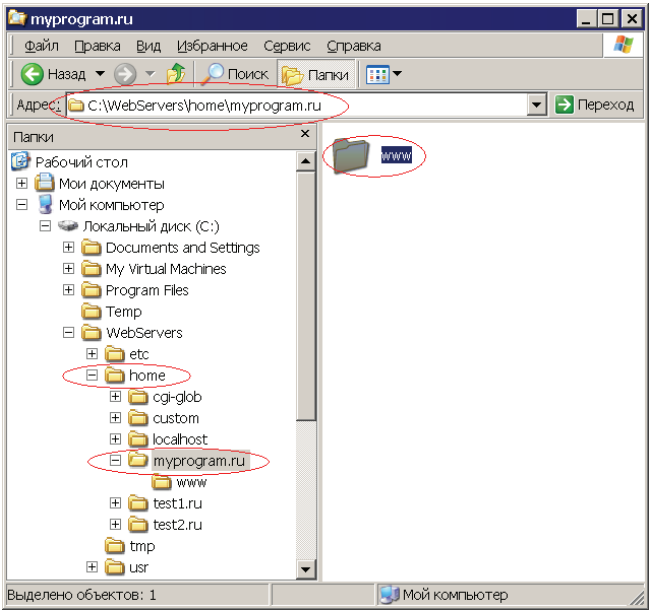


Рис. 4.16. Создание собственного виртуального сайта

Для того чтобы добавить новый сайт, достаточно создать в каталоге home папку с именем будущего сайта, только ее имя должно быть без www. Например, вы хотите создавать ваши программы, работая с виртуальным сайтом **www.myprogram.ru**, для этого вам необходимо создать папку myprogram.ru в каталоге home. Но это еще не все, внутри вновь созданной папки вам нужно создать еще одну с именем www (рис. 4.16).

После этого останется только перезапустить Web-сервер с помощью ярлыка **Restart servers**, и ваш новый сайт будет готов к работе.

4.5. Конфигурационные файлы

Замечание

Удобство Денвера заключается еще и в том, что большинство параметров конфигурационных файлов переведено на русский язык, таким образом, процесс настройки упрощается.

Каким бы программным обеспечением вы не пользовались, если вы знаете, каким способом оно настраивается, то вам намного легче разобраться в нестандартных ситуациях, которые могут возникнуть при использовании данного ПО. Основное средство настройки Apache, PHP и MySQL — это специальные конфигурационные файлы (далее мы рассмотрим, где их можно найти). Все они спокойно открываются с помощью обычного блокнота. Конечно,

Рис. 4.17. Пиктограмма конфигурационного файла MySQL

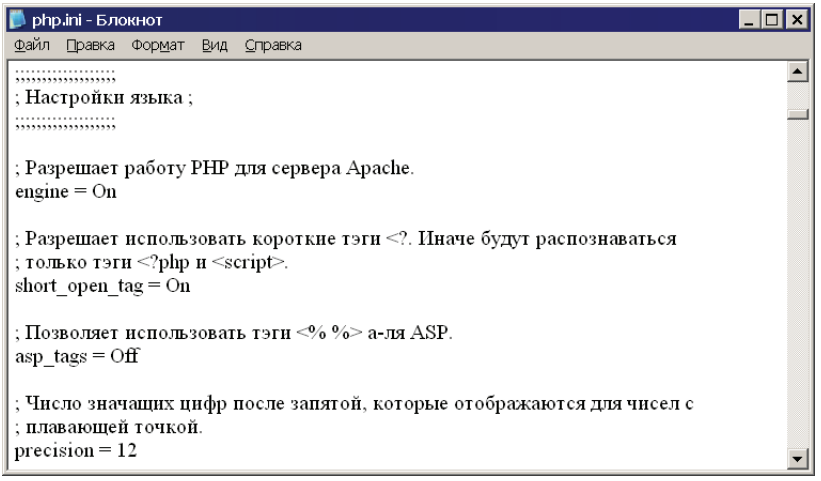


Рис. 4.18. Файл конфигурации PHP — php.ini, открытый в Блокноте

никаких изменений в настройки комплекса мы вносить не будем, но эта информация может вам пригодиться в будущем.

- etc\CONFIGURATION.txt — файл конфигурации Денвера (не рекомендуется вносить изменения в этот файл).
- usr\local\apache\conf\httpd.conf — файл конфигурации Apache.
- usr\local\php\php.ini — файл конфигурации PHP.
- usr\local\mysql4\my.cnf — файл конфигурации MySQL, он имеет свою иконку (рис. 4.17), но так же, как и все предыдущие файлы, спокойно открывается в текстовом редакторе.

На рис. 4.18 вы можете видеть файл конфигурации PHP под названием php.ini, открытый в Блокноте.

4.6. Информация о PHP

Сейчас мы напишем небольшую программу. Создайте файл info.php следующего содержания:

```
<?php
phpinfo();
?>
```

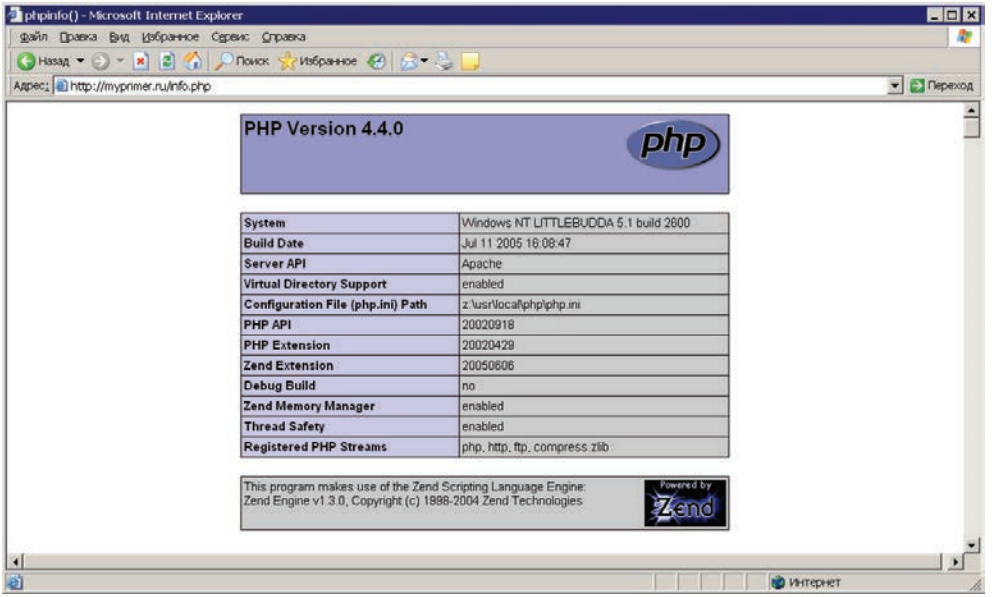


Рис. 4.19. Информация о PHP

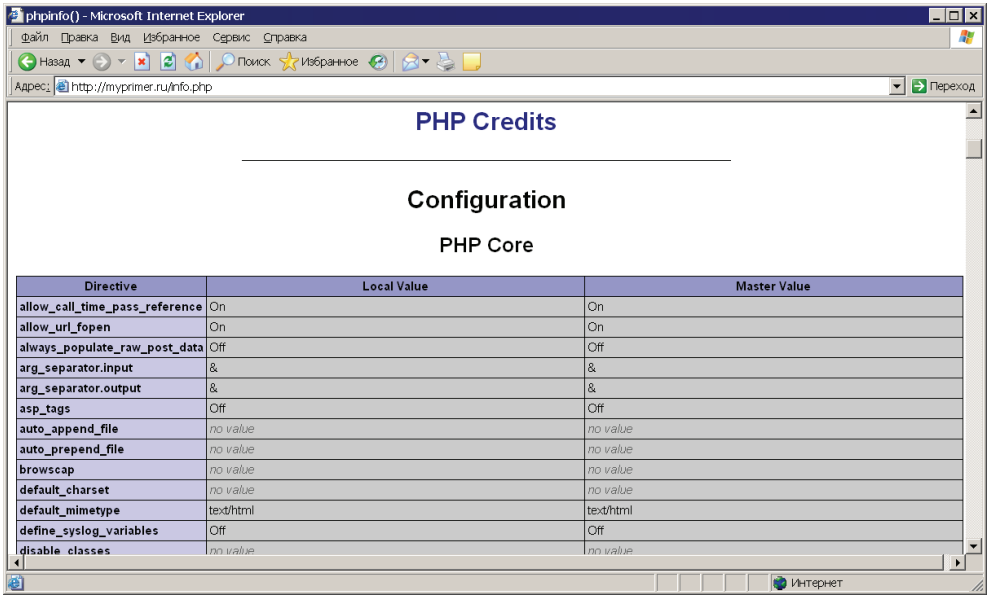


Рис. 4.20. Информация о ядре PHP

Теперь сохраните этот файл на вашем виртуальном сайте **myprimer.ru** в папке www, откройте браузер и укажите в нем адрес **myprimeer.ru/info.php**, вы увидите следующий результат — рис. 4.19.

Как видите, с помощью специальной команды `phpinfo()` мы можем получить системную информацию о Apache, PHP и MySQL. Если воспользоваться вертикальной линейкой прокрутки, то можно увидеть следующее — рис. 4.20.

Большинство настроек вам будут непонятны, но после прочтения данной книги ситуация во многом прояснится, т. к. вы будете не раз встречаться с ними.

*Несмотря на то, что последней версией PHP является 5, и для нее имеется Денвер, дистрибутив которого доступен по адресу **http://www.denwer.ru/dis/Base_PHP5/latest**, с ним наблюдаются некоторые проблемы, например, начинают капризничать Mambo и phpBB и не хотят запускаться, а файл php.ini не имеет русских комментариев. Будем надеяться, что со временем эта ситуация изменится. К тому же у большинства хостинг-провайдеров установлен PHP 4-й версии, и они пока не торопятся его менять.*

Глава 5
Наши первые программы на PHP

Прежде чем приступить к процессу разработки программ, предлагаю вам создать новый виртуальный сайт, например **kodding.ru** (данный процесс был рассмотрен в *разд. 4.4*), где вы будете сохранять примеры из данной главы.

5.1. Программируем — начнем с простого

Давайте напишем текст программы first_prog.php следующего содержания — листинг 5.1.

Листинг 5.1. Файл first_prog.php

```
<!-- HTML-теги -->
<html>

<head>
  <title>Моя первая программа на PHP</title>
</head>

<body>

<!--Далее идут команды на PHP -->
<?php
//Выводим на экран строку, заключенную в кавычки
echo "Сегодня чудесный день.";
//Выводим на экран строку, заключенную в кавычки
echo "Скоро я стану профи в PHP!";
?>
```

```
17 <!--Конец команд на PHP -->
18
19 <!-- HTML-теги -->
20 </body>
21 </html>
```

Нумерация строк кода выполнена для удобства, в исходном коде программы ее быть не должно.

На рис. 5.1 можно увидеть, как будет выглядеть скрипт в окне редактора PHP Expert Editor 3.3.

Прежде чем запускать программу с помощью браузера, давайте рассмотрим для чего предназначена каждая ее команда. В первой строчке находятся HTML-комментарии, до PHP мы пока еще не дошли. Зачем вообще нужны комментарии? Они играют важную роль в любой программе, т. к. помогают описывать отдельные участки кода. Обратимся к примеру, предположим, к вам в руки попала программа на доработку следующего содержания:

```
Команда234
Команда199
Команда250
Команда270
```

Вам нужно всего лишь найти команду вывода на экран и поменять ее на команду вывода на принтер. Но вы не являетесь гуру в языке программирования, в котором была написана данная программа, и соответственно не знаете все его команды наизусть. Вам придется потратить достаточное количество времени, чтобы разобраться для начала в назначении каждой команды, и только потом вы сможете приступить к исправлению кода. Теперь тот же самый код, но с комментариями:

```
Комментарий: Вывод заставки программы
Команда234
Комментарий: Запрос пароля у пользователя
Команда199
Комментарий: Проверка пароля, если пароль верен, то разрешаем вход
в систему пользователю
Команда250
Комментарий: Вывод на экран прогноза погоды на текущий день
Команда270
```

Думаю, второй пример более наглядный — в нем разобраться намного проще, т. к. это сможет сделать достаточно быстро даже неопытный программист. К тому же, комментируя написанный код, вы следите за логикой программы, таким образом, вы избавляете себя от лишних ошибок и сокращаете время на разработку.

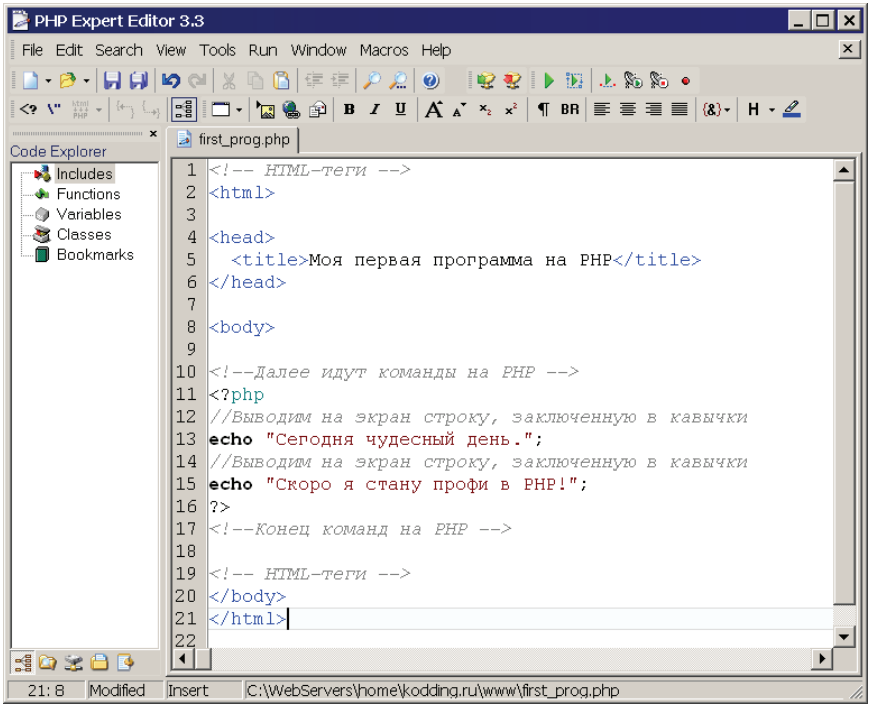


Рис. 5.1. Программа first_prog.php, созданная в редакторе PHP Expert Editor

В строках со **2** по **8**, как и говорится в комментариях, идут стандартные HTML-теги. В строках **11** и **16** используются уже знакомые вам:

- открывающий тег `<?php`, указывающий, что далее будет участок кода на PHP;
- закрывающий тег `?>`, который означает конец участка кода на PHP.

В строке **12** мы опять встречаем комментарий, правда он уже является частью PHP-кода. Комментарии на PHP можно обозначать несколькими способами (табл. 5.1).

Использование комментариев аналогично хорошим манерам, нужно начинать программирование с них и потом это войдет в привычку.

В строке **13** используется команда `echo`, которая предназначена для вывода данных на экран. В этом случае у нас будет выводиться текст, который заключен в двойные кавычки, а именно:

```
"Сегодня чудесный день."
```

Таблица 5.1. Способы обозначения комментариев в PHP

№ варианта	Способ обозначения	Пример
1	//	`// Это однострочный комментарий`
2	#	`# Это однострочный комментарий`
3	/* */	`/* Это многострочный комментарий */`

Более точно будет сказать, что команда `echo` формирует строку, которая в дальнейшем будет выведена в окне браузера пользователя, т. к. именно оно выступает в роли своеобразного экрана.

В PHP в большинстве случаев двойные кавычки могут быть заменены на одинарные, т. к. являются дополнительным способом обозначения строки или символьного типа данных (о типах данных мы поговорим чуть позже), т. е., например, команда:

```
echo "Сегодня чудесный день.";
```

может быть спокойно заменена на:

```
echo 'Сегодня чудесный день.';
```

Результат выполнения программы от этого не изменится.

Каждая команда в PHP заканчивается точкой с запятой, вы можете увидеть этот знак в конце текста, заключенного в кавычки, который будет выводиться с помощью команды `echo`. То есть к команде относится не только ее название, а также и данные, которые ей нужно обработать, в этом случае в качестве данных выступает текст, заключенный в кавычки.

Смысл точки с запятой можно выразить на примере сочинения на уроке литературы: вы всегда ставите точку в конце предложения, чтобы отделить одну мысль от другой, также и при написании программы на PHP, нужно в конце каждой команды ставить своеобразную точку, т. е. точку с запятой.

Теперь самое время сохранить программу и запустить ее посредством браузера. Результат выполнения можно увидеть на рис. 5.2.

Предлагаю посмотреть на то, что вернул нам сервер. В Internet Explorer для этого достаточно выбрать пункт меню **Вид | Просмотр HTML-кода**, результат представлен на рис. 5.3.

Работа препроцессора PHP обсуждалась в разд. 2.1.

Мы видим только HTML-теги и комментарии, сделанные с помощью HTML. Команды `echo` и комментарии, сделанные в PHP-коде, пропали, т. к. были обработаны препроцессором.

Рассмотрим еще один пример. Создайте файл second_prog.php (листинг 5.2).

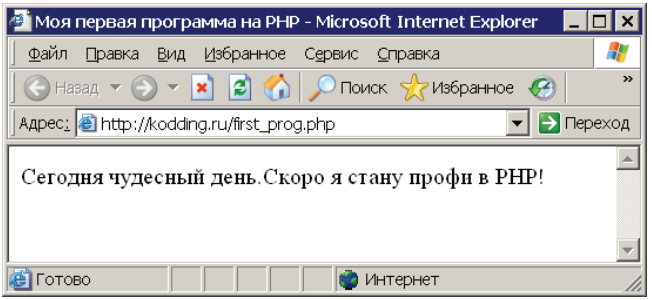


Рис. 5.2. Результат выполнения программы first_prog.php

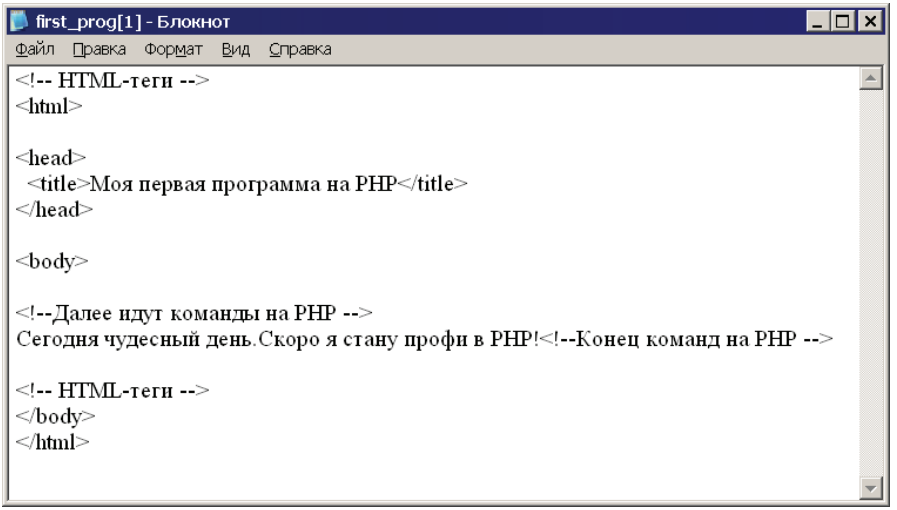


Рис. 5.3. Результат, который сервер возвратил браузеру после выполнения first_prog.php

Листинг 5.2. Файл second_prog.php

```
1  <!-- HTML-теги -->
2  <html>
3
4  <head>
5    <title>Вторая программа на PHP</title>
6  </head>
7
8  <body>
9
```

```
<!--Далее идут команды на PHP -->
<?php
//Выводим на экран строку, заключенную в кавычки
echo "Сегодня чудесный день.<BR>";
//Выводим на экран строку, заключенную в кавычки
echo "<I>Скоро я стану профи в PHP!</I>";
?>
<!--Конец команд на PHP -->

<!-- HTML-теги -->
</body>
</html>
```

Результат выполнения программы можно увидеть на рис. 5.4.

Давайте посмотрим, какие данные сервер вернул браузеру — рис. 5.5.

Рассмотрим принцип работы команды `echo` или, другими словами, каким образом она будет обработана "препроцессором гипертекста":

В файле second_prog.php существует фрагмент кода на PHP:

```
<?php
echo "Сегодня чудесный день.<BR>";
echo "<I>Скоро я стану профи в PHP!</I>";
?>
```

Когда данный фрагмент будет обработан "препроцессором гипертекста", то вместо него получится обычный текст, содержащий HTML-теги:

```
Сегодня чудесный день.<BR><I>Скоро я стану профи в PHP!</I>
```

Браузер отобразит это следующим образом:


```
Сегодня чудесный день.
Скоро я стану профи в PHP!
```

Переходим к следующему примеру, выводящему таблицу, сверху и снизу которой находится текст, сформированный с помощью PHP-кода. Создайте файл third_prog.php (листинг 5.3).

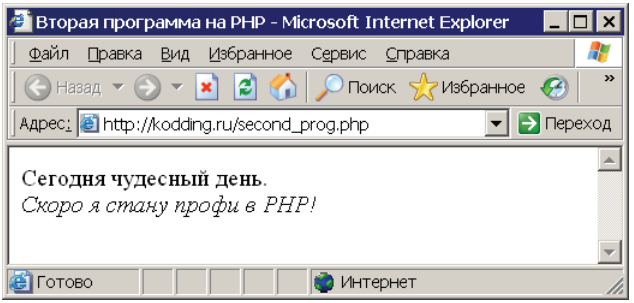


Рис. 5.4. Результат выполнения программы second_prog.php

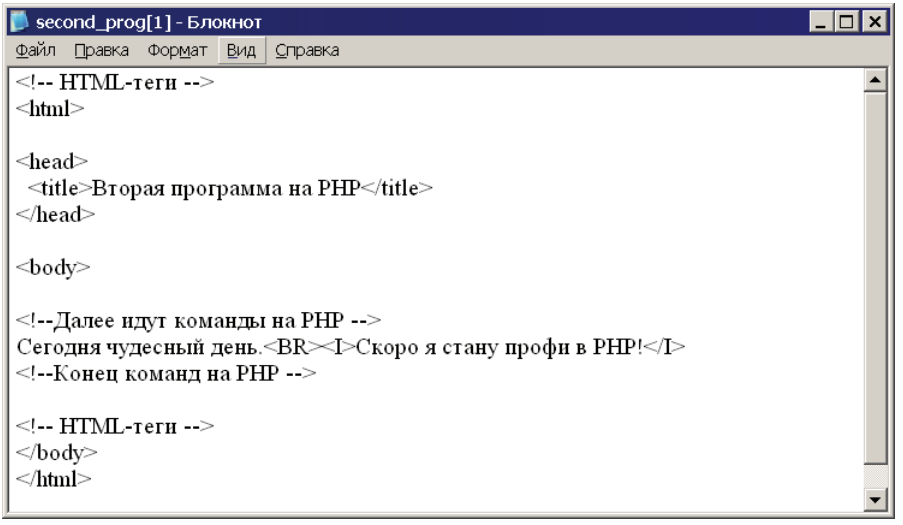


Рис. 5.5. Результат, который сервер возвратил браузеру после выполнения second_prog.php

Листинг 5.3. Файл third_prog.php

```
1  <html>
2  <head>
3    <title>Третья программа на PHP</title>
4  </head>
5  <body>
6
7  <?php
8  //Первое объявление PHP-кода
```

```
9  echo "Верхняя надпись: Таблица состоит из двух ячеек";
10 ?>
11
12 <TABLE BORDER=1>
13 <TR>
14    <TD>Первая ячейка<TD>
15 </TR>
16 <TR>
17    <TD>Вторая ячейка</TD>
18 </TR>
19 </TABLE>
20
21 <?php
22 //Второе объявление PHP-кода
23 echo "Нижняя надпись: По всем вопросам обращаться в кабинет 203";
24 ?>
25
26 </body>
27 </html>
```

Результат выполнения программы изображен на рис. 5.6.

В HTML-документах можно многократно объявлять фрагменты PHP-кода, обозначая их специальными тегами: "<?php" и "?>". В результате чего у нас получится полноценная программа, поэтому имя файла должно иметь соответствующее расширение.

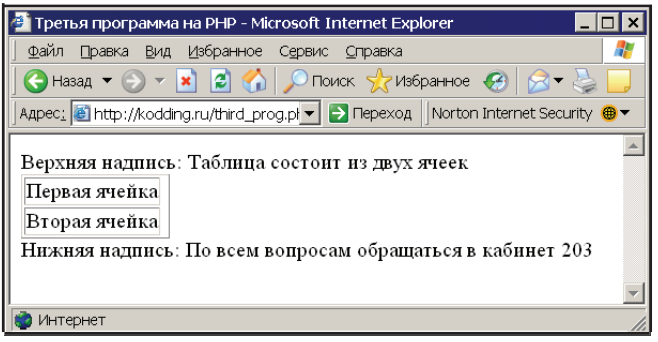


Рис. 5.6. Результат выполнения программы third_prog.php

Следующий пример делает то же самое, что и предыдущий, главное его отличие — он полностью написан на PHP. Создайте файл fourth_prog.php (листинг 5.4).

Листинг 5.4. Файл fourth_prog.php

```
<?php
//Четвертая PHP-программа полностью написана на PHP.
//Вывод HTML-тегов осуществляется с помощью команды echo.
echo "<html>";
echo "<head>";
echo "<title>Четвертая программа на PHP</title>";
echo "</head>";
echo "<body>";

echo "Верхняя надпись: Таблица состоит из двух ячеек";

echo "<TABLE BORDER=1>";
echo "<TR>";
echo "   <TD>Первая ячейка<TD>";
echo "</TR>";
echo "<TR>";
echo "   <TD>Вторая ячейка</TD>";
echo "</TR>";
echo "</TABLE>";

echo "Нижняя надпись: По всем вопросам обращаться в кабинет 203";

echo "</body>";
echo "</html>";
?>
```

Вы можете запустить программу и убедиться в ее работоспособности. Важно отметить, что данный способ написания PHP-скриптов не самый лучший пример для подражания и приведен здесь только в качестве учебного варианта. Главная проблема — это увеличение времени обработки программы сервером, т. к. каждая ее строка будет нуждаться в обработке "препроцессором гипертекста". На локальном компьютере это незаметно, но представьте ситуацию, когда размер вашего скрипта увеличивается в 100 раз, и к нему одновременно обращается большое количество людей, а именно так и будет, если вы участвуете в разработке даже среднего по величине проекта. Вот тут и проявится медлительность вашего творения.

5.2. Переменные

Начнем этот раздел с примера. Создайте файл variable_prog.php (листинг 5.5).

Листинг 5.5. Файл variable_prog.php

```
<html>
<head>
  <title>Изучение переменных</title>
</head>
<body>

<?php
//Объявляем символьную переменную
$a="Мы продолжаем обучаться PHP <BR>";
//Выводим значение переменной $a
echo $a;
//Объявляем числовую переменную
$b=5;
//Выводим значение переменной $b
echo "Это наша ".$b." программа";
?>

</body>
</html>
```

Результат выполнения программы представлен на рис. 5.7.

В данном скрипте используется новый элемент языка — *переменная*. В PHP она всегда обозначается с помощью знака доллара: `$`.

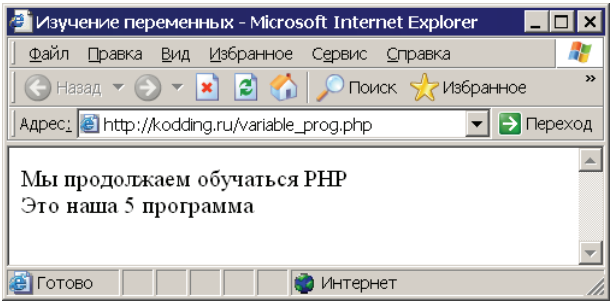


Рис. 5.7. Результат выполнения программы variable_prog.php

Переменная — это определенная область памяти, которая имеет имя и зарезервирована для хранения данных определенного типа. Тип данных определяет, какой вид информации хранит переменная и какие действия можно совершать с этой информацией.

В табл. 5.2 характеризуются простые типы данных, работа с которыми описывается в данном разделе.

Если вы хотите просто вывести символ "$" на экран, то для этого перед ним необходимо поставить "\".

Таблица 5.2. *Простые типы данных*

Названия типа данных	Обозначение в PHP	Комментарии
Строковый (или текстовый)	STRING	Любой печатаемый символ, набор символов или текст
Целочисленный	INTEGER	Любое целое число
Вещественный	FLOAT или DOUBLE	Число с плавающей точкой или дробное число
Логический	BOOLEAN	Может принимать только два значения: TRUE или FALSE или их сокращенный числовой вариант 0 (это FALSE) и 1 (это TRUE)

Прежде чем начать работать с переменной, ее необходимо объявить, что и было сделано в строках **9** и **13** программы variable_prog.php. Если вы заметили, тип переменной назначен не был, т. к. PHP определит его автоматически. Для присвоения значения переменной используется знак равно (=).

Тип данных в PHP — понятие достаточно условное, т. к. он может меняться автоматически в зависимости от того, где мы используем переменную. Вы сможете убедиться в этом немного позже.

После того как переменная больше не нужна, либо выполнение программы, в которой происходила работа с ней, закончено, переменную удаляют. Это можно сделать с помощью специальной команды, но является совсем не обязательным действием, потому что PHP отлично справляется с этим самостоятельно.

На рис. 5.8 изображена схема, характеризующая принцип работы с переменными в PHP на примере программы variable_prog.php.

Вернемся к программе variable_prog.php. В строке **9** мы создаем переменную `$a` и присваиваем ей текст, следовательно, переменная будет строкового типа, т. к. наличие кавычек укажет PHP на это. В строке **11** мы используем команду `echo` и выводим значение переменной `$a` (рис. 5.9).

Переменную можно объявлять практически в любом месте программы.

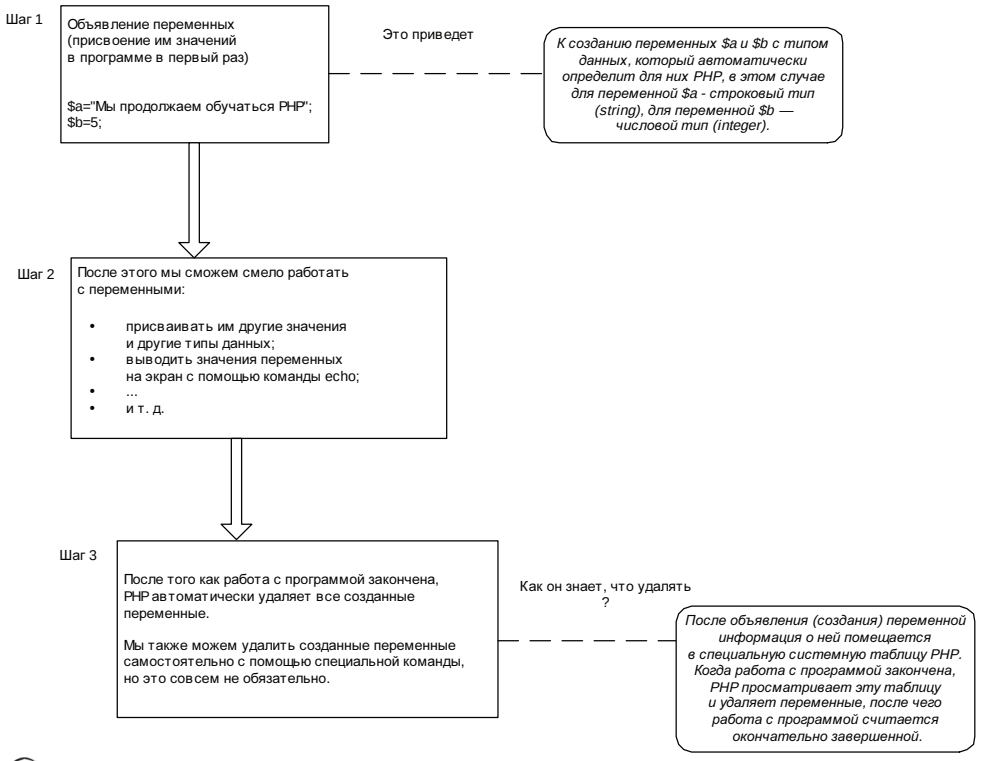


Рис. 5.8. Принцип работы с переменными на примере программы variable_prog.php

Обратите внимание, что после команды присвоения значения переменной (когда это делается первый раз в программе, то это называется объявлением переменной, т. к. это приведет к ее созданию) стоит точка с запятой. Как уже говорилось ранее, это знак отделения одной команды от другой. Присвоение переменной значения тоже является командой.

В строке **13** нашей программы мы создаем переменную `$b`, на этот раз типа `integer`, т. к. справа от знака равно находится целое число. Далее мы выводим значение переменной, но немного другим способом, чем ранее:

```
echo "Это наша ".$b." программа";
```

Команде `echo` передается значение, состоящее из трех частей:

1. `"Это наша "`
2. `$b`
3. `" программа"`

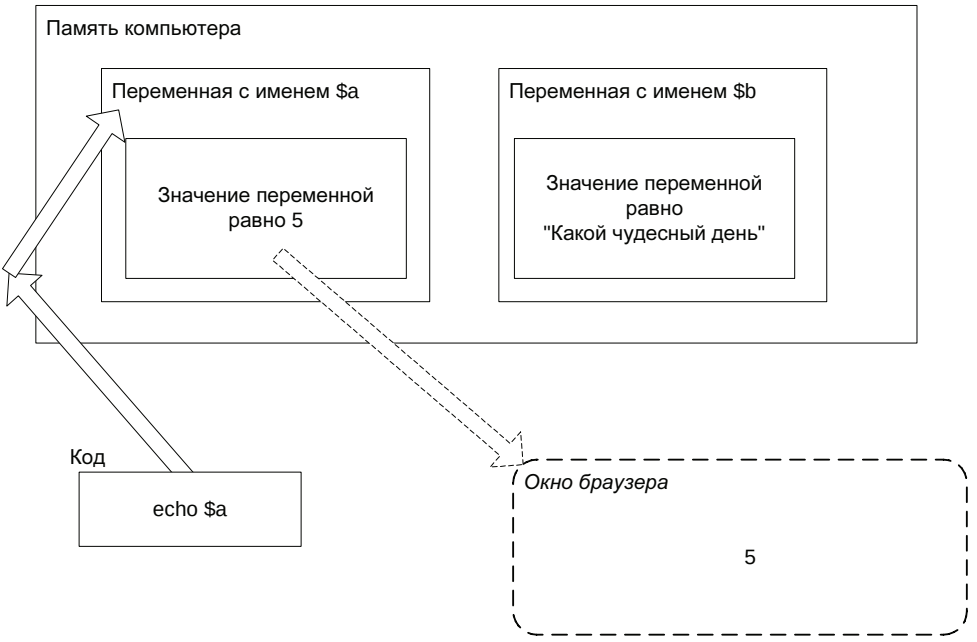


Рис. 5.9. Вывод значения переменной `$a` с помощью команды `echo`

С помощью точки мы объединяем все эти три части в одну строку. То есть можно было бы написать:

```
echo "Это наша ";
echo $b;
echo " программа";
```

Но вариант `echo "Это наша ".$b." программа";` более удобный и легкочитаемый.

Точка — это своеобразный клей, который позволяет объединять данные в одну строку.

5.3. Практикуемся в работе с переменными

Перейдем к более практическим действиям. Предлагаю написать небольшую программу, где создаются переменные всех простых типов, рассмотренных ранее, и далее их значения выводятся на экран. Создайте файл all_variable_prog.php (листинг 5.6).

Листинг 5.6. Файл all_variable_prog.php

```
<html>
<head>
  <title>Шестая программа</title>
</head>
<body>

<?php
//Объявляем логическую переменную
$variable_boolean=TRUE;
//Объявляем целочисленную переменную
$variable_integer=666;
//Объявляем вещественную переменную
$variable_float=50.255;
//Объявляем строковую переменную
$variable_string="Мы продолжаем знакомство с переменными";

//Выводим значения всех переменных
echo "Логическая: ".$variable_boolean."<BR><HR>";
echo "Целочисленная: ".$variable_integer."<BR><HR>";
echo "Вещественная: ".$variable_float."<BR><HR>";
echo "Строковая: ".$variable_string."<BR><HR>";
?>

</body>
</html>
```

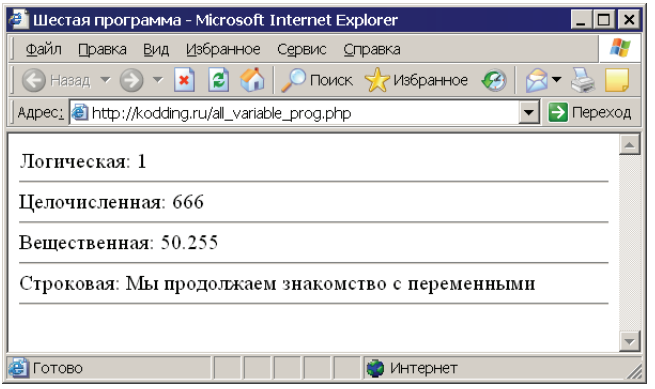


Рис. 5.10. Результат выполнения программы all_variable_prog.php

Результат выполнения программы представлен на рис. 5.10.

PHP является регистрозависимым (от слова регистр) языком. Поэтому переменные `$test_variable` и `$Test_variable` будут абсолютно разными с точки зрения PHP. Рассмотрим это на практике: создайте программу test_variable_prog.php (листинг 5.7, там специально допущена небольшая ошибка).

Листинг 5.7. Файл test_variable_prog.php

```
<html>
<head>
  <title>Тестируем работу переменных</title>
</head>
<body>

<?php
//Объявляем переменную
$test_variable="ТЕСТ";
//Выводим значение переменной на экран, но специально
//первую букву в ее названии меняем на большую
echo $Test_variable;
?>

</body>
</html>
```

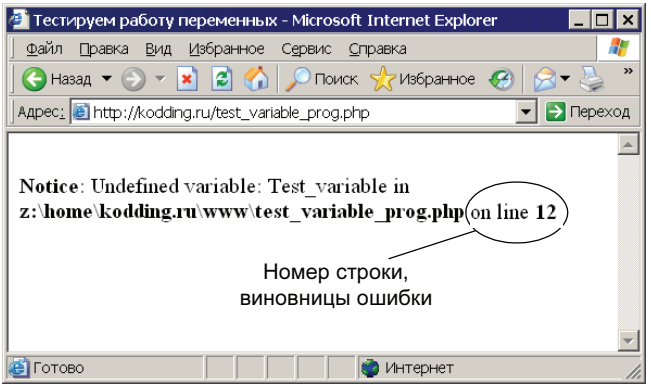


Рис. 5.11. Результат выполнения программы test_variable_prog.php

Как видите, нет особой загадки в том, что PHP может автоматически определять тип объявленной в коде переменной. Строковые переменные он определяет по кавычкам, если кавычек нет и после знака равно используются цифры, то присвоение переменной целого (`integer`) или вещественного (`float`) типа зависит от точки, которая является признаком дробного числа. Логическая переменная определяется по наличию значения `TRUE` или `FALSE`.

*Если в строке **12** имя переменной написать с маленькой буквы, то скрипт будет успешно работать.*

Результат выполнения программы представлен на рис. 5.11.

Ничего удивительного, что в браузере выводится сообщение об ошибке. Текст `Undefined variable` говорит об использовании в программе не объявленной (или не существующей) переменной. `Test_variable` указывает, что этой переменной является `$Test_variable`. Ниже можно увидеть номер строки — виновницы ошибки.

Когда вы даете имя переменной, следует руководствоваться следующими двумя правилами, чтобы избежать лишних ошибок:

- имя может содержать латинские буквы, цифры и символ подчеркивания;
- имя не должно начинаться с цифры.

В табл. 5.3 показаны примеры правильных и неправильных имен переменных.

***Таблица 5.3.** Правильные и неправильные имена переменных*

Имя переменной	Комментарии
`$Name_of_person`	Правильное имя переменной, т. к. оно начинается с латинской буквы и не содержит запрещенных символов
`$11p`	Неправильное имя, начинается с цифры
`$_SSS2`	Правильное имя переменной, т. к. начинается с символа подчеркивания
`$автомобиль`	Хотя в официальной документации и сказано, что переменные могут иметь такие имена, но будьте уверенны, что в дальнейшем у вашей программы это вызовет множество проблем. Такое имя использовать крайне не рекомендуется
`$comp#any`	Неправильное имя, потому что оно содержит запрещенный символ #

В заключение данного раздела рассмотрим команду удаления переменной. Как уже обсуждалось ранее, PHP автоматически удаляет все переменные после того, как работа программы закончена. Очень редко встречаются ситуации, в которых это приходится делать самостоятельно, но все-таки данная необходимость может возникнуть. Создайте программу del_variable_prog.php (листинг 5.8).

Листинг 5.8. Файл del_variable_prog.php

```
1  <?php
2  //Объявляем переменную $в
3  $в="Осваиваем команду удаления переменной";
4  //Выводим значение $в на экран
5  echo $в;
6  //Удаляем переменную $в
7  unset($в);
8  //Следующая команда вызовет ошибку,
9  //т.к. переменная была удалена
10 echo $в;
11 ?>
```

Результат выполнения программы, он опять будет с ошибкой, представлен на рис. 5.12.

В строке **5** осуществляется вывод значения переменной `$в`. В строке **10** попытка повторить то же самое действие вызвала ошибку, т. к. переменная `$в` уже была удалена командой `unset`, специально предназначенной для этого.

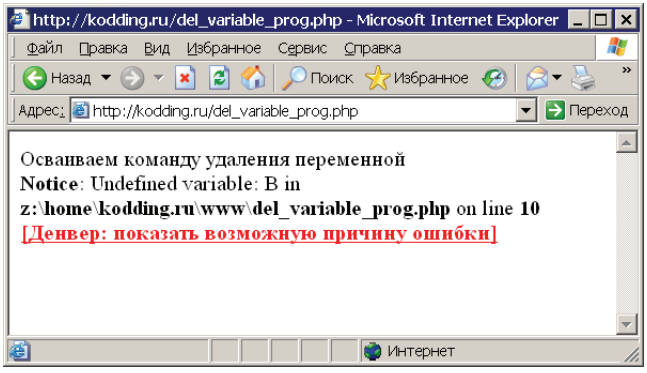


Рис. 5.12. Результат выполнения программы del_variable_prog.php

В данном листинге не используются HTML-теги, описывающие документ: его начало, тело и окончание. Такой стиль программирования не является ошибочным и был выбран, чтобы сделать акцент на более значимой информации. Стоит отметить, что в реальных проектах рекомендуется не отказываться от HTML-тегов.

5.4. Функции

Функция — это некоторый участок кода с определенным именем, к которому можно многократно обращаться в программе. Функция обладает одной отличительной чертой: она всегда возвращает какой-то результат в то место программы, откуда произошел ее вызов. На рис. 5.13 изображена схема, иллюстрирующая этот процесс.

Скобки после имени функции являются обязательным элементом, почему так происходит, будет рассмотрено чуть позже.

Функция может располагаться в файле, в котором происходит ее вызов, а может находиться совершенно в другой программе. Предположим, у нас есть функция с именем `GET_KURS_DOLLAR`, возвращающая текущий курс доллара в виде числа. Для того чтобы ею воспользоваться, достаточно сделать следующую запись: `GET_KURS_DOLLAR()`.

Во время выполнения программы данное обращение к функции заменится на возвращенный ею результат, т. е. в этом месте программы будет обыкновенное число. Поэтому в тех случаях, когда результат, который вернула функция, нужно использовать в программе, применяют следующую конструкцию:

```
Переменная=Имя функции();
```

применительно к описанному ранее примеру получится следующая строка:

```
$KURS=GET_DOLLAR();
```

То есть результат выполнения функции сохраняют в переменную, что дает возможность многократно использовать это значение на протяжении всего выполнения скрипта.

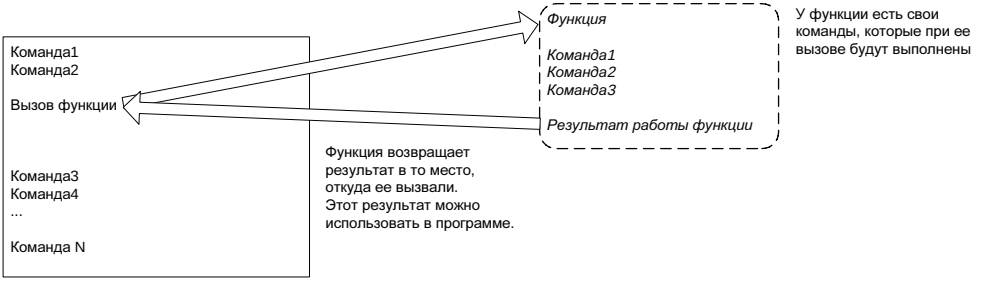


Рис. 5.13.
Использование функции в программе

Еще к одной особенности функции относится то, что ей можно передавать данные, в этом случае они перечисляются через запятую в скобках и называются *входными данными*. Если функция вызывается без параметров, то используются пустые скобки `()`, но это обязательный элемент, не указав который мы столкнемся с ошибкой во время выполнения программы.

В PHP большое количество стандартных функций, созданных разработчиками PHP. Использовать их в своих программах очень просто, для этого достаточно указать имя функции и при необходимости передать ей входные данные.

Вернемся к практике. Давайте рассмотрим стандартную PHP функцию `gettype()`, которая возвращает тип данных для переменной, имя которой передано функции в качестве параметра. Создайте программу use_gettype.php (листинг 5.9).

В других языках программирования для использования стандартных функций необходимо подключать специальные модули, в которых описаны эти функции, в противном случае вы не сможете их применить. В PHP этого делать не надо, что является еще одним примером его простоты.

Листинг 5.9. Файл use_gettype.php

```
<html>
<head>
  <title>Использование функции gettype</title>
</head>
<body>

<?php
//Объявляем логическую переменную
$variable_boolean=TRUE;
//Объявляем целочисленную переменную
$variable_integer=666;
//Объявляем вещественную переменную
$variable_double=50.255;
//Объявляем строковую переменную
$variable_string="Знакомимся с функцией gettype";

//Выводим все значения переменных
```

```
18 echo "<I>Значения переменных:</I><BR>";
19 echo "1) Логическая: ".$variable_boolean."<BR>";
20 echo "2) Целочисленная: ".$variable_integer."<BR>";
21 echo "3) Вещественная: ".$variable_double."<BR>";
22 echo "4) Строковая: ".$variable_string."<BR>";
23
24 //Добавляем две пустые строки
25 echo "<BR>";
26 echo "<BR>";
27
28 //Выводим тип данных для переменных
29 echo "<I>Выводим типы данных тех же самых переменных:</I><BR>";
30 echo "1)".gettype($variable_boolean)."<BR>";
31 echo "2)".gettype($variable_integer)."<BR>";
32 echo "3)".gettype($variable_double)."<BR>";
33 echo "4)".gettype($variable_string)."<BR>";
34 ?>
35
36 </body>
37 </html>
```

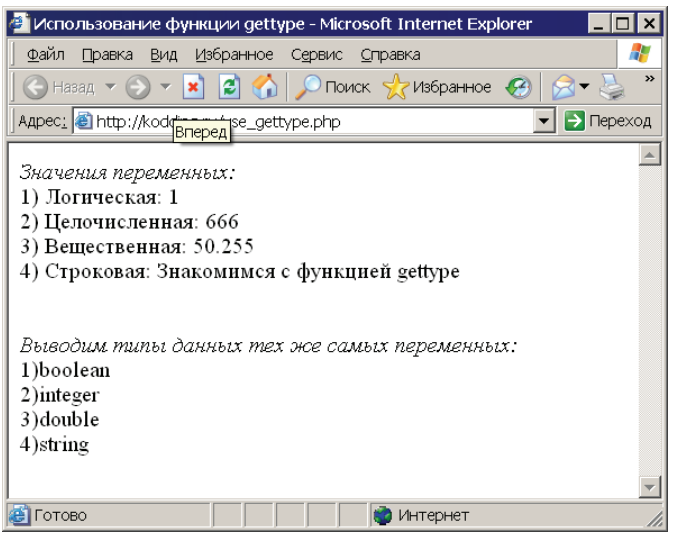


Рис. 5.14. Результат выполнения программы use_gettype.php

Результат выполнения программы представлен на рис. 5.14.

Сначала мы объявляем 4 переменные различных типов данных: `boolean`, `integer`, `double` и `string`. Затем мы выводим значения всех переменных на экран, а далее выводим тип данных каждой переменной, используя функцию `gettype()` вместе с командой `echo`. Конечно, можно было использовать следующую запись:

```
$f=gettype($variable_boolean);
echo "1)".$f."<BR>";
$f=gettype($variable_integer);
echo "2)".$f."<BR>";
$f=gettype($variable_double);
echo "3)".$f."<BR>";
$f=gettype($variable_string);
echo "4)".$f."<BR>";
```

но в этом случае получились бы четыре лишних строчки кода.

Очень интересный момент с переменной `$variable_boolean` логического типа. Дело в том, что для PHP значение `TRUE` соответствует единице, а `FALSE` — нулю. Почему так происходит, будет рассмотрено в следующем разделе, посвященном теме констант.

Рассмотренная ранее команда `unset` на самом деле является функцией, которой в качестве параметра передается имя переменной для удаления. Функция `unset()` специально была названа командой, для того чтобы не путать читателя, т. к. тема функций тогда еще не была рассмотрена. В качестве результата функция `unset()` возвращает `TRUE`, если переменная была успешно удалена, или `FALSE`, если этого сделать не удалось, например, в случае, когда переменная не существует или уже была удалена.

5.5. Константы

Константа — это частный случай использования переменной. Она задается один раз и характеризует значение (или факт), которое будет неизменным на протяжении всей программы. Примеры констант, встречающихся в повседневной жизни:

- в году 4 сезона;
- новый год начинается 1 января;
- в сутках 24 часа;
- число PI.

Для создания константы предназначена функция `define()`, ее синтаксис:

При создании константы, так же как и при ее использовании, знак `$` не используется.

```
define("имя константы", значение константы);
```

Значение константы может быть только простым типом данных (см. табл. 5.2). Рассмотрим пример, в котором задана константа HOURS_IN_DAY, характеризую-

Листинг 5.10. Файл use_constant.php

```
1  <?php
2  //Объявляем константу HOURS_IN_DAY со значением 24
3  define("HOURS_IN_DAY",24);
4  //Объявляем переменную $h,
5  //в которой будет храниться количество часов
6  //для преобразования в дни
7  $h=240;
8
9  //Переводим часы в дни и сохраняем результат
10 //в переменной $our_result
11 $our_result=$h/HOURS_IN_DAY;
12 //Выводим получившийся результат
13 echo "Количество дней в ".$h." часах = ".$our_result;
14 ?>
```

Результат выполнения программы представлен на рис. 5.15.

В строке **3** мы задаем константу HOURS_IN_DAY, в строке **7** объявляем переменную $h, которая будет хранить в виде числа количество часов для перевода их

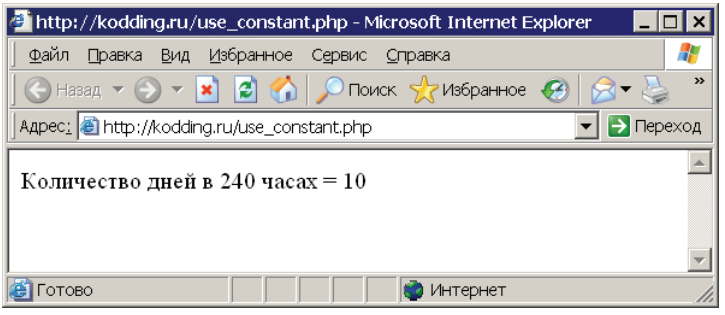


Рис. 5.15. Результат выполнения программы use_constant.php

в дни. В реальной программе данной переменной может которое указал пользователь.

В строке **11** осуществляется деление заданного количества часов (переменная `$h`) на количество часов в дне (константа `HOURS_IN_DAY`), результат сохраняем в переменной `$our_result` и затем в строке **13** выводим его на экран.

В PHP есть такое понятие, как *предопределенные константы*. Это константы, созданные разработчиками PHP. Их ярким примером являются константы `TRUE` и `FALSE`, которые хранят значения 1 и 0 соответственно. Помните, в программе use_gettype.php переменной `$boolean_variable` было присвоено значение `TRUE`, а потом выведено на экран, в результате чего вы увидели 1? Теперь вы знаете, почему так произошло.

Создайте файл another_constant.php следующего содержания:

```
<?php
echo PHP_VERSION;
?>
```

В результате выполнения программы вы сможете увидеть номер версии PHP, используемой вами, т. к. именно это значение хранит предопределенная константа `PHP_VERSION`.

Конечно, вместо константы можно использовать и обычную переменную, но данные, которые не изменят свое значение на протяжении всей программы, принято обозначать как константы, тем самым вы делаете свою программу более логичной, а следовательно, для вас и для других людей, которые, возможно, будут использовать ваш код в дальнейшем, процесс доработки программы или внесения в нее изменений будет более простым.

*Стоит отметить, что с числами в PHP можно выполнять все элементарные операции, которые используются в реальной жизни: сложение (знак +), вычитание (знак –), умножение (знак *) и деление (знак /).*

5.6. Массивы

Предположим, вы решили написать программу для своей коллекции музыкальных дисков. Так как с базами данных вы еще работать не умеете, в данной ситуации лучше всего использовать массивы, тогда информацию о дисках можно будет хранить следующим образом:

```
$disk[1]= "Золотые хиты";
$disk[2]= "Инструментальная музыка";
$disk[3]= "Классическая музыка";
...
```

... а может тебе еще ключ от квартиры, где деньги лежат?

Таким образом, для работы с данными достаточно использовать запись вида `$disk[номер]`, где `номер` — это номер элемента, с которым вы хотите работать.

Массив — набор значений, которые хранятся в определенной последовательности, доступ к ним осуществляется посредством специального ключа, который указывает, к какому именно значению мы хотим обратиться в данный момент (какой именно элемент массива нам нужен). Именно ключ позволяет однозначно отделять данные друг от друга, поэтому каждый ключ массива должен быть уникальным, т. е. двух ключей с одинаковыми именами быть не должно. *Ключ* может быть либо текстом (тип данных `string`), либо целым числом (тип данных `integer`). Значения, которые хранит массив (или каждый элемент массива), совершенно не зависят друг от друга и могут иметь свой тип данных.

Создать (или объявить) массив можно несколькими способами.

Если сильно упростить определение массива, то можно сказать, что это своего рода переменная, которая позволяет хранить множество других переменных.

- Создание пустого массива:

```
$mas = array();
```

В данном случае используется специальное слово `array` (в переводе с англ. "массив") с пустыми скобками. Что приведет к созданию массива, который не будет содержать абсолютно никаких данных.

- Последовательное перечисление:

```
$mas = array("Room1" => "Столовая",
"Room2" => "Зал",
  55=>"Специальное число", 13=>100);
```

В данном варианте все данные перечисляются следующим образом: `Ключ=>значение`. В роли ключа может выступать как строка (например, `Room1`), в этом случае она заключается в кавычки, так и число (например, 55).

- Отдельное объявление каждого элемента:

```
$mas["Room1"]="Столовая";
$mas["Room2"] = "Зал";
$mas[55]="Специальное число"
$mas[13]=100;
```

То есть сначала указывается имя массива, затем в квадратных скобках его ключ, далее знак равно и значение этого элемента массива.

- Автонумерация — суть данного способа заключается в том, что указываются только значения элементов массива без ключей. Это можно сделать либо последовательным перечислением:

```
$mas = array("Столовая", "Зал", "Специальное число", 100);
```

либо отдельным объявлением каждого элемента:

```
$mas[]="Столовая";
$mas[] = "Зал";
$mas[]="Специальное число";
$mas[]=100;
```

В обоих случаях будет создан массив, ключ первого элемента которого равен 0, а все последующие больше на единицу предыдущего, т. е. ключ элемента `"Столовая"` будет равен 0, ключ элемента `"Зал"` будет равен 1, ключ элемента `"Специальное число"` будет равен 2, и ключ элемента `"100"` будет равен 3.

Для того чтобы обратиться к элементу массива, нужно использовать следующую запись:

```
Имя массива[ключ]
```

Рассмотрим пример, в котором создается массив, а затем все его элементы выводятся на экран. Сохраните листинг 5.11 под именем use_array.php.

Листинг 5.11. Файл use_array.php

```
<?php
//Объявляем массив способом последовательного перечисления
$mas = array("Room1"=>"Столовая", "Room2"=>"Зал",
             55=>"Специальное число", 13=>100);

//Чтобы вывести знак $, используется запись \$
echo "<B>Элементы массива \$mas:</B><BR>";

//Выводим элемент с ключом "Room1"
echo $mas["Room1"]."<BR>";
//Выводим элемент с ключом "Room2"
echo $mas["Room2"]."<BR>";
//Выводим элемент с ключом 55
echo $mas[55]."<BR>";
//Выводим элемент с ключом 13
echo $mas[13]."<BR>";
?>
```

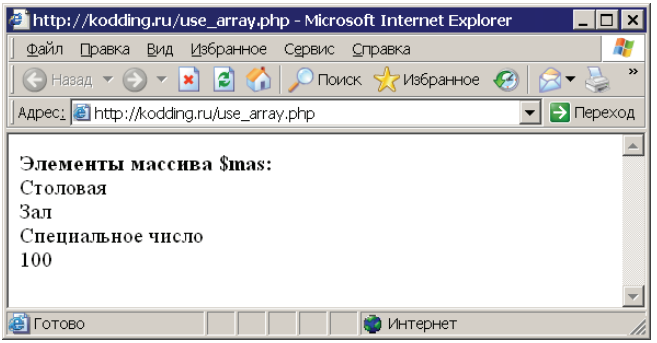


Рис. 5.16. Результат выполнения программы use_array.php

Результат выполнения программы представлен на рис. 5.16.

Предыдущий пример неудобен тем, что для вывода каждого элемента приходится использовать отдельную строку кода, а что делать, если массив будет содержать 50 элементов? Для таких ситуаций в PHP предусмотрена функция `print_r()`. В качестве параметра ей всего лишь нужно передать имя массива. Модифицируйте пример use_array.php следующим образом — листинг 5.12.

Листинг 5.12. Модифицированный use_array.php

```
<?php
//Объявляем массив способом последовательного перечисления
$mas = array("Room1"=>"Столовая", "Room2"=>"Зал",
             55=>"Специальное число", 13=>100);

//Выводим элементы массива вместе с их ключами одним махом
print_r($mas);
?>
```

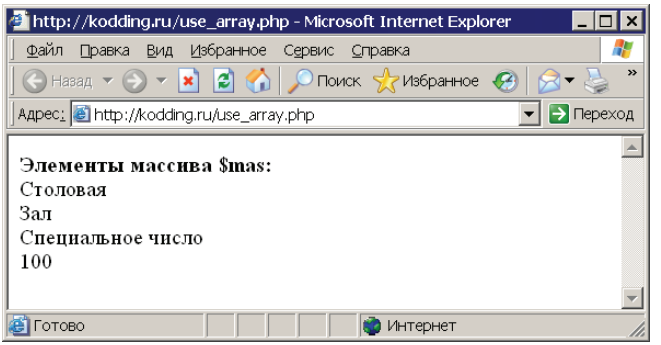


Рис. 5.17. Результат выполнения модифицированной программы use_array.php

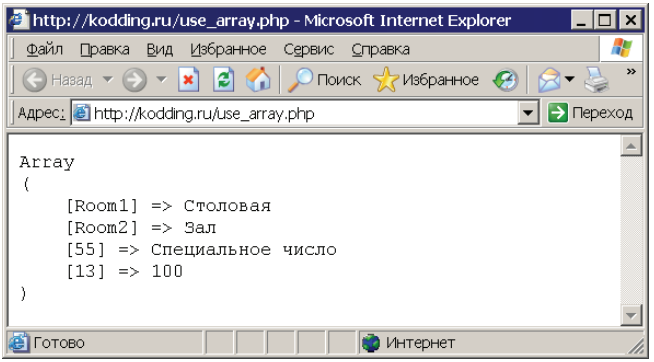


Рис. 5.18. Более изящная форма вывода содержимого массива

Результат выполнения модифицированной программ лен на рис. 5.17.

Важно отметить, что ключи массива, так же как и его имя, чувствительны к регистру, поэтому `$mas["k"]` и `$mas["K"]` будут с точки зрения PHP двумя абсолютно разными элементами.

Замечание

Если функцию `print_r()` поместить между HTML-тегами `<pre>` и `</pre>`, то содержимое массива будет выведено в более изящной форме (рис. 5.18).

5.7. Переменные окружения

Как вы знаете, к одному скрипту могут обращаться одновременно множество пользователей. Для каждого пользователя на сервере создается своеобразный сеанс выполнения скрипта. То есть в результате каждый пользователь работает в своем сеансе, не затрагивая других. Когда вы запрашиваете через браузер какую-то страничку, то это не вся информация, которая отправляется на сервер, кроме этого, туда уходит много дополнительных данных, как системных, необходимых для взаимодействия браузера с сервером, так и пользовательских. Эти данные называются *переменными окружения* и хранятся в специальном массиве с именем `$_SERVER`. На рис. 5.19 изоб-

Листинг 5.13. Файл var_of_enviroment.php

окружения на экран (листинг 5.13).

```
<pre>
<?php
//Выводим все переменные окружения на экран
print_r($_SERVER);
?>
</pre>
```

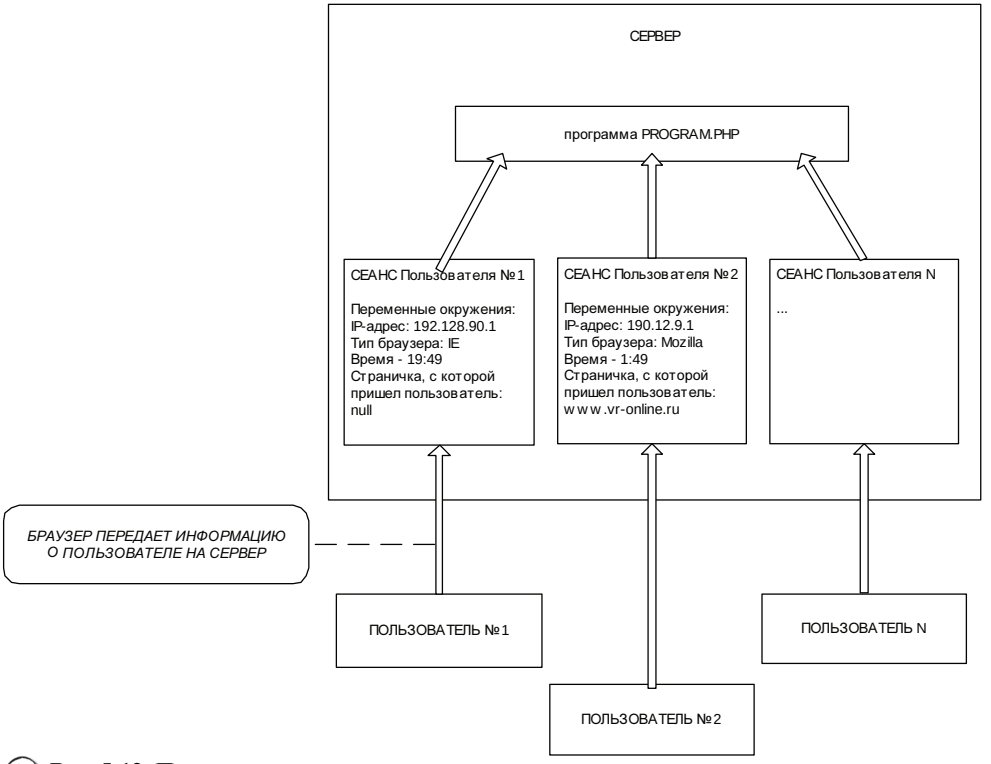


Рис. 5.19. Передача дополнительной информации на сервер

Результат выполнения программы представлен на рис. 5.20.

Разработаем еще одну программу (листинг 5.14), которая будет выводить приветствие пользователю, а также:

- его IP-адрес, информация о котором хранится в `$_SERVER['REMOTE_ADDR']`;
- используемый им тип браузера, информация о котором хранится в `$_SERVER['HTTP_USER_AGENT']`;

Листинг 5.14. Файл info_user.php

```
<?php
echo "Ваш IP-адрес: ".$_SERVER['REMOTE_ADDR']."<BR>";
echo "Ваш браузер: ".$_SERVER['HTTP_USER_AGENT']."<BR>";
echo "Адрес программы, которую вы запросили: ";
echo $_SERVER['SCRIPT_FILENAME']."<BR>";
?>
```

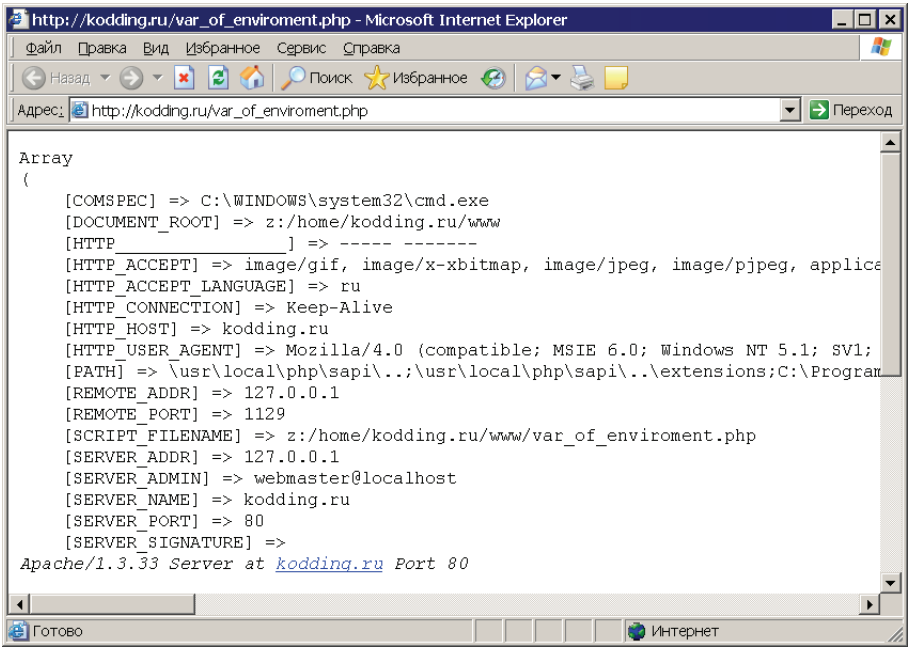


Рис. 5.20. Результат выполнения программы var_of_enviroment.php

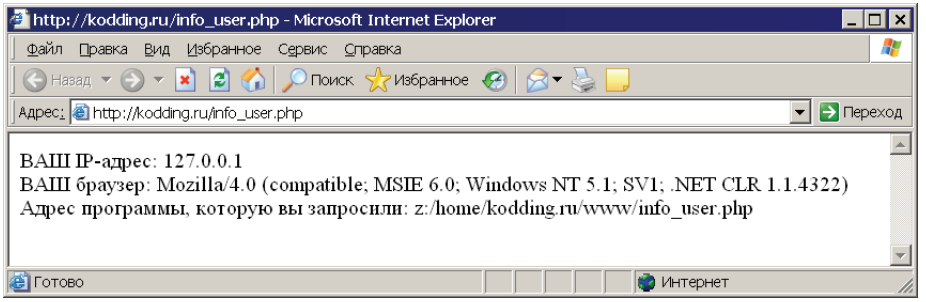


Рис. 5.21. Результат выполнения программы inf_user.php

Результат выполнения программы представлен на рис. 5.21.

Стоит отметить, что переменные окружения очень удобно использовать для реализации лог-журнала, в который заносится информация о каждом посетителе вашего сайта. Таким образом вы можете узнавать их общее количество и отслеживать, растет ваша аудитория или нет.

Глава 6
Счетчик посещений

У большинства сайтов, размещенных в Интернете, вы можете видеть счетчики, которые отображают, какое число людей посетили данную страничку. В этой главе мы рассмотрим реализацию собственного счетчика посещений сайта. Также поговорим об альтернативных счетчиках, которые позволяют еще и просматривать статистику посещения вашего сайта от таких гигантов Web-мира, как MAIL и RAMBLER. Коснемся темы "информеров" и их использования.

6.1. Разработка программы

Предположим, вы создали HTML-страничку следующего вида:

Листинг 6.1. HTML-страничка

```
<html>

<head>
  <title>Домашняя страничка Иванова Пети</title>
</head>
<body>

<H1>Домашняя страничка Иванова Пети</H1>
<BR>
<BR>
Привет!
Меня зовут Петя.
Я изучаю PHP.
</body>
</html>
```

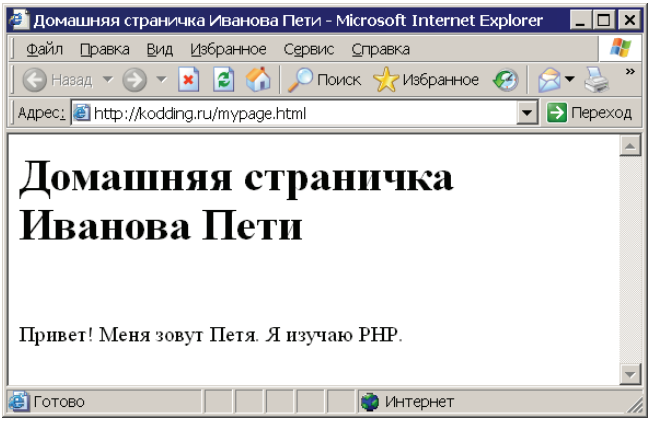


Рис. 6.1. Простая HTML-страница

Сохраните ее под именем mypage.html и запустите посредством браузера (рис. 6.1).

Необходимо, чтобы на этой страничке размещался счетчик, который показывал бы количество посещений данной страницы. Для этого разработаем отдельную программу с именем counter.php (листинг 6.2).

Листинг 6.2. Файл counter.php

```
<?php
//Открываем файл
$f=fopen("counter.txt","a+t") or die("Не могу открыть файл");

//Блокируем файл, чтобы никто не мог к нему обратиться
//пока мы с ним не закончим работу
flock($f,2);

//Читаем в переменную $s значение счетчика
$s=fgets($f);
//увеличиваем значение $s на 1
$s=$s+1;
//Удаляем все содержимое файла counter.txt
ftruncate($f,0);
//записываем новое число
fputs($f,$s);

//снимаем блокировку
```

```
19 flock($f,3);
20 //закрываем файл, теперь с ним могут работать другие
21 fclose($f);
22 //выводим показания счетчика на экран
23 echo $s;
24 ?>
```

Логика работы данной программы построена следующим образом: число посещений хранится в обычном текстовом файле с именем counter.txt, при каждом новом обращении к описанной ранее программе это число увеличивается на единицу.

В PHP с файлом можно работать в двух режимах:

- текстовом — означает, что вы работаете со всей информацией, содержащейся в файле, как с текстом, состоящим из строк. Данный вариант можно сравнить, как будто работа ведется в обычном блокноте. То есть с точки зрения программы в текстовом режиме файл counter.txt будет иметь следующее представление — рис. 6.2;
- бинарном — подразумевает, что файл состоит из байтов данных. Этот режим в основном предназначен для работы со всеми типами файлов отличных от текстового, например, графических.

Прежде чем работать с файлом, его нужно открыть, что мы и делаем в строке **3**, с помощью функции `fopen()`, она имеет следующий синтаксис:

```
fopen(имя файла, режим работы);
```

Данная функция возвращает специальное число, которое называется *дескриптор открытого файла*, его нужно запомнить (сохранить в переменной), чтобы в дальнейшем можно было обращаться к файлу, потому что вся работа с ним осуществляется через этот дескриптор. Именно это мы и делаем, присваивая результат работы `fopen()` обычной переменной `$f`. То есть теперь мы можем работать с файлом через обычную переменную, и нам не нужно постоянно указывать ни имя файла, ни режим работы с ним (к режимам работы мы еще вернемся чуть позже).

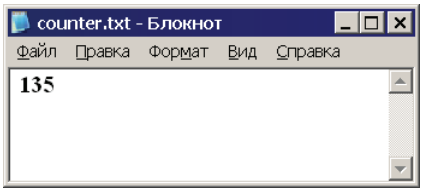


Рис. 6.2. Работа в текстовом режиме похожа на работу с файлом в Блокноте

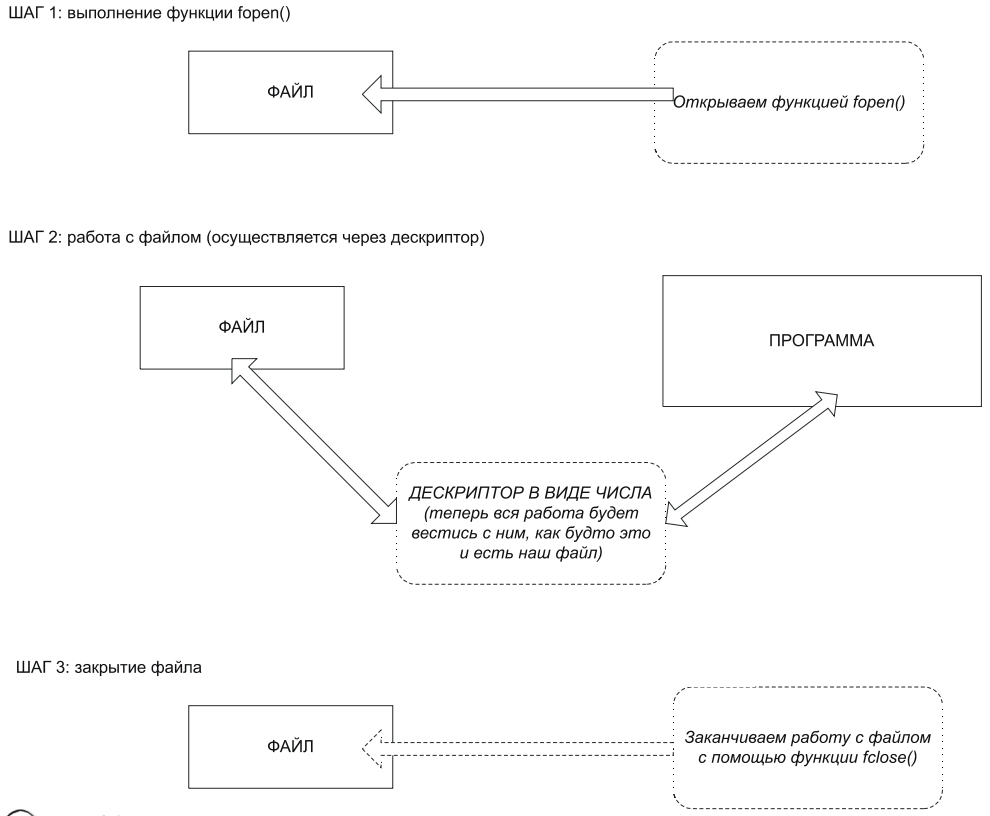


Рис. 6.3. Упрощенная схема работы с файлом в PHP

На рис. 6.3 изображена упрощенная схема работы с файлом в PHP.

Как уже было сказано ранее, одним из параметров функции `fopen()` является *режим работы*. На самом деле этот параметр имеет две составляющие, одну мы уже рассмотрели (вид — текстовый и бинарный), а другую, назовем ее условно *способ работы с файлом*, опишем сейчас — табл. 6.1.

Замечание

Дескриптор файла относится к типу данных `resource`, поэтому переменная `$f` будет тоже этого типа. Тип данных `resource` предназначен для хранения ссылки на внешний ресурс. В данном случае в роли ссылки выступает специальное число, которое указывает на открытый нами файл с определенным режимом работы.

Таблица 6.1. Способы работы с файлом

Способ работы с файлом	Описание
r	Файл открывается для чтения, указатель текущей позиции помещается в начало файла. Если файла не существует, то возникнет ошибка
r+	Файл открывается для чтения и записи, указатель текущей позиции помещается в начало файла. Если файл не существует, то возникнет ошибка
w	Создается пустой файл и открывается только для записи. Если файл с таким именем существует, то он перезаписывается
w+	Создается пустой файл и открывается для записи и для чтения. Если файл с таким именем существует, то он перезаписывается
a	Файл открывается для записи, указатель текущей позиции помещается в конец файла. Если файла не существует, то он создается
a+	Файл открывается для записи и чтения, указатель текущей позиции помещается в конец файла. Если файла не существует, то он создается

Именно в зависимости от указания способа работы будет зависеть то, что мы сможем делать с открытым файлом. Например, если он открыт для чтения, то при попытке записать в него какую-то информацию, возникнет ошибка. Итак, полная запись второго параметра функции `fopen()` выглядит следующим образом:

```
Способ_работыРежим_работы
```

где `Способ_работы` — это одно из значений, представленных в табл. 6.1. `Режим_работы` — это один из двух приведенных далее вариантов.

В табл. 6.1 используется словосочетание "указатель текущей позиции", это понятие будет рассмотрено позже.

- `t` — текстовый.

  Например, как в программе counter.php: `a+t`.

  `a+` будет означать, что файл открывается для записи и чтения, указатель помещается в конец файла, если файла не существует, то он создается. А `t` будет означать, что работа с открытым файлом предполагается как с набором строк, т. е. как с обычным текстом.
- `b` — бинарный.

  Например: `a+b`.

  Здесь опять используем `a+`, но теперь работа с открытым файлом предполагается как с набором байтов.

Весь процесс работы с файлом counter.txt, после его открытия, упрощенно можно описать следующим образом:

1. Чтение значения счетчика в переменную.
2. Увеличение этого значения на 1.
3. Запись нового значение в файл.

В строке **7** используется функция блокировки файла `flock()`. Благодаря этому, пока один пользователь работает с файлом, никто другой, кроме него, не сможет вносить изменения в данный файл. На рис. 6.4 изображена схема, поясняющая этот процесс.

Рассмотрим случай, когда блокировка файла не используется, т. е. когда существует вероятность, что два пользователя считывают значение файла counter.txt примерно в одинаковый промежуток времени. После этого первый пользователь записывает новое значение, увеличенное на 1, второй это делает чуть позже, например, из-за небольших задержек в сети. В результате значение счетчика увеличивается не на 2, как должно быть, а всего лишь на 1, что неправильно. Благодаря блокировке, после того как кто-то начал работу с файлом, никто другой не сможет работать с ним, пока блокировка не будет снята. Другие пользователи в этом случае будут выстроены в некоторую очередь, т. е. они смогут обратиться к файлу только после того, как он будет разблокирован.

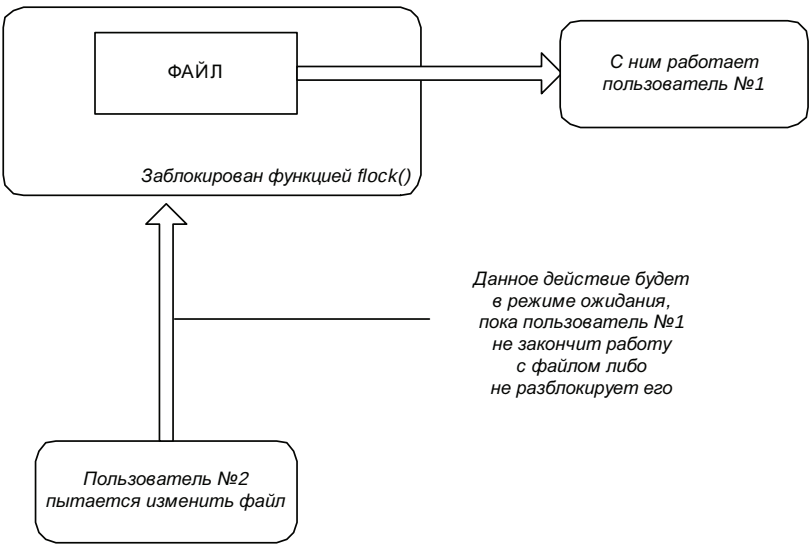


Рис. 6.4. Принцип работы блокировки файла

Далее представлен синтаксис функции `flock()`:

```
flock(дескриптор файла, числовое
значение);
```

где `дескриптор файла` — это имя нашей переменной `$f`, которой мы присвоили значение дескриптора, возвращенного функцией `fopen()`, а числовое значение предназначено для указания варианта блокировки. Возможны следующие варианты (причем можно указывать как числовое значение, так и его альтернативное значение в виде константы):

- `2` или `LOCK_EX` — исключительный вариант блокировки, потому что пока один пользователь работает с файлом, никто другой не может к нему обратиться (используется в программе counter.php);
- `3` или `LOCK_UN` — снятие установленной ранее блокировки.

Замечание

Есть еще вариант мягкой (разделяемой) блокировки. Он устанавливается указанием 1 или константы `LOCK_SH`. В этом случае работа с файлом более лояльна — нет жесткого ограничения на то, что с файлом может работать только один пользователь. Не рекомендуется использовать данный вариант, т. к. нельзя дать 100% гарантии, что при работе с файлом не произойдет сбой, к тому же этот вариант не работает в операционной системе Windows.

Теперь еще немного углубимся в процесс работы с файлом и рассмотрим такое понятие, как *указатель текущей позиции*. Его назначение аналогично курсору при работе в текстовом редакторе — обозначение текущей позиции в открытом файле. Если необходимо что-то записать, то запись данных начнется именно с места, где установлен указатель. Так же, если необходимо прочитать данные из файла, процесс чтения будет осуществлен, начиная с текущего положения указателя. Таким образом, можно сказать, что вся работа с файлом строится в соответствии с позицией указателя.

Замечание

В табл. 6.1 вы можете найти описание того, где будет находиться указатель после открытия файла при различных режимах работы с ним.

Изначально, после открытия файла в режиме `a+`, указатель будет установлен в нулевую позицию, т. е. в начало файла.

В строке **10** мы получаем данные из файла с помощью функции `fgets()`, она имеет следующий синтаксис:

```
fgets(файловая переменная);
```

Данная функция читает очередную строку из файла. Единственный параметр, который необходимо передать в функцию `fgets()`, — это дескриптор файла. В его качестве мы опять указываем переменную `$f`, в которой этот дескриптор был сохранен ранее.

В результате выполнения в программе строки **10** функция `fgets()` возвратит первую и единственную строку файла counter.txt, где как раз и содержится значение счетчика. Эту строку мы сохраняем в переменной `$s`.

Переходим к строке **12**, где происходит увеличение значения переменной `$s` (т. е. увеличение значения счетчика) на единицу. Для этого действия используется следующая конструкция:

```
$s=$s+1;
```

Стоит отметить, что ее можно заменить на альтернативный более короткий вариант:

```
$s+=$s;
```

который расшифровывается как "увеличить (знак плюс) на единицу значение переменной, находящейся с правой стороны от знака равно, и присвоить результат переменной `$s` (находящейся слева от знака операции)".

Теперь необходимо записать новое значение счетчика. Но логично перед этим удалить предыдущее. Как раз для этого используется функция `ftruncate()` в строке **14** программы, ее синтаксис выглядит следующим образом:

```
ftruncate(файловый дескриптор, размер);
```

Данная функция обрезает файл, файловый дескриптор которого передан в качестве первого параметра, до заданного размера, переданного в качестве второго параметра и указанного в байтах. Если процесс завершился успешно, то функция вернет `TRUE`, в противном случае — `FALSE`.

Мы указываем в качестве второго параметра ноль (строка **14**), что означает: полностью очистить файл — урезать его до нулевого размера.

Теперь ничто не мешает осуществить запись нового значения счетчика в файл counter.txt, что и происходит в строке **16** с помощью функции `fputs()`. Ее полный синтаксис выглядит следующим образом:

```
fputs(файловый дескриптор, данные);
```

Эта функция осуществляет запись данных в файл. В строке **16** она как раз используется для записи значения переменной `$s` в файл counter.txt.

В строке **19** происходит снятие блокировки с файла counter.txt уже знакомой вам функцией `flock()`, для этого в качестве ее второго параметра указывается цифра 3. Далее работа с файлом заканчивается, т. к. он закрывается использованием функции `fclose()`, после выполнения которой связь программа-дескриптор-файл будет разорвана (см. рис. 6.3, шаг 3). Синтаксис функции `fclose()`:

```
fclose(файловый дескриптор);
```

В строке **23** мы выводим значение счетчика (переменной `$s`) на экран.

Можно сказать, что весь код программы counter.php практически разобран, конечно, за исключением нескольких моментов: когда происходило открытие

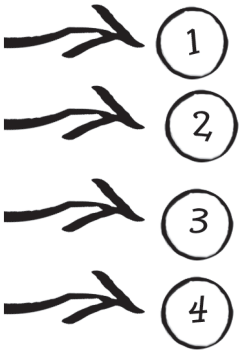
файла (строка **3**) с помощью функции `fopen()`, также была использована еще и функция `die()`, которая имеет следующий синтаксис:

```
die(текст сообщения об ошибке);
```

Данная функция выводит на экран текст, переданный в качестве параметра, и осуществляет немедленный выход из программы. Поэтому `die()` не используется сама по себе, а как правило, в сочетании с другими (в данном случае с функцией `fopen()`), как вариант защитного механизма. Таким образом, если функции `fopen()` не удалось открыть файл, то в этом случае она вернет отрицательный результат (`FALSE`), и сработает защитный механизм в лице функции `die()`, которая выведет текст "Не могу открыть файл", после чего выполнение программы будет прекращено. Если бы защитный механизм не использовался, то при дальнейшем выполнении программы она пыталась бы читать и записывать данные в никуда, т. к. файл не был открыт.

Когда две функции используются рядом, строится *логическое выражение*, которое нужно разделить с помощью специально предназначенного для этого *логического оператора*. В данном случае в качестве логического оператора используется `or` (в переводе с англ. "или").

Строка **3** будет обрабатываться препроцессором PHP следующим образом:



Выполнение программы на PHP начинается справа налево (в данном случае выполнение строки **3** не будет исключением).

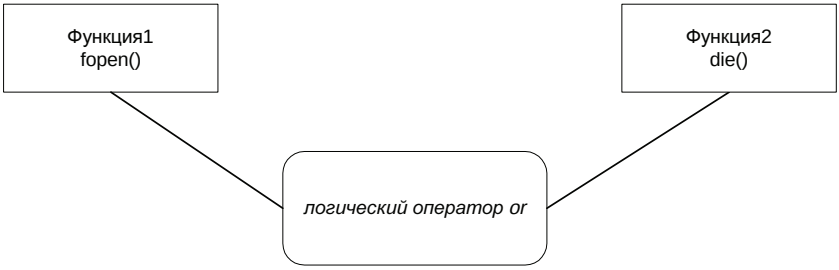
Сначала будет выполнена функция `fopen()`.

Если `fopen()` выполнилась удачно, т. е. произошло открытие файла и был возвращен файловый дескриптор, то далее будет выполняться строка **7**.

Если функции `fopen()` не удалось открыть файл, то она вернет `FALSE` в качестве результата, и в этом случае будет выполнена функция `die()`, после чего выполнение программы прекратится.

Логический оператор `or` позволяет строить логическую конструкцию (логическое выражение), состоящую из нескольких функций (в данном случае двух), в которой будет соблюдаться условие успешного выполнения одной из этих функций. Так как выполнение *всегда* начинается слева направо, то первой в логической конструкции *всегда* будет исполняться функция, находящаяся слева от `or`, если ее выполнение успешно, то условие логической конструкции соблюдено и, значит, вторую функцию выполнять не надо. В противном случае, если слева от оператора `or` функция вернула `FALSE`, произойдет выполнение функции справа, в результате чего будет опять соблюден принцип логической конструкции, который гласит, что одна из функций обязательно должна быть успешно выполнена. На рис. 6.5 изображена схема, поясняющая описанный процесс.

Запустите программу counter.php с помощью браузера, после этого нажмите несколько раз кнопку **Обновить**, и вы сможете наблюдать, как значение счетчика увеличивается (рис. 6.6).



Если Функция1 вернула FALSE, то только в этом случае будет выполнена Функция2

Рис. 6.5. Схема работы логической конструкции с использованием логического оператора `or`

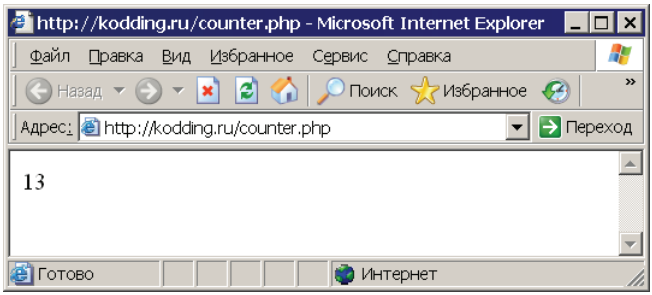


Рис. 6.6. Результат работы скрипта counter.php после нескольких нажатий кнопки **Обновить**

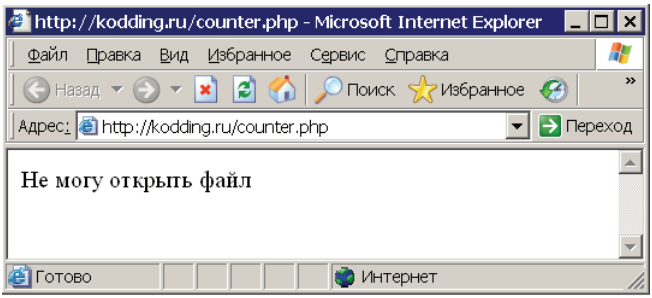


Рис. 6.7. Результат срабатывания защитного механизма в лице функций `die()` при неудачной попытке открыть файл counter.php

Если по какой-то причине при открытии файла возникнет ошибка, то вы увидите следующее — рис. 6.7.

Основная работа сделана, осталось только подключить счетчик к разработанной ранее HTML-страничке (см. рис. 6.1). Для этого необходимо изменить файл mypage.html следующим образом (см. строки с **14** по **22**) — листинг 6.3.

Листинг 6.3. Измененный файл mypage.html

```
1  <html>
2
3  <head>
4    <title>Домашняя страничка Иванова Пети</title>
5  </head>
6  <body>
7
8  <H1>Домашняя страничка Иванова Пети</H1>
9  <BR>
10 <BR>
11 Привет!
12 Меня зовут Петя.
13 Я изучаю PHP.
14 <BR><BR><BR>
15 <?php
16 //Выводим поясняющую надпись
17 echo "Данную страницу посетило уже: ";
18 //в середине поясняющей надписи выводим значение счетчика
19 require_once("counter.php");
20 //окончание поясняющей надписи
21 echo " человек";
22 ?>
23 </body>
24 </html>
```

Как видите, появилось несколько новых строчек кода. В строке **14** был три раза добавлен стандартный HTML-тег перевода строки `<BR>`. В строке **15** объявляется фрагмент кода на PHP, затем в строке **17** выводится текст с помощью `echo`. А вот в строке **19** используется новая функция языка `require_once()`, которая предназначена для подключения модуля, имя которого передается в качестве параметра и должно быть заключено в кавычки. В качестве модуля может выступать как программа на PHP, так и обычный HTML-документ. Таким образом, в строке **19** мы подключаем разработанный ранее модуль counter.php.

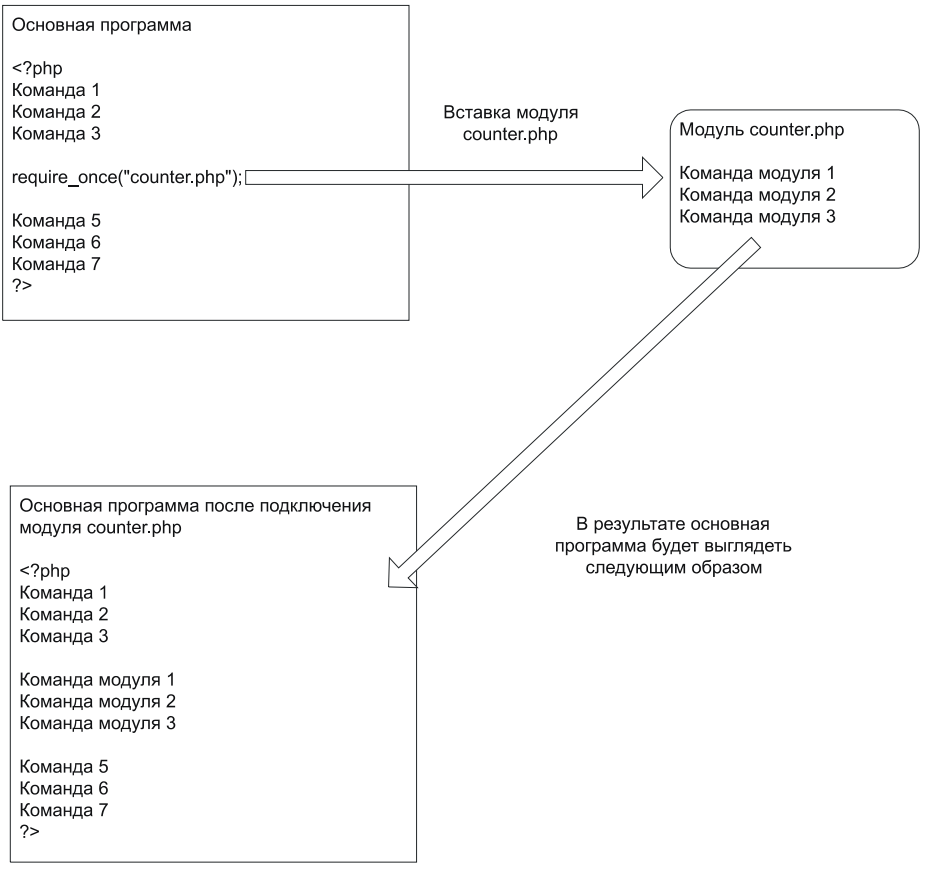


Рис. 6.8. Принцип подключения модуля в программу

Вообще понятие модуля достаточно условно, как правило, так называется обычная PHP-программа или ее фрагмент, которая специально была разработана для работы в составе другой программы. То есть модули изначально ориентированны на работу в составе чего-то.

Подключение модуля аналогично тому, если бы мы просто скопировали содержимое подключаемого файла и вставили в программу, откуда происходит его вызов. На рис. 6.8 изображена схема, описывающая этот процесс.

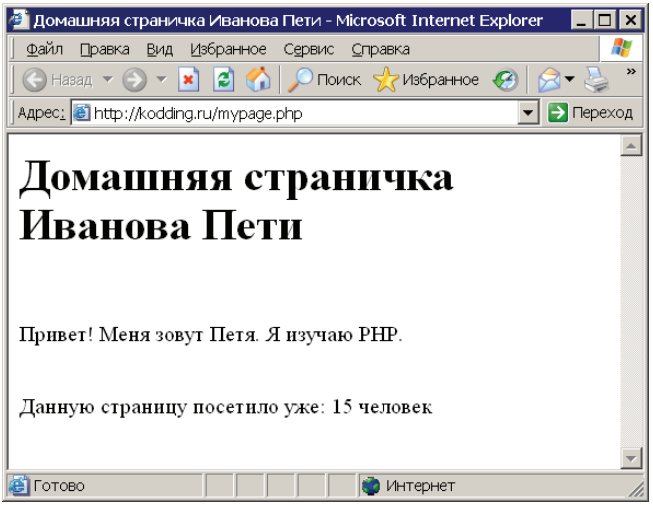


Рис 6.9. Окончательный вариант Web-странички с размещенным на ней счетчиком посещений

Идем далее. В строке **21** опять происходит вывод текста с помощью `echo`. Таким образом, значение счетчика будет находиться в следующем тексте:

```
Данную страницу посетило уже  ЗНАЧЕНИЕ СЧЕТЧИКА  человек
```

В строке **22** указано окончание фрагмента кода на PHP. Теперь самое главное — нужно поменять расширение файла mypage с html на php, потому что теперь он содержит полноценную PHP-программу. После запуска mypage.php в браузере вы сможете увидеть следующий результат — рис. 6.9.

*В качестве эксперимента вы можете скопировать содержимое файла counter.php в строку **19** файла mypage.php (переименованный mypage.html). Только надо учесть, что выделять код в counter.php надо без открывающего `"<?php"` и закрывающего `"?>"` тегов PHP, т. к. они в mypage.php уже имеются. Но, как вы можете заметить, при использовании функции `require_once()` наличие этих тегов в подключаемом модуле воспринимается препроцессором PHP без проблем. Как видите, еще один плюс в сторону удобства PHP.*

6.2. Счетчики, которые не надо разрабатывать

Предыдущая часть главы получилась очень насыщенной новой информацией. Сейчас мы немного отвлечемся и поговорим о готовых решениях, которые позволят вам разнообразить свой сайт. На многих сайтах вы можете видеть стандартные счетчики, которые показывают не только информацию о количестве посетивших данную страницу, но и позволяют посмотреть статистику по сайту, например, количество уникальных посетителей за определенный промежуток времени. Если вы щелкнете левой кнопкой мыши на счетчике, расположенном в левом части главной страницы **www.vr-online** (рис. 6.10), то увидите следующую информацию — рис. 6.11.

На рис. 6.11 изображено не что иное, как система статистики по сайту, которая позволяет оценить рейтинг популярности сайта по отношению к другим сайтам схожей с ним тематики, а также путь развития сайта — растет он в рейтинге или падает. Таким образом, простой маленький счетчик может быть очень полезным, хотя с виду это и не скажешь, а самое главное, вы можете спокойно воспользоваться услугами такой системы и абсолютно бесплатно, для этого вам всего лишь нужно будет зарегистрироваться в ней.

Рис. 6.10. Счетчик от **mail.ru**, размещенный на **vr-online.ru**

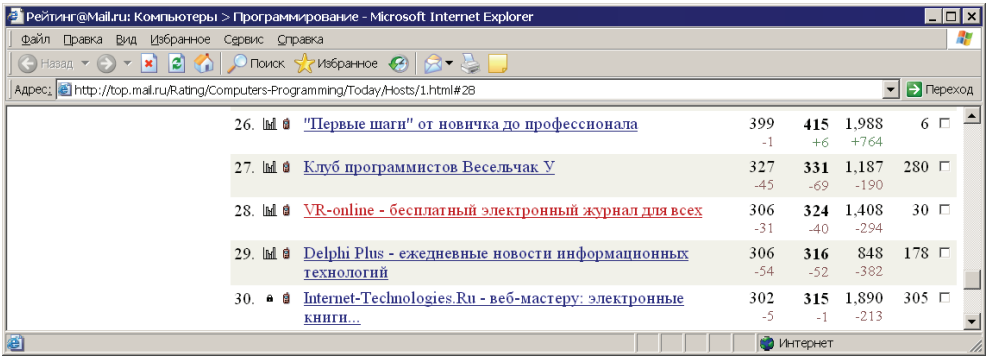
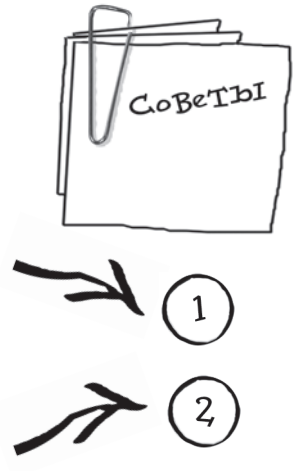


Рис. 6.11. Статистика посещений сайта

Систем статистики достаточно много, но общий принцип у них всех похожий:



Сначала вы выбираете тематику вашего сайта. Когда вы регистрируете свой сайт в определенной системе, допустим Mail или Rambler, он помещается в так называемый каталог, который представляет собой набор различных тем, например, на рис. 6.12 вы можете увидеть каталог ***mail.ru****.*

Заполняете данные о себе и о сайте в специальной форме, также указываете e-mail для связи с вами и подтверждаете свое желание зарегистрироваться в каталоге нажатием специальной кнопки (причем этот шаг может быть совмещен с предыдущим, например, так осуществлен процесс регистрации у ***rambler.ru)****.*

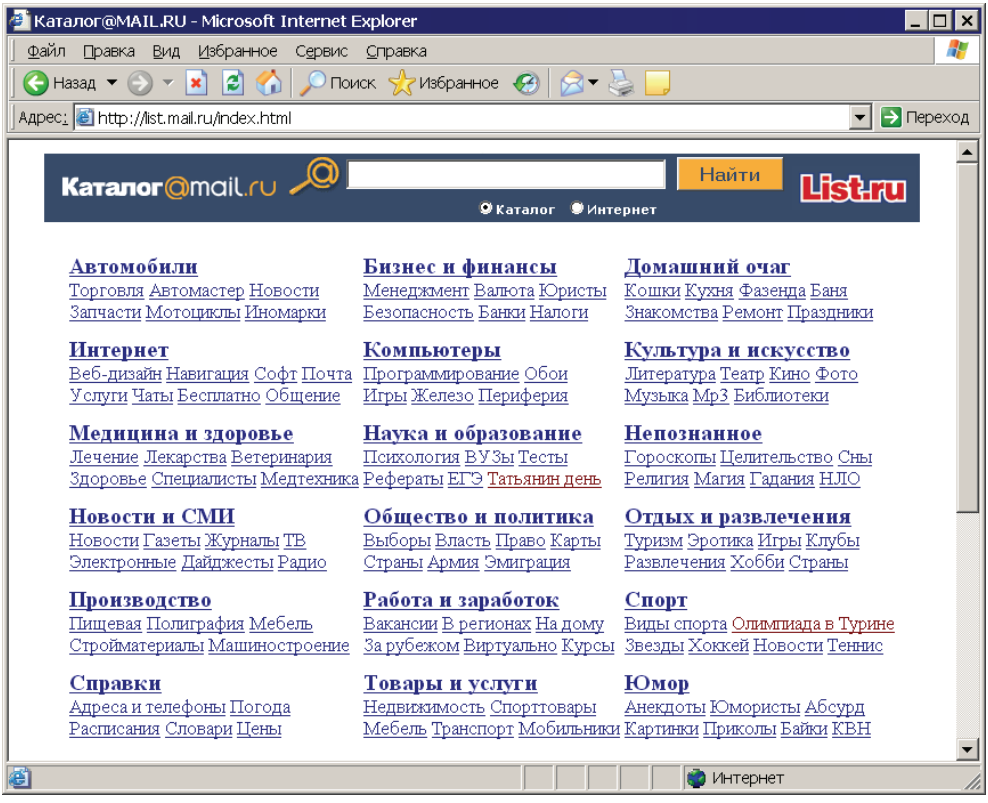


Рис. 6.12. Каталог сайтов **mail.ru**

3

Через некоторое время, после того как администратор сервиса (или системы, в которой вы зарегистрировались) удостоверится в том, что ваш сайт не несет ничего противозаконного, вам на почту поступит дополнительная информация, по поводу того, как разместить счетчик. На рис. 6.13 вы можете увидеть этап выбора вида счетчика при регистрации в каталоге ***rambler.ru.***

Замечание

Для того чтобы начать регистрацию на ***mail.ru****, вам необходимо зайти на этот сайт и выбрать раздел* ***КАТАЛОГ****. Чтобы зарегистрироваться на* ***rambler.ru****, вам необходимо зайти на* ***www.partner.rambler.ru*** *и далее выбрать пункт* ***ДОБАВИТЬ САЙТ****.*

Рис. 6.13. Выбор варианта счетчика при регистрации в каталоге **rambler.ru**

6.3. Информеры

В Интернете вы можете найти очень удобную штуку под названием *информер* — это элемент, который отображает некоторый вид информации на вашем сайте (например, прогноз погоды или курс доллара). Для его использования достаточно вставить на Web-страничку несколько строк готового кода (как со счетчиком). Причем всю работу выполняет специальный сервер, а размещенный вами информер только обращается к этому серверу и выводит готовый результат.

Некоторые сайты предлагают платные информеры, это совершенно странный подход, т. к. можно найти их аналоги, использование которых совершенно бесплатно и по качеству они ни в чем не будут уступать платным.

6.3.1. Информеры от Rambler

Для доступа к информерам от Rambler достаточно ввести в браузере адрес **www.partner.rambler.ru\informers**, после чего вы попадете на следующую страничку — рис. 6.14.

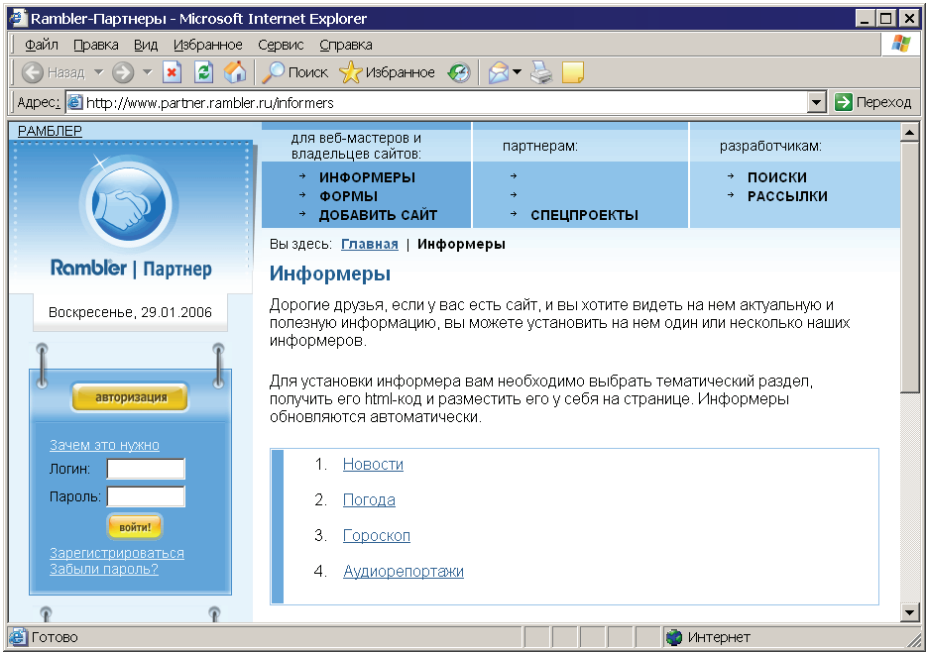


Рис. 6.14. Информеры от Rambler

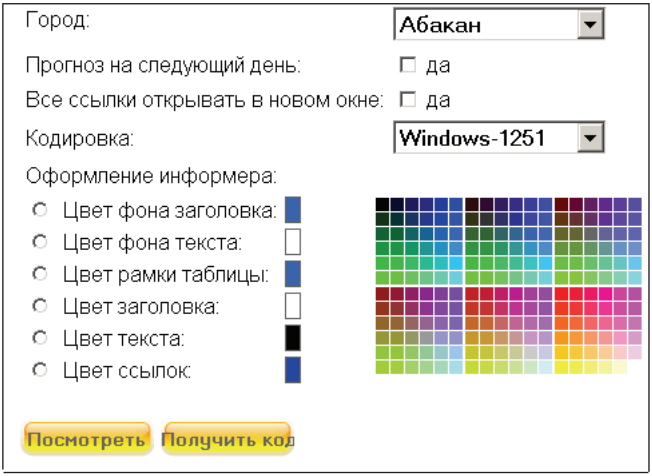


Рис. 6.15. Элементы настройки информера **Погода**

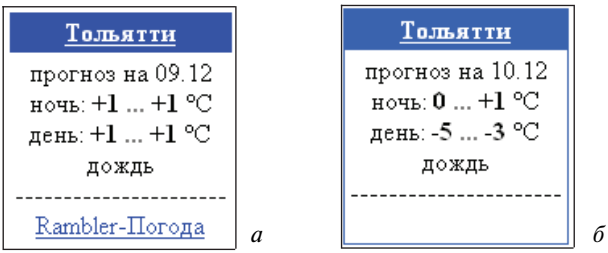


Рис. 6.16. Два варианта информера **Погода** с различной цветовой гаммой: *а* — стандартные цвета; *б* — цвета подобраны таким образом, что надписи **Rambler-Погода** не видно

Как видите, у нас есть из чего выбрать. Давайте рассмотрим установку информера, который показывает прогноз погоды, работа с остальными информерами аналогична. Выберите **Погода**, вам станут доступны элементы настройки (рис. 6.15).

Вам остается только выбрать город (из выпадающего списка), прогноз погоды для которого вы хотите, чтобы отображался на вашем сайте. Затем выбрать нужную цветовую гамму, причем ее можно подобрать таким образом, что надпись **Rambler-Погода**, которая автоматически отображается на информере, сольется с его фоном и будет не видна (рис. 6.16).

С помощью кнопки **Посмотреть** вы сможете осуществить предварительный просмотр информера, т. е. увидеть, как он будет выглядеть у вас на сайте. Чтобы разместить информер, необходимо нажать кнопку **Получить код** и скопировать полученный код в нужное место вашего сайта (это HTML-код, поэтому он будет работать даже в обычном HTML-документе).

*Есть также отличная альтернатива информеру **Погода от Rambler**. Это информер от **GISMETEO**, доступный по адресу **http://informer.gismeteo.ru/**, там достаточно широкий набор различных вариантов, например, очень интересна Flash-версия.*

Вы можете самостоятельно поэкспериментировать с остальными информерами, например, можно разместить информер от **Rambler** под названием **Новости**, который будет отображать новости компьютерного мира на вашем сайте, и это всего лишь с помощью нескольких щелчков мыши.

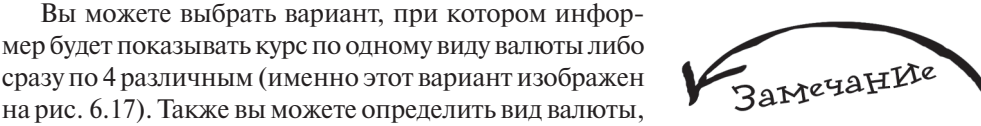
6.3.2. Курс валют

Курс валют — очень полезный информер, особенно если вы предоставляете какие-то услуги, например, такая информация будет не заменима для сайта туристического агентства. Отличные варианты информера **Курс валют** можно получить на сайте **http://www.othello.ru/currency/informs.htm** (рис. 6.17).

Вы можете выбрать вариант, при котором информер будет показывать курс по одному виду валюты либо сразу по 4 различным (именно этот вариант изображен на рис. 6.17). Также вы можете определить вид валюты, которая будет отображаться с помощью информера, и один из нескольких предложенных вариантов цветовой расцветки.

Есть альтернативный вариант, при котором вы только получаете информацию по курсам, а отображаете ее своими силами. Такую возможность предоставляет сайт **www.rbk.ru**, на нем вы можете обратиться к XML-файлу, в котором содержатся курсы абсолютно по всем видам валют. Введите в строке браузера **http://www.cbr.ru/scripts/XML_daily.asp**, вы попадете на следующую страничку — рис. 6.18.

Так выглядит XML-файл. Для того чтобы получить из него курс доллара, необходимо использовать следующий код — листинг 6.4.



*Обратите внимание, в листинге 6.4 строка **8** не поместилась полностью, поэтому ее фрагмент пришлось перенести чуть ниже. В своей программе этот код вам необходимо написать в одну строчку, чтобы он выглядел следующим образом:*

```
preg_match('#<CharCode>USD</CharCode>.*?<Value>(.*?)</Value>#si', $data,$data);.
```

Далее в листингах перенесенные строки будут помечаться звездочкой (*).

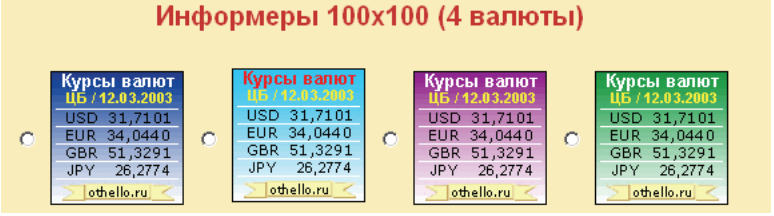


Рис. 6.17. Несколько вариантов информера **Курс валют** от **http://www.othello.ru/currency/informs.htm**

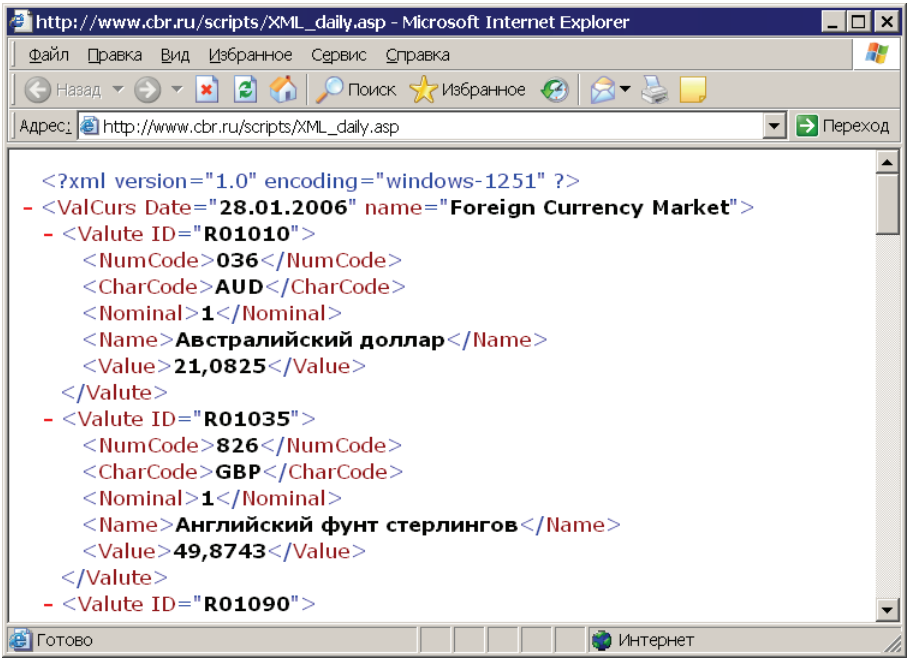


Рис. 6.18. Курс валют от **www.rbk.ru**

Листинг 6.4. Код получения курса доллара из XML-файла

```
1  <?
2  $fh = @fopen('http://www.cbr.ru/scripts/XML_daily.asp','rt');
3  if($fh)
4  {
5    while(!feof($fh)) @$data.=fread($fh,4096);
6    fclose($fh);
7
8*   preg_match('#<CharCode>USD</CharCode>.*?<Value>(.*?)</Value>#si',
   $data,$data);
9    echo 'Текущий курс USD '.@$data[1].' рублей';
10 }
11 else echo 'Текущий курс USD не доступен';
12 ?>
```

Некоторые фрагменты кода вам будут знакомы, некоторые — нет. В программе используются условная конструкция `if` (она рассматривается в *главе 9*) и регулярные выражения (в данной книге они не рассматриваются, т. к. это сложная тема для начинающих, дополнительную информацию по ней вы можете получить, обратившись по адресу **http://ru.php.net/preg_match**, **http://phpfaq.ru/regexp** или введя в любом поисковике фразу `Регулярные выражения`).

Если вы хотите, например, получить курс евро, то замените в строке ***8*** *USD на EUR.*

6.3.3. Цены на российские автомобили

Последний информер, рассматриваемый в этой главе, — это тот, который отобра-жает информацию по ценам на российские автомобили. Он изображен на рис. 6.19.

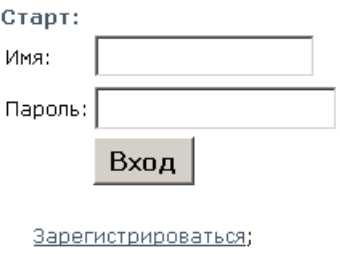
Данный информер можно найти по следующему адресу: **http://tlt.ru/informer/**. Размещается он простым копированием кода, который также находится на приведенном ранее сайте.

Цены на автомобили ВАЗ в Тольятти, (тыс. руб.)				
28.01.2006	Мин	Срд	Мак	Кол
ВАЗ 2112	220	242	308	160
ВАЗ 2107	125	137	150	131
ВАЗ 21101	0	228	251	130
ВАЗ 2114	0	191	209	115
ВАЗ 1111	86	98	115	112
ВАЗ 2115	0	200	212	92
ВАЗ 2111	222	239	251	78
ВАЗ 21104	0	234	307	71
ВАЗ 1118	0	216	267	66
ВАЗ 2105	118	125	128	62
ВАЗ 2113	184	191	200	59
ВАЗ 21214	203	203	229	57
ВИС 23461	210	255	461	49
Шеви-Нива	9	304	520	47
ВАЗ 2131	215	265	566	42
снижен.	стабил.	повышен.		
PRICE.TLT.RU				

Рис. 6.19. Информер, отображающий информацию по ценам на российские автомобили

Глава 7
Все, что нужно знать о формах

С формами вы наверняка много раз встречались, когда заходили на различные сайты, т. к. они являются практически неотъемлемой их частью. Вводите ли вы электронный адрес для подписки на какую-то рассылку, регистрируетесь ли вы на форуме или вводите пароль и имя пользователя на ваш почтовый ящик через окно браузера — вы используете формы. Например, на рис. 7.1 приведен фрагмент формы для входа на сайт **www.vr-online**.



Замечание

Работать с формой очень просто, для этого всего лишь нужно знать принципы, заложенные в ее основу.

Рис. 7.1. Пример формы

7.1. Назначение форм

Основная задача формы — это оживить страничку, придать ей некоторую интерактивность или, другими словами, дать возможность пользователю вести диалог с ней. Под диалогом подразумевается следующее: у пользователя запрашивается какая-то информация или его просят сделать выбор, он вводит информацию или

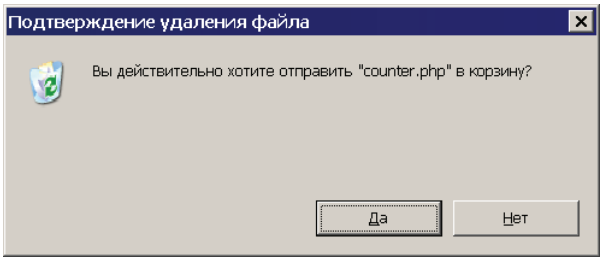


Рис. 7.2. Форма в стиле операционной системы Windows

делает выбор и в соответствии с этим достигает какого-то результата — либо он принял участие в голосовании, либо вошел на форум под своим "ником", либо подписался на рассылку и т. д.

Например, когда вы удаляете файл на своем компьютере (предположим, у вас установлена операционная система Windows), то прежде чем это произойдет, появится окно **Подтверждение удаления файла** (конечно, если вы специально не отключали данную возможность), и вы либо нажмете кнопку **Да**, тем самым подтверждая выбор и совершая действие, либо отказываетесь от него и нажимаете кнопку **Нет** (рис. 7.2). Это является простейшим примером использования формы и соответственно механизма диалога пользователя с операционной системой.

Web-форму можно сравнить с диалоговыми окнами Windows, т. к. форма представляет собой некоторое их подобие, только, как правило, мы не видим границ формы, потому что они условны, а видим лишь ее элементы, в которые пользователь вводит свои данные либо осуществляет какой-то выбор, после этого он обязательно нажимает кнопку подтверждения, таким образом отправляет свои данные или свой выбор серверу.

7.2. Создание формы

Вы можете удивиться, но форма является элементом языка гиперразметки документов (HTML), а PHP всего лишь предоставляет программисту механизм работы с ним. Объявление формы начинается со специального тега `<FORM>`, у которого есть 3 необязательных параметра. Заканчивается объявление формы закрывающимся тегом `</FORM>`. Полная запись создания формы выглядит следующим образом:

```
<FORM
        name="имя формы"
        action="путь к программе-обработчику формы"
        method="метод передачи данных">
Элементы формы - или тело формы
</FORM>
```

Рассмотрим назначение каждого параметра более подробно:

- `name` — имя формы;
- `action` — указывает имя программы (скрипта), которому будут отправлены данные формы. Отправка формы — это обычный запрос к серверу (наподобие того, что мы рассматривали на примере рис. 2.6), только вместе с запросом также передается дополнительная информация — данные формы. Более подробно об этом мы поговорим немного позже. Если параметр `action` не указать, форма отправляется текущей программе. На рис. 7.3 и 7.4 изображена схема, иллюстрирующая назначение параметра `action`.

Если пользователь запросил файл form.php, который содержит форму, и после этого нажал кнопку **Send**, то именно к этому скрипту произойдет обращение (т. к. при создании формы в качестве значения атрибута `action` было указано obrabotka.php — рис. 7.3).

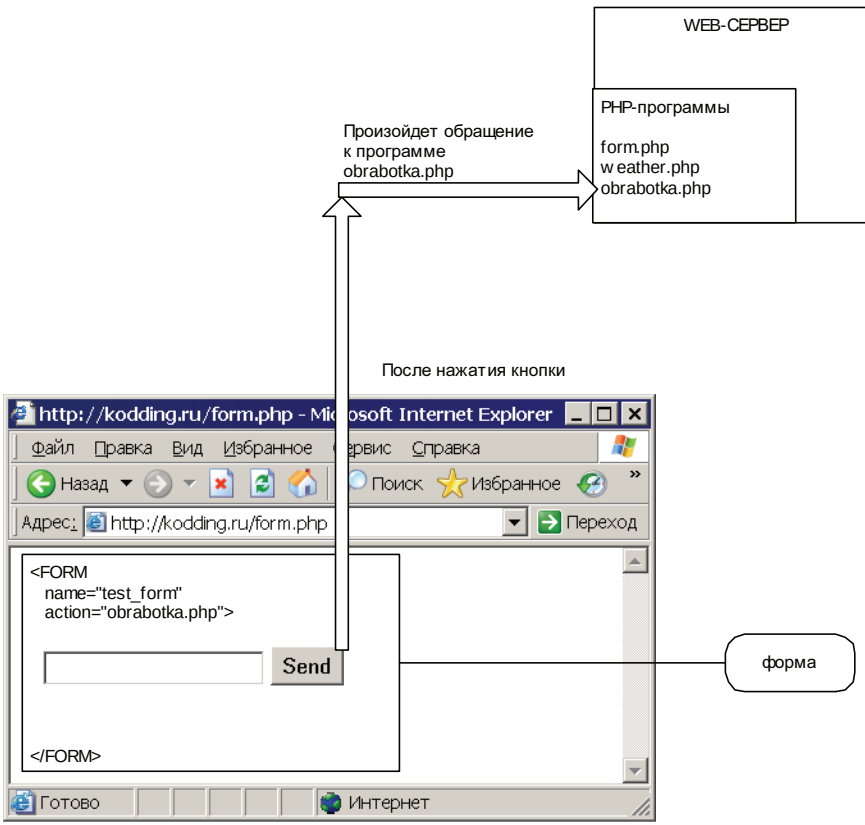


Рис. 7.3. Работа формы при указанном параметре `action`

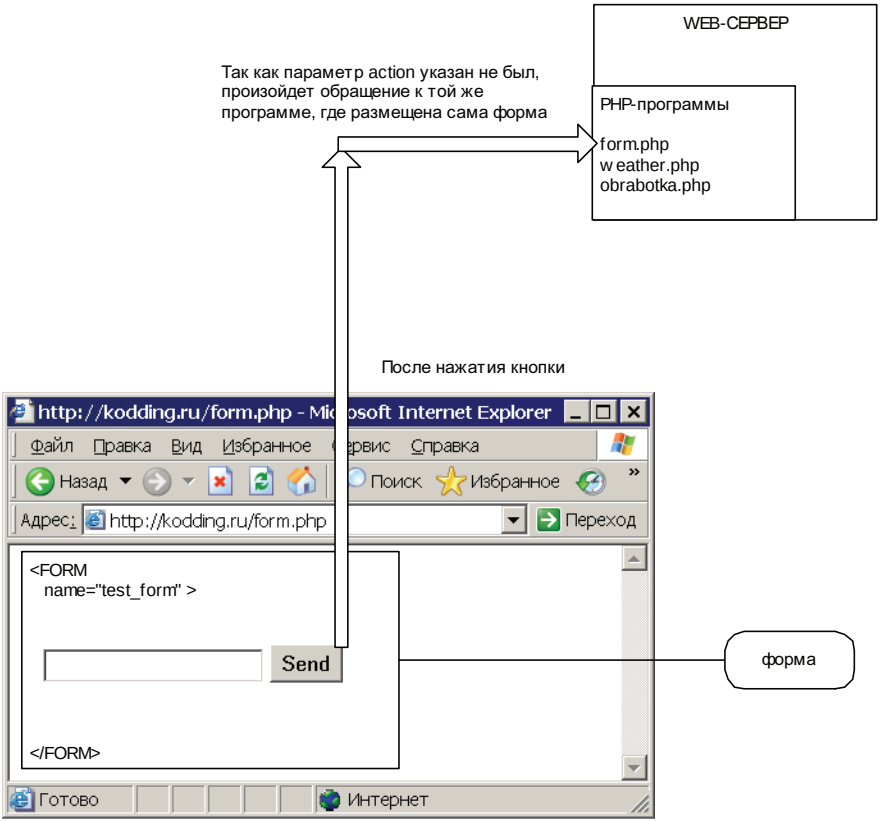


Рис. 7.4. Работа формы без параметра `action`

Теперь такая же ситуация, только параметр `action` не указан (рис. 7.4), значит, произойдет обращение к той же программе, где размещена сама форма, т. е. к программе form.php.

`method` — способ отправки формы на сервер. Их два: `GET` и `POST`, различия между ними будут рассмотрены чуть позже. Если метод не указан, то подразумевается `GET`.

На рис. 7.5 изображена упрощенная схема Web-формы.

Так как параметры `name`, `action` и `method` являются не обязательными, можно использовать сокращенную запись создания формы:

```
<form>
    <элементы формы>
</form>
```

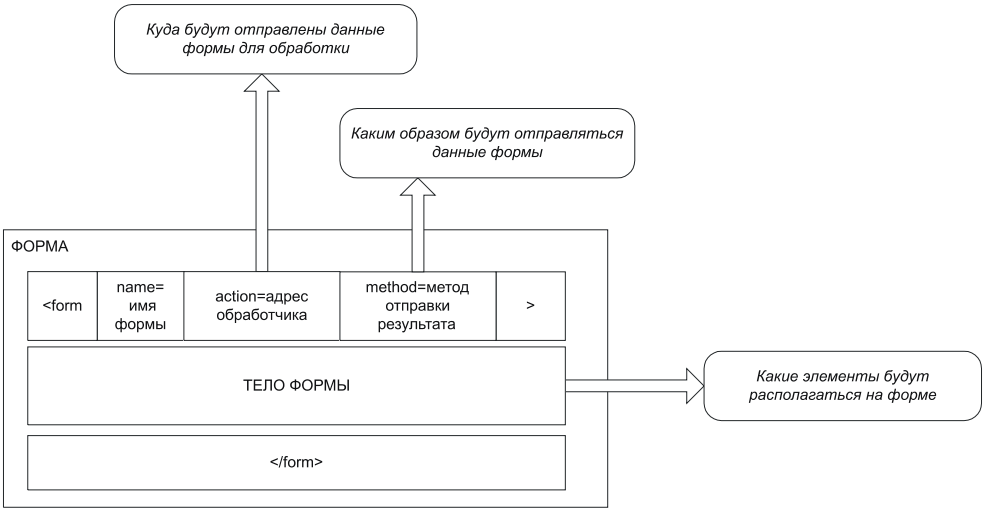


Рис. 7.5. Упрощенная схема Web-формы

В этом случае обработчиком данной формы будет та же программа — если мы говорим о файле с расширением php (или документ, если мы говорим о файле с расширением html), в которой эта форма и объявляется, данные будут отправляться методом `GET`.

На одной страничке может быть несколько форм, причем каждая из них может содержать абсолютно разное количество элементов. Но формы не могут быть вложенными, т. е. записать следующим образом:

```
<FORM name="form1">
            <FORM name="form2">
               <элементы формы form2>
            </form
   <элементы формы form1>
</form>
```

нельзя, т. к. это вызовет ошибку.

7.3. Простые элементы формы: поле ввода и кнопка

Основное назначение формы — быть контейнером для различных элементов. Один из самых распространенных элементов — **Edit** (Поле ввода) (рис. 7.6).

Рис. 7.6. Элемент формы **Edit**

Объявляется он с помощью тега <input>:

```
<input
    type="text"
    name="имя элемента"
    value="значение элемента"
    title="всплывающая подсказка"
    size=длина
    maxlength=максимальное количество элементов>
```

Описание параметров:

- type — тип элемента, в данном случае text, что означает поле для ввода;
- name — имя элемента, когда форма будет отправляться на сервер, данные для этого элемента будут отправлены в виде имя_элемента=значение_элемента;
- value — текст, который будет отображаться в поле ввода;
- title — всплывающая подсказка, которая будет отображаться при наведении курсора мыши на поле ввода;
- size — длина поля ввода, указывается числом;
- maxlength — максимальное число символов, которое можно будет ввести в данное поле.

Все параметры, за исключением type и name, являются необязательными.

Например, запись вида:

```
<input
        type="text"
        name="name_of_user"
        value="Введите ваше имя"
        title="Поле для ввода имени"
        size=40
        maxlength=50>
```

создаст следующее поле ввода — рис. 7.7.

Можно также использовать и сокращенную запись для создания этого элемента:

```
<input
    type="text"
    name="имя элемента">
```

Например, запись вида:

```
<input
        type="text"
        name="name_of_user">
```

Создаст следующее поле ввода — рис. 7.8.

С помощью тега <input>, как вы в дальнейшем убедитесь, можно создавать различные элементы, для этого необходимо всего лишь указывать для каждого из них различные значения в параметре type.

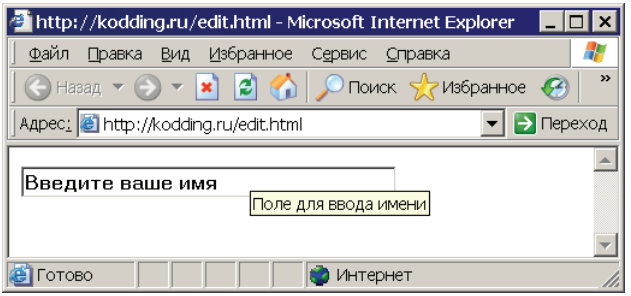


Рис. 7.7. Поле ввода с именем `name_of_user` и текстом по умолчанию

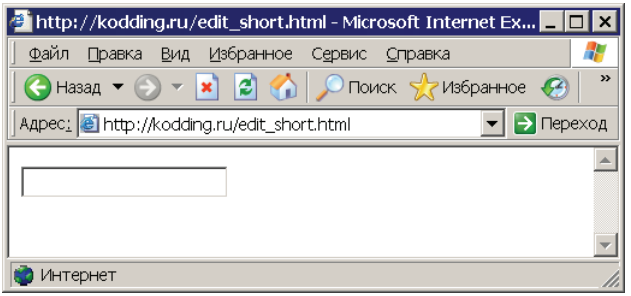


Рис. 7.8. Поле ввода с именем `name_of_user`

Так зачем же задавать необязательные атрибуты, когда и без них все великолепно работает? Потому что это показатель качественного программирования и признак того, что вы заботитесь о пользователях вашей программы. Например, те же самые подсказки помогают более точно понять предназначение элемента, а ограничение на максимальное количество введенных символов позволяет не получать лишние данные. Предположим, вы запрашиваете имя пользователя, очевидно, что оно не может состоять из 500 символов, но если вы не поставите ограничение, то теоретически пользователь сможет ввести такое длинное значение.

Все великолепно работает!

Результат заполнения формы отправляется на сервер по нажатию кнопки, поэтому она является одним из самых главных элементов. **Submit** (Кнопка) изображена на рис. 7.9.

Кнопка создается следующим образом (с помощью уже знакомого вам тега `<input>`):

```
<input
        type="submit"
        name="имя элемента"
        value="текст кнопки"
        title=" всплывающая подсказка " >
```

Рис. 7.9. Элемент формы **Submit**

Описание параметров:

- `type` — тип элемента, в данном случае `submit`, что означает кнопка отправки формы на сервер (или, другими словами, значений элементов формы);
- `name` — имя кнопки, когда результат будет отправляться на сервер, если параметр `name` указан, то также будет передана строка вида `имя_кнопки=текст_кнопки`;
- `value` — текст, который будет отображаться на кнопке, например, для кнопки, изображенной на рис. 7.9, значение этого параметра равно `Send`;
- `title` — всплывающая подсказка, которая будет отображаться при наведении курсора мыши на кнопку.

Параметры `title` и `name` являются необязательными. Параметр `value` также не обязателен, но если его не указывать, то отсутствие текста на кнопке может вызвать у пользователя непонимание ее назначения.

Например, запись вида:

```
<input
        type="submit"
        value="ВОЙТИ В СИСТЕМУ"
        name="enter_system"
        title="Нажмите, если вы ввели имя пользователя и пароль" >
```

Приведет к созданию кнопки, изображенной на рис. 7.10.

Можно также создать кнопку в виде картинки, это значит, что вы будете видеть обычную картинку, но при нажатии на нее левой кнопкой мыши она будет работать в точности, как обычная кнопка. Это делается следующим образом:

```
<input
        type="image"
        name="имя элемента"
        src="путь к файлу-рисунку"
        title=" всплывающая подсказка ">
```

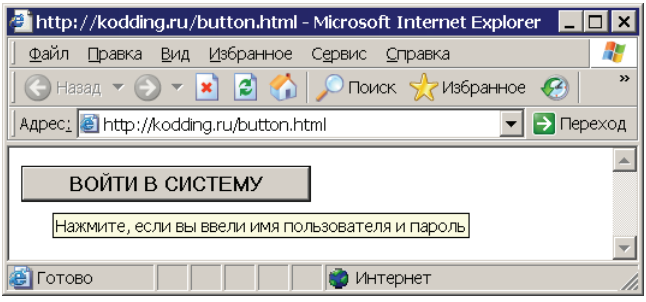


Рис. 7.10. Кнопка с именем `enter_system`, текстом на кнопке **ВОЙТИ В СИСТЕМУ** и подсказкой

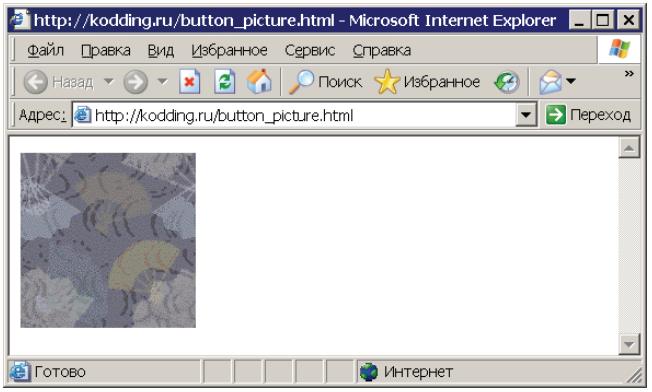


Рис. 7.11. Кнопка-картинка с именем `button_picture` и всплывающей подсказкой

Описание параметров:

- `type` — тип элемента, в данном случае `image`, что означает кнопка в виде картинки, нажатие которой вызывает отправку данных на сервер;
- `name` — имя кнопки;
- `src` — путь к файлу-рисунку для кнопки;
- `title` — всплывающая подсказка, которая будет отображаться при наведении курсора мыши на кнопку в виде картинки.

Параметры `name` и `title` являются необязательными. Например, запись вида:

```
<input
  type="image"
  name="button_picture"
  src="picture_button.bmp"
  title="Это картинка на самом деле кнопка, нажмите ее">
```

создаст кнопку в виде картинки, изображенную на рис. 7.11.

Замечание

В данном примере нужно, чтобы файл, содержащий картинку (picture_button.bmp), располагался в том же каталоге, что и файл с описанным ранее фрагментом кода.

7.4. Немного практики

Давайте напишем небольшую программу, предназначенную для работы с формами. Она будет состоять из двух файлов. Первый — это обычный HTML-документ, который будет предназначен для отображения формы. Второй — это программа на PHP, которая будет вызываться после того, как пользователь нажал на форме кнопку **ВОЙТИ В СИСТЕМУ**. Ее предназначение заключается в выводе надписи "ПРИВЕТ!".

Создайте файл с именем form.html (листинг 7.1).

Листинг 7.1. Файл form.html

```
<html>
<head>
  <title>Первая программа для работы с формой</title>
</head>
<body>

<!--Объявляем форму с обработчиком program_for_form.php-->
<FORM
   name="my_form"
   action="program_for_form.php">

   Представьтесь:
   <!--Объявляем поле для ввода имени-->
   <input
       type="text"
       name="name_of_user"
       value=""
       title="Поле для ввода имени">

   <!--Переходим дважды на следующую строку-->
   <BR>
   <BR>

   <!--Объявляем кнопку-->
   <input
       type="submit" value="ВОЙТИ В СИСТЕМУ"
       name="enter_system"
       title="Нажмите, если вы ввели имя, чтобы увидеть результат" >

</FORM>

</body>
</html>
```

Теперь необходимо создать второй файл с именем program_for_form.php (листинг 7.2).

Листинг 7.2. Файл program_for_form.php

```
1  <html>
2  <head>
3    <title>Обработчик program_for_form.php</title>
4  </head>
5  <body>
6
7  <?php
8  echo "ПРИВЕТ!";
9  ?>
10
11 </body>
12 </html>
```

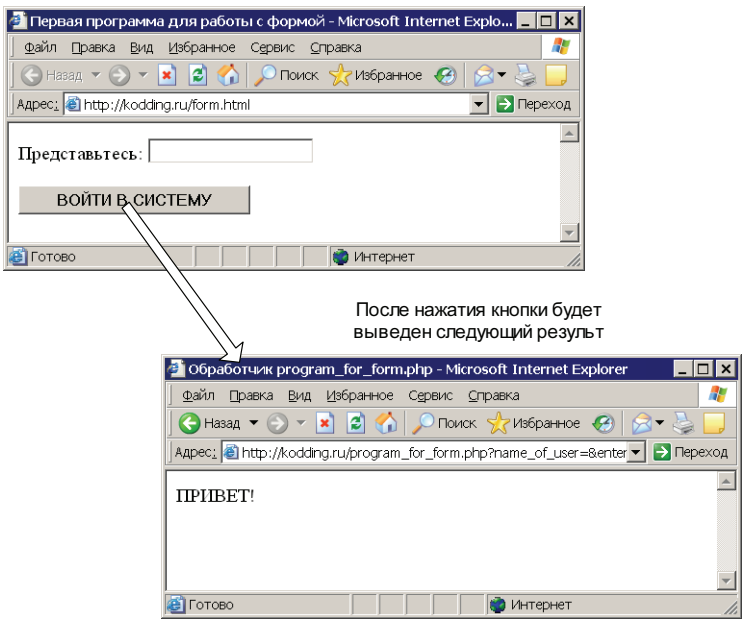


Рис. 7.12. Взаимодействие form.html и program_for_form.php

Чтобы протестировать разработанную программу, откройте файл form.html через браузер, введите имя и нажмите кнопку **ВОЙТИ В СИСТЕМУ** (рис. 7.12).

Немного модифицируем пример — сделаем так, чтобы после слова "ПРИВЕТ!" выводилось имя, которое ввел пользователь. Для того чтобы получить данные формы, в PHP-программе можно использовать 1 из 3 массивов:

- `$_GET` — применяется, когда данные формы передаются методом `GET`;
- `$_POST` — используется, когда данные формы передаются методом `POST`;
- `$_REQUEST` — применяется как альтернативный вариант двум предыдущим, т. к. заменяет их, не важно каким методом передавались данные, они все равно будут доступны в этом массиве.

Так как в нашем примере у кнопки установлен параметр `value`, то можно получить его значение с помощью `$_REQUEST[имя кнопки]` или `$_GET[имя кнопки]`.

Имя элемента массива, в котором хранятся данные об имени пользователя, такое же, как имя элемента формы, в которое было введено это самое значение. В силу того, что у поля ввода в предыдущем примере было имя `name_of_user`, при обращении к `$_REQUEST['name_of_user']` или `$_GET['name_of_user']` (если метод не указан, то данные отправляются методом `GET`, а метод указан не был) можно получить значение, которое ввел пользователь. Чтобы добавить новую возможность в программу, необходимо заменить строку **8** на следующий код:

```
echo "ПРИВЕТ! ".$_GET['name_of_user'];
```

Результат выполнения измененной программы представлен на рис. 7.13.

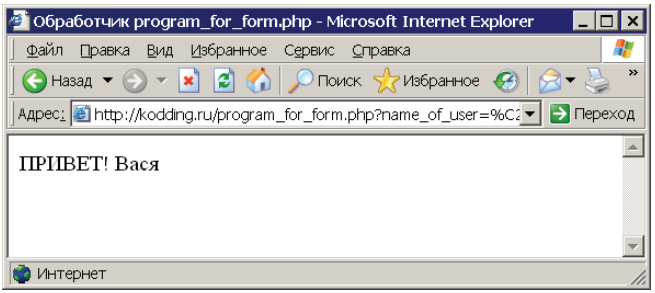


Рис. 7.13. Получение имени пользователя, введенного в поле ввода с именем `name_of_user` и вывод его на экран

7.5. Методы отправки данных формы

Как было сказано ранее, форма может быть отправлена на сервер двумя способами (методами): `GET` и `POST`. В данном разделе будут рассмотрены их особенности и отличия.

7.5.1. Метод GET

Метод GET означает, что данные будут передаваться присоединенными к имени программы обработчика, указанного в параметре action формы, то есть:

```
скрипт, указанный в action + данные формы
```

Чтобы сервер смог их отличить, они присоединяются по специальным правилам.

- *Сначала после имени программы-обработчика ставится знак вопроса (?), таким образом указывается, что будут передаваться данные. Например, если в параметре* `action` *формы указано* `program_for_form.php`*, то результат будет следующим:*

```
program_for_form.php?
```

- *Потом указывается имя первого объявленного на форме элемента, например, если первым указано поле для ввода с именем* `for_name`*, то результат будет следующим:*

```
program_for_form.php?for_name
```

- *Далее идет знак равно, который разделяет имя элемента и его значение:*

```
program_for_form.php?for_name=
```

- *После знака равно идет само значение элемента, например:*

```
program_for_form.php?for_name=Vasya
```

- *Если элемент на форме один, то формирование запроса считается оконченным, и он отправляется на сервер.*
- *Если же на форме несколько элементов, то передача следующего значения отделяется от первого знаком амперсанда (&), а далее все происходит по вышеописанной схеме: имя элемента, знак равно, значение элемента. Предположим, что после поля ввода с именем* `for_name` *располагается такое же поле ввода, только с именем* `password`*, тогда строка, сформированная методом* `GET` *для отправки на сервер, будет следующей:*

```
program_for_form.php?for_name=Vasya&password=123
```

- *Таким же образом с помощью знака & отделяются все последующие элементы — третий, четвертый и т. д., если конечно они есть.*

Метод GET удобен тем, что вы можете видеть сформированную браузером строку запроса для передачи на сервер и соответственно напрямую передавать обработчику формы данные, минуя саму форму, для этого достаточно сформировать вручную в браузере строку по описанному ранее правилу. Стоит отметить, что русские буквы передаются в закодированном виде, так что вы не увидите `for_name=Вася` в строке запроса при передаче данных методом GET (для кодирования используется

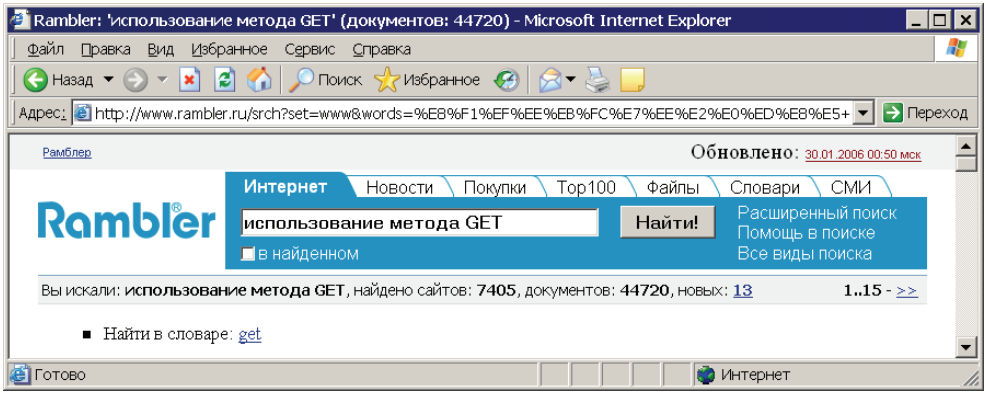


Рис. 7.14. При работе с поисковыми системами используется метод `GET`

шестнадцатеричное представление каждой русской буквы). Например, строка запроса для формы, состоящей из одного поля с именем `for_name`, содержащем значение `Вася`, будет следующей:

```
program_for_form.php?for_name=%C2%E0%F1%FF
```

Именно метод `GET` используется, когда вы ищете что-то с помощью поисковых систем. После того как в строку поиска введено какое-то слово или фраза (а это не что иное, как поле ввода, размещенное на форме) и нажата кнопка **ПОИСК** (после этого запрос будет отправлен на сервер), вы можете наблюдать, какой запрос формируется в строке браузера, например, при вводе "использование метода `GET`" в поисковой системе Rambler можно увидеть следующий результат (обратите внимание на строку запроса браузера) — рис. 7.14.

7.5.2. Метод POST

В главе 8 будет рассмотрено, каким образом можно увидеть данные, передаваемые методом `POST`.

При использовании метода `POST` данные отправляются незаметно для пользователя в теле запроса. Сам запрос при этом будет состоять из трех частей (рис. 7.15):

- запрос к программе-обработчику, указанной в параметре `action` формы;
- сообщение о том, какой объем данных будет передаваться;
- передача самих данных.

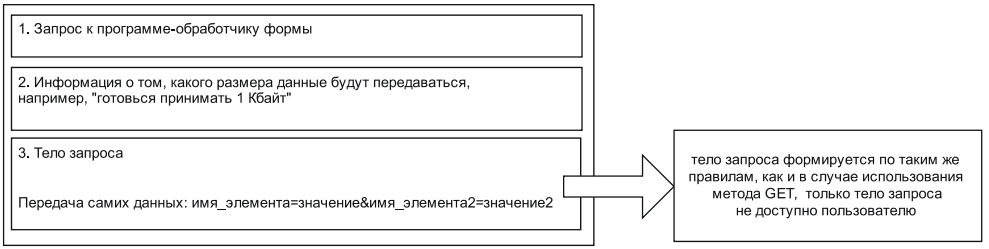


Рис. 7.15. Структура запроса на сервер при передачи формы методом `POST`

Давайте рассмотрим интересный пример объявления формы:


```
<FORM    action="program_for_form.php?razdel=
1&punkt=10" method=POST>
         <input type="text" name="for_name">
         <input type="submit" value="ВОЙТИ">
</FORM>
```

Как видите, несмотря на то, что объявлена `POST`-форма (это краткое название формы, данные которой отправляются методом `POST`), в параметре `action` вместе с программой-обработчиком указывается дополнительная информация, сформированная по всем правилам метода `GET`. Это не ошибка, а очень удобная возможность, которая позволяет использовать сразу два способа отправки данных на сервер. На самом деле, весь секрет достаточно прост. Когда рассматривались составные части `POST`-запроса на сервер, под первым пунктом было выделено "обращение к скрипту-обработчику", а это не что иное, как обычный `GET`-запрос. Таким образом, `POST`-форму можно использовать для передачи данных методом `GET`, но не наоборот.

7.5.3. Что лучше: GET или POST?

Метод `GET` рекомендуется использовать, когда происходит запрос некоторой информации (например, в поисковом сервере) или вы создаете что-то, содержащее иерархическую структуру, скажем, форум или сайт с множеством подразделов (например, там может быть раздел **Скачать**, подразделы **Программы**, **Документация**, которые в свою очередь могут содержать несколько подразделов). Он предпочтительнее, потому что тогда можно поставить закладку средствами браузера на нужную информацию, либо отправить ссылку, сформированную в строке запроса браузера, другу или коллеге по работе.

Недостатком метода GET является ограничение на объем передаваемых данных, у POST это ограничение тоже имеется, но оно слабее — так что с помощью POST-метода можно передавать даже целые файлы.

Метод POST лучше использовать, когда не следует отображать пересылаемые данные, например, при авторизации на форуме, когда вы вводите свой логин и пароль (хотя на самом деле и в этом случае можно просмотреть, что именно отправляется на сервер, данный вопрос освещается в *главе 8*).

Конечно, однозначных рекомендаций по поводу использования того или иного метода дать сложно, многое зависит от ваших личных предпочтений и опыта. Я думаю, после прочтения данной книги у вас будет больше ясности в отношении этого вопроса.

7.6. Остальные элементы формы

7.6.1. Поле для ввода пароля

Объявляется точно так же, как и поле ввода, только в параметре type необходимо указать значение password. Например, следующая запись:

```
<input
        type="password"
        name="pas_of_user"
        value="ваш пароль"
        title="Поле для ввода пароля"
        size=20
        maxlength=10>
```

создаст поле для ввода пароля (все вводимые символы будут выглядеть как жирные точки) с именем pas_of_user, значением ваш пароль, подсказкой Поле для ввода пароля, длиной 20 символов, и максимальным количеством символов для ввода, равным 10 (рис. 7.16).

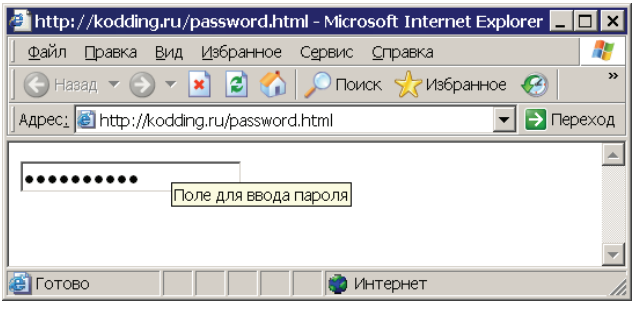


Рис. 7.16. Поле для ввода пароля

Можно использовать сокращенную запись, она такая же, как и в случае создания обычного поля ввода.

7.6.2. Переключатель (Radio button)

Элемент для выбора одного варианта из нескольких заранее предопределенных. Для создания переключателя используется запись вида:

```
<input type=radio
    name="имя"
    value="значение"
    [checked]
> Текст для переключателя
```

Описание параметров:

- `type` — тип элемента, в данном случае `radio`, что означает создание переключателя;
- `name` — имя элемента (данный параметр используется для создания группы переключателей);
- `checked` — является необязательным параметром, может указываться только для единственного элемента из группы (элементы одной группы должны иметь одинаковое имя) и означает, что данный элемент будет выбран;
- под текстом подразумевается любой текст — он будет следовать сразу после переключателя.

Например, запись вида:

```
<input name="group1" type="radio" value="Yes">Да
<input name="group1" type="radio" value="No">Нет
<input name="group1" type="radio" value="I don't know" checked>Не знаю
```

создаст группу переключателей с именем `group1`, состоящую из трех элементов, причем третий элемент будет выбран (рис. 7.17).

Данные для переключателей передаются на сервер в виде `имя=значение`, причем имя передается один раз, т. к. оно у всей группы одинаковое, а значением будет значение параметра `value` для выбранного переключателя. Если бы рассмотренный

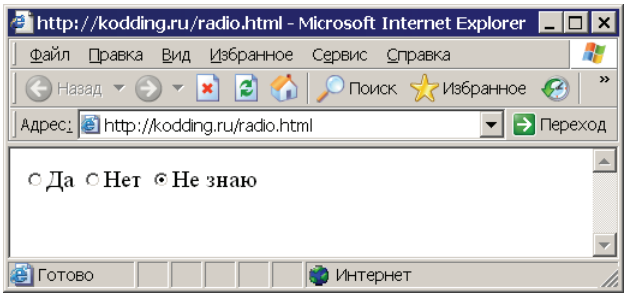


Рис. 7.17. Группа переключателей

ранее пример был размещен на форме с обычной кнопкой, то после ее нажатия на сервер ушел бы следующий результат:

```
group1=I don't know
```

7.6.3. Флаг (CheckBox)

Данный элемент предназначен для организации выбора одного или нескольких вариантов из заранее предопределенных. Объявляется следующим образом:

```
<input
   type="checkbox"
   name="имя"
   value="значение"
   [checked]
   > Текст флага
```

Все параметры вам должны быть хорошо знакомы, т. к. они уже были рассмотрены ранее. Остановимся только на `checked` — он применяется с такой же целью, как и для переключателя, только здесь данный параметр может быть использован столько раз, сколько вы хотите, чтобы было установлено флагов, т. к. применение флагов подразумевает множественный выбор. Давайте рассмотрим следующий пример:

```
<input  type="checkbox"  name="chose1" value="flag1"  checked> Флаг1
<input  type="checkbox"  name="chose2" value="flag2"> Флаг2
<input  type="checkbox"  name="chose3" value="flag3"  checked> Флаг3
```

который создает 3 флага, с именами `chose1`, `chose2`, `chose3`, причем два из них будут установлены (рис. 7.18).

Как частный случай, флаг может быть использован для организации выбора одного варианта (когда он размещается в единственном экземпляре), с переключателем так не получится, потому что он предназначен для выбора минимум между двумя вариантами. Это легко проверяется на практике, если разместить один флаг, то после щелчка на нем левой кнопкой мыши, он становится выбранным, щелкаем еще раз — выбор отменяется. Когда мы размещаем один переключатель, то после

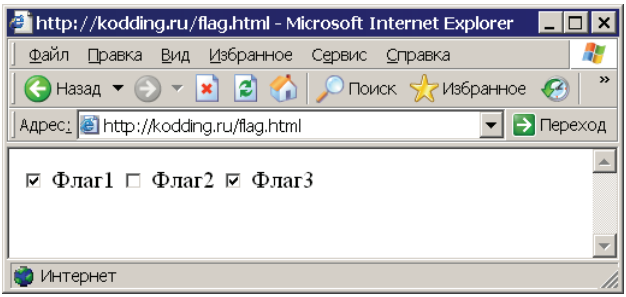


Рис. 7.18. Три флага

того как выполняется щелчок на нем, он становится выбранным, правда последующий щелчок уже ничего не изменит, он так и останется выбранным.

7.6.4. Список

Данный элемент существует в трех вариациях:

- в виде выпадающего списка (рис. 7.19);
- в виде списка с фиксированным числом видимых элементов, когда элементов больше заданного числа, становится доступным вертикальный скроллинг (рис. 7.20);
- в виде списка с фиксированным числом видимых элементов и возможностью одновременного выбора нескольких элементов (с помощью удержания клавиши <Ctrl>), когда элементов в списке больше заданного числа, становится доступным вертикальный скроллинг (рис. 7.21).

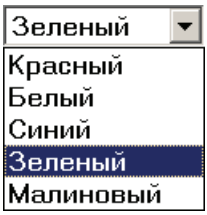


Рис. 7.19. Выпадающий список

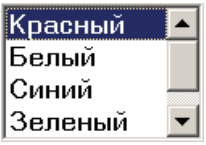


Рис. 7.20. Список с фиксированным числом видимых элементов

Рис. 7.21. Выпадающий список с фиксированным числом видимых элементов и возможностью множественного выбора

Все три вида списка объявляются с помощью одного тега `<select>`, какой вариант будет доступен, зависит от указанных вами параметров. Схема создания списка:

```
<select
      size="количество видимых элементов"
      name="имя списка"
      [multiple]>

          <option value="значение1" [selected]>Отображаемая
                                              строка 1</option>
          ...
          <option value="значение N" [selected]>Отображаемая
                                               строка N</option>
</select>
```

В теге <select> задаются параметры списка, далее указываются его элементы, каждый из которых объявляется парным тегом <option> и </option>. После того как все элементы списка объявлены, ставится закрывающий тег </select>.

Параметр size задает количество отображаемых в списке строк, если он равен 1, то получится первый вариант списка — выпадающий, при значениях больше единицы получится список с фиксированным числом видимых элементов. Если будет задан необязательный параметр multiple, то получится список с возможностью множественного выбора, правда если задать этот параметр при создании выпадающего списка, то ничего не изменится, т. к. выпадающий список изначально подразумевает выбор одного элемента.

Если при объявлении элемента с помощью тега <option> в нем указывается необязательный параметр selected, то это будет означать, что данный элемент будет выбранным. Соответственно selected можно использовать несколько раз в случае создания списка множественного выбора.

Рассмотрим пример. Создайте HTML-документ 3variants_of_list.html, который будет выводить на форме все три вида списка (выпадающий, с фиксированным числом видимых элементов, а также и список множественного выбора), и кнопку (листинг 7.3).

Листинг 7.3. Файл 3variants_of_list.html

```
<form name="form_list" action="list_obrabotka.php" method="get">

<!--Создание списка 1 вида -->
Выберите вид транспорта
<BR>
<select size="1" name="spisok1">
  <option value="value1">          Красный</option>
  <option value="value2">          Белый</option>
  <option value="value3">          Синий</option>
  <option value="value4" selected>Зеленый</option>
  <option value="value5">          Малиновый</option>
</select>

<BR>
<BR>
<!--Создание списка 2 вида -->
Выберите вид транспорта
<BR>
<select size="4" name=" spisok2">
```

```
  <option value="value1">          Красный</option>
  <option value="value2" selected>Белый</option>
  <option value="value3">          Синий</option>
  <option value="value4">          Зеленый</option>
  <option value="value5">          Малиновый</option>
</select>

<BR>
<BR>
<!--Создание списка 3 вида -->
Выберите вид транспорта
<BR>
<select size="4" name=" spisok3" multiple>
  <option value="value1" selected>Автомобиль</option>
  <option value="value2">          Самолет</option>
  <option value="value3" selected>Вертолет</option>
  <option value="value4">          Катер</option>
  <option value="value5">          Подводная лодка</option>
</select>

<br><br>
<input type="submit" name='ff' value="OK">
</form>
```

На рис. 7.22 можно увидеть результат открытия файла 3variants_of_list.html в браузере.

Если нажать кнопку **OK**, то на сервер уйдут следующие данные:

- для выпадающего списка будет передано `имя=value`, где `имя` — это имя самого списка, заданное с помощью параметра `name` в теге `<select>`, а `значение` — значение выбранного элемента, которое задается с помощью параметра `value` в теге `<option>` (для первого списка с рис. 7.22 на сервер будет отправлено `spisok1=value4`);
- для списка с фиксированным числом видимых элементов будет все так же, как и для выпадающего списка (для второго списка с рис. 7.22 на сервер будет отправлено `spisok2=value2`);
- для каждого выбранного значения списка множественного выбора будет сформирована строка вида `name=value`, причем имя будет у всех одинаковое, т. к. параметр `name` задается один раз в теге `<select>`, а вот значение будет разное, т. к. оно задается в теге `<option>` и для каждого элемента разное (для третьего списка с рис. 7.22 на сервер будет отправлено `spisok3=value1&spisok3=value3`).

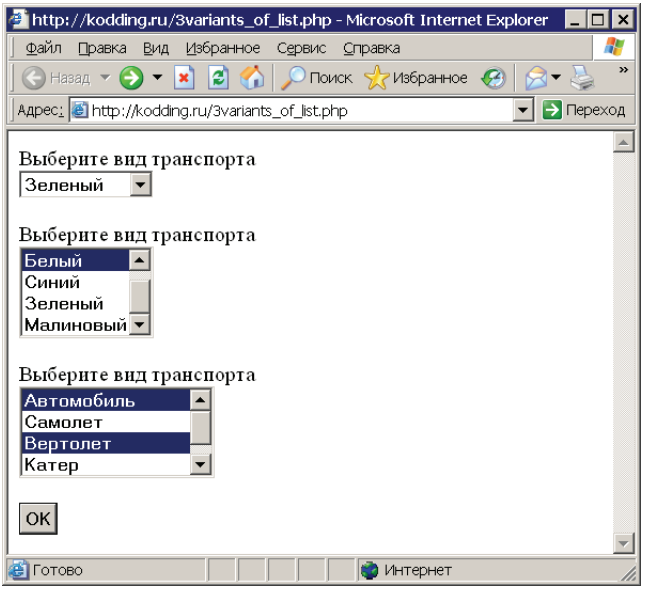


Рис. 7.22. Три варианта списка: выпадающий, с фиксированным числом видимых элементов, список множественного выбора

7.6.5. Поле ввода многострочного текста (TextArea)

Объявляется следующим образом:

```
<textarea
     name="имя элемента"
     rows= высота элемента
     cols= длина элемента
>
     ТЕКСТ, КОТОРЫЙ БУДЕТ СОДЕРЖАТЬ ЭЛЕМЕНТ
</textarea>
```

Описание параметров:

- `name` — имя элемента;
- `rows` — высота элемента, задается числом;
- `cols` — длина элемента, задается числом.

Как видите, элемент `TextArea` задается тегами `<textarea>` и `</textarea>`, между ними размещается текст, который будет отображаться внутри данного элемента. Все параметры, кроме `name`, являются не обязательными, т. е. можно записать:

```
<textarea>
     ТЕКСТ, КОТОРЫЙ БУДЕТ СОДЕРЖАТЬ ЭЛЕМЕНТ
</textarea>
```

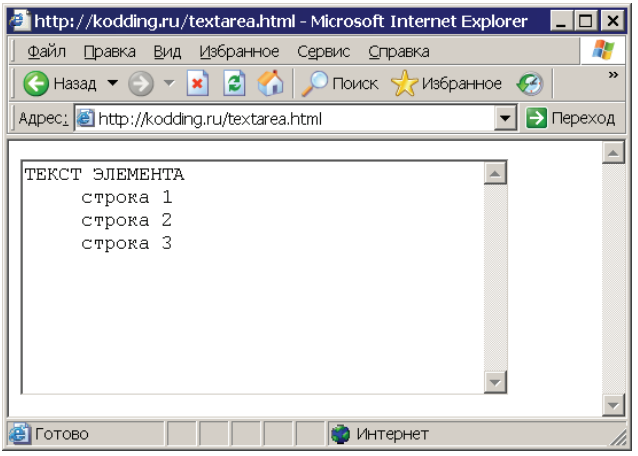


Рис. 7.23. Поле ввода многостраничного текста

Следующий код создаст многострочное поле длиной 40, высотой 10 и именем `new_element` (рис. 7.23).

```
<textarea
      name="new_element"
      rows=10
      cols=40>ТЕКСТ ЭЛЕМЕНТА
      строка 1
      строка 2
      строка 3
</textarea>
```

Данные элемента `TextArea` передаются на сервер в следующем виде `name=текст`.

При работе с полем ввода многострочного текста лучше использовать метод `POST`.

7.6.6. Скрытое поле

Данный элемент нельзя увидеть на экране, поэтому он и называется скрытым, его назначение в передаче данных, которые не должны изменяться пользователем. Например, этот элемент часто используют при реализации скрипта по загрузке файлов на сервер, в этом случае создается скрытое поле, которое содержит максимально возможный размер файла для загрузки, таким образом можно проверить соответствие условию еще перед тем, как начнется процесс загрузки файла. Скрытое поле создается следующим образом:

```
<input
      name="имя"
      type="hidden"
      value="значение"
```

Оно создается так же, как и поле ввода, только в качестве значения параметра `type` указывается `hidden`. Для этого элемента серверу отправятся данные вида `name=value`.

Использование такого метода для хранения параметров, которые не должны изменяться пользователем, очень противоречиво и подходит лишь для тех значений, которые не являются критичными, в том плане, что их изменение не может повлиять на работоспособность скрипта. Дело в том, что хоть элемент и скрытый, его все равно можно изменить. Рассмотрим следующий пример:

```
<FORM name="FormName" action="" method="get">
      <input name="Name" type="hidden"
value="111">
      <input type="submit" value="Send">
<FORM>
```

Запись `action=""` равнозначна тому случаю, когда вообще не указывается параметр `action`. Это означает, что форма будет отправляться той же самой программе, в которой она описана.

Во-первых, если вы отправляете форму методом `GET`, то данные скрытого поля можно увидеть в строке запроса в виде `имя=значение`. Ничего вам не помешает сформировать точно такую же строку вручную, только подправив значение скрытого параметра, и затем отправить ее на сервер. А если использовать следующее:

```
<FORM name="FormName" action="joker.php" method="post">
      <input name="Name" type="hidden" value="111">
      <input type="submit" value="Send">
<FORM>
```

мы не увидим значение скрытого элемента в браузере. Но его все равно можно изменить, для этого достаточно выбрать в браузере пункт меню **Вид | Просмотр HTML-кода** (имеется в виду браузер Internet Explorer, для других браузеров имя пункта может незначительно отличаться), рис. 7.24.

Здесь можно запросто исправить значение скрытого поля на то, которое нам нужно, например на 222. Далее сохранить этот файл на своем жестком диске, после этого запустить его посредством браузера, и нажатием кнопки **Send** отправить форму на сервер, только значение скрытого поля будет уже другим.

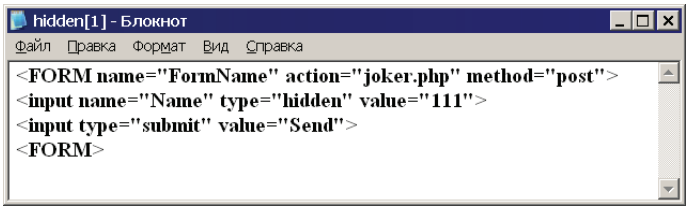


Рис. 7.24. Подсматриваем значение скрытого элемента

Глава 8
Что скрывает браузер

Как вы знаете, строка ввода адреса в браузере не единственный источник обмена информацией с сервером. Чтобы увидеть всю отсылаемую информацию, для браузера Internet Explorer нужно установить программу ieHTTPHeaders, которая является абсолютно бесплатной. Ее можно скачать по следующему адресу (размер файла 132 Кбайт):

http://www.blunck.info/ieHTTPHeadersSetup.exe.

После того как закачаете файл дистрибутива, запустите его, вы увидите следующее окно (рис. 8.1).

Процесс установки элементарен. Вам необходимо лишь в каждом диалоговом окне соглашаться с предложенным. После того как установка будет завершена, запустите Internet Explorer и выберите пункт меню **Вид | Панели обозревателя | ieHTTPHeaders v1.6** (рис. 8.2).

Теперь зайдите на **www.rambler.ru**, вы увидите много интересного (рис. 8.3).

Если вы используете браузер Mozilla, то, чтобы увидеть скрытую информацию, можно скачать программу, расположенную по адресу **http://livehttpheaders.mozdev.org/**, ее размер составляет 70 Кбайт.

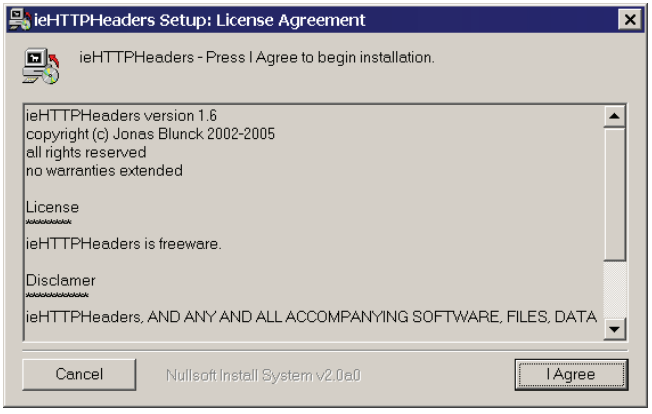


Рис. 8.1. Окно установки программы ieHTTPHeaders

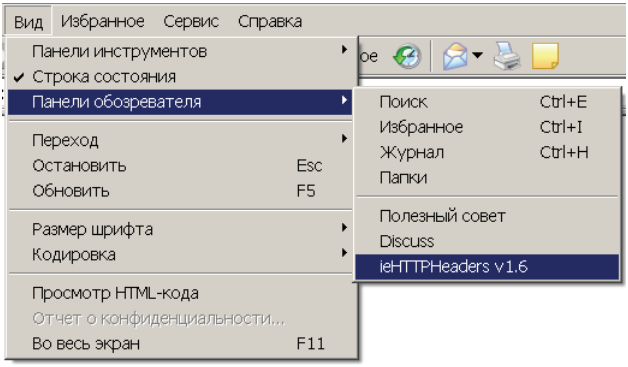


Рис. 8.2. Пункт меню, выбор которого позволит просматривать скрытую информацию, отсылаемую браузером серверу

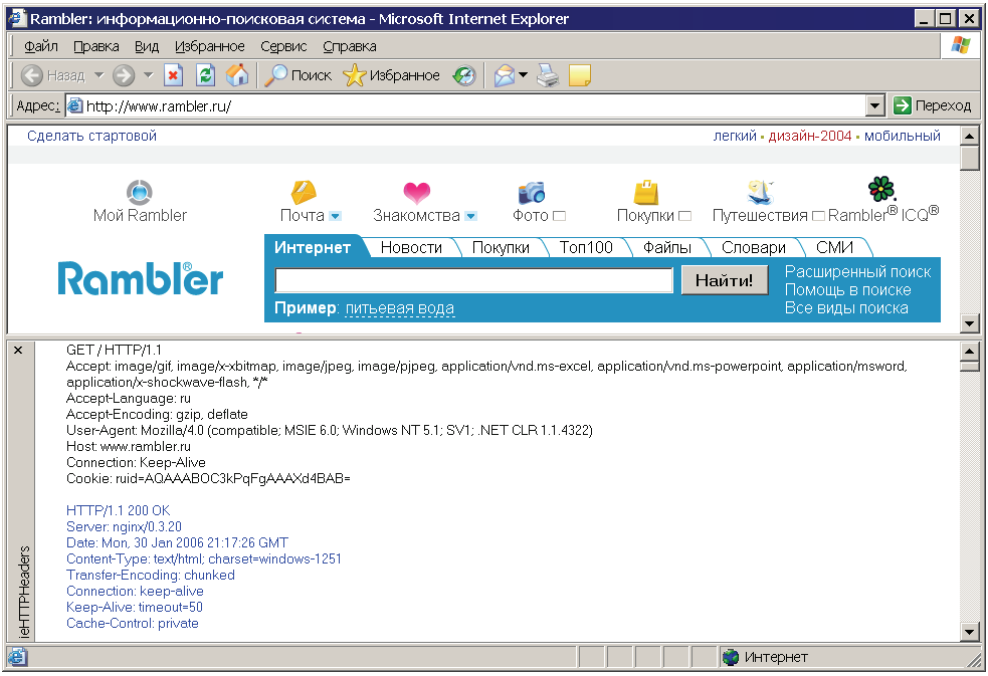


Рис. 8.3. Скрытая информация, которой обмениваются сервер и браузер

Можно также воспользоваться альтернативным решением в виде Web-сайта, который делает то же самое — показывает информацию, которой обмениваются браузер и сервер. Например, сайт **web-sniffer.net**, работа с которым будет рассмотрена далее (рис. 8.4).

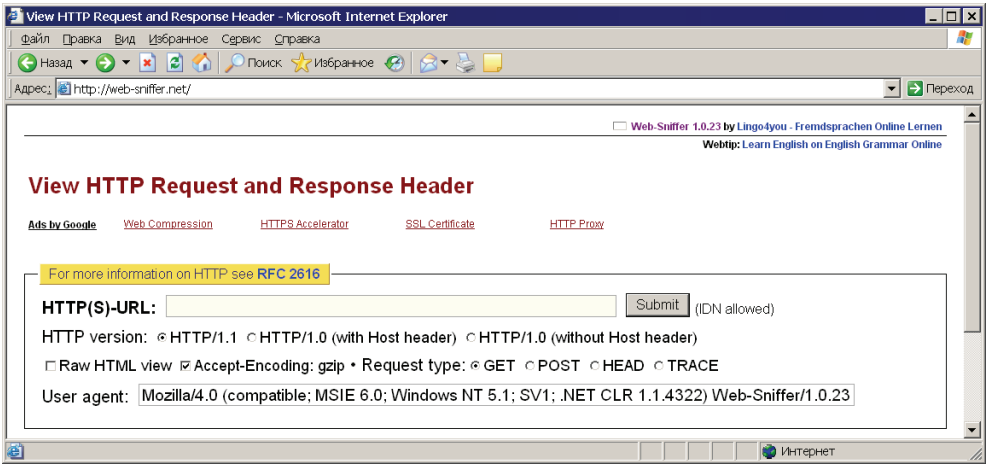


Рис. 8.4. Сайт **www.web-sniffer.net**

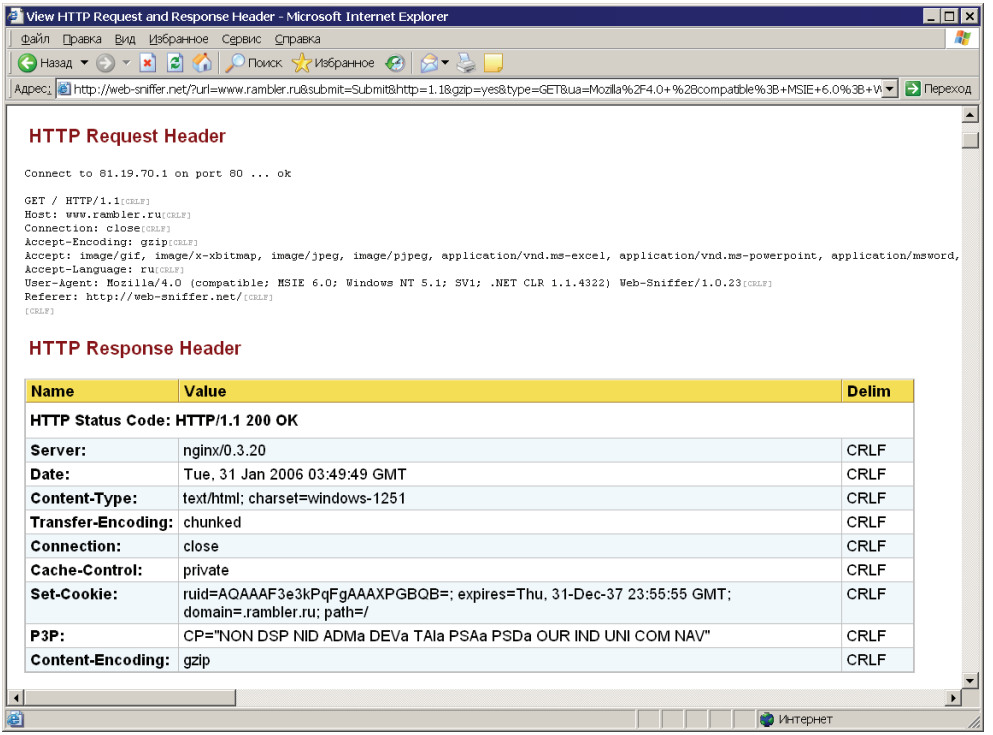


Рис. 8.5. Просмотр скрытой информации с помощью сайта **www.web-sniffer.net**

В поле **HTTP(S)-URL** введите **www.rambler.ru** и нажмите кнопку **Submit**, после чего вы увидите следующий результат — рис. 8.5.

Браузер и сервер общаются, посылая друг другу запросы. Все эти запросы стандартизированы, это значит, что точно определены возможные варианты запросов и ответы на них. Тут можно провести аналогию с реальным миром на примере официальной встречи двух президентов — такие встречи заранее строго расписаны: начиная от того, как главы государств будут проходить по ковровой дорожке, и заканчивая тем, какие вопросы они будут обсуждать. Применительно к общению браузера и сервера все то же самое, в их способ обмена информацией положен набор правил, который называется *HTTP-протоколом*. Сам процесс обмена информацией называется *обменом HTTP-заголовками*.

Замечание

Если вас по какой-то причине не устраивают предложенные в данной главе средства просмотра скрытой информации, то вы можете найти альтернативные им, просто введя в любом поисковом сервере фразу `HTTP sniffer`.

Вернемся к заголовкам, которые были отправлены браузером при обращении на **www.rambler.ru** (рис. 8.3 и 8.5). Первой строкой является `GET / HTTP/1.1`, это означает, что используется `GET`-запрос (он применяется всегда, когда происходит запрос к сайту), и браузер сообщает серверу, что он будет работать с ним по протоколу HTTP версии 1.1 (на сегодняшний день она последняя).

Заголовок `Host: www.rambler.ru` указывает сайт, к которому мы обращаемся. В заголовке `Accept` брау-зер перечисляет данные, с которыми он может работать (например, `image/gif` означает, что браузер может отображать картинки в формате GIF).

Замечание

В данной главе основной упор сделан на рассмотрение технологии общения браузера и сервера между собой. Если вы захотите узнать более подробно о протоколе HTTP, посетите следующие сайты: ***http://www.w3.org/Protocols/rfc2616/rfc2616.html*** *— спецификация (или описание) протокола HTTP 1.1 на английском языке (см. на сайте раздел 14);* ***http://seolab.ru/add/rfc2068/*** *— спецификация протокола HTTP 1.1 на русском языке (см. на сайте разделы 4.3, 5.2 и 13).*

После обращения браузера к серверу последний пришлет ему следующий ответ: `HTTP/1.1 200 OK`, который означает, что запрос был успешно обработан сервером.

Давайте попробуем найти что-нибудь с помощью поискового сервера. Для этого зайдите на любой из них, например **rambler.ru**, введите какое-нибудь слово и нажмите **Поиск**. Результат обмена заголовками в этом случае будет следующим — рис. 8.6.

Как видите, при поиске используется метод `GET`. Также вы можете увидеть параметр `words=php`, в котором записано искомое нами слово. Теперь мы "знаем", какое имя у элемента поля для ввода искомой фразы поискового сервера **Rambler**.

Рис. 8.6. Просмотр отправленных браузером заголовков при поиске слова "PHP" на **rambler.ru**

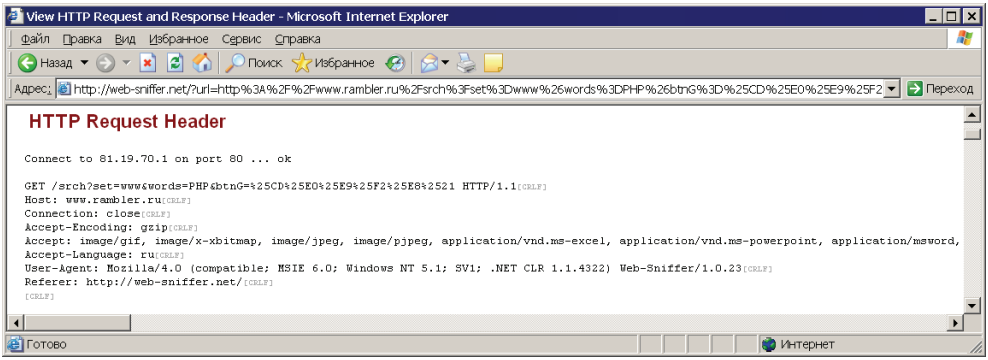


Рис. 8.7. Просмотр отправленных браузером заголовков с помощью сайта **www.web-sniffer.net**

Если вы просматриваете заголовки, используя **www.web-sniffer.net**, то сначала вам все равно придется зайти на поисковый сервер, ввести искомое слово, нажать кнопку **Найти**, затем скопировать строку запроса, которая получится в результате, и вставить ее в поле **HTTP(S)-URL**. Далее нажмите **Submit**, после чего вы увидите следующий результат — рис. 8.7.

Давайте опять зайдем на **rambler.ru** и попробуем войти в почтовый ящик. Сейчас не важно, есть ли он у вас или нет, это нужно только для эксперимента, поэтому можно вводить абсолютно неправильные данные, например, я ввел в качестве имени `Name`, в качестве пароля — `Password` (рис. 8.8).

Нажмите кнопку **Войти**. На рис. 8.9 можно увидеть результат обмена заголовками при попытке входа в почтовый ящик от **rambler.ru**.

В случае использования Internet Explorer и ieHTTPHeaders ответ сервера будет выделен синим цветом, в случае использования сайта ***www.web-sniffer.net*** *он будет располагаться после фразы* ***HTTP Response Header****.*

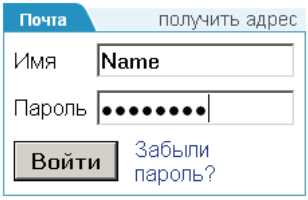


Рис. 8.8. Форма для входа в почтовый ящик от **rambler.ru**

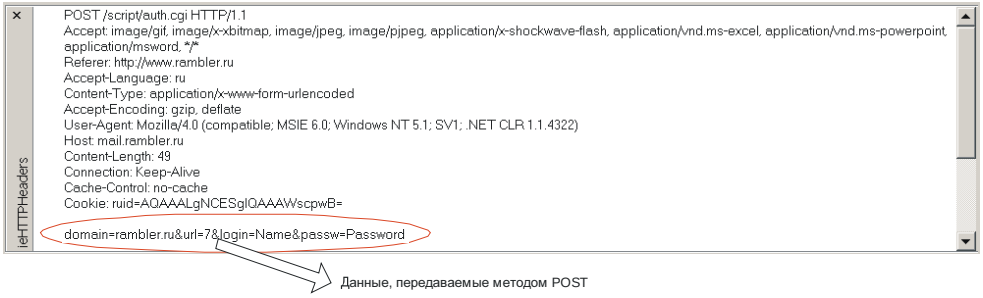


Рис. 8.9. Обмен заголовками браузера с сервером при входе на почтовый ящик от **rambler.ru**

Сразу можно заметить, что используется метод `POST`. Обратите внимание на новый заголовок `Content-Length`, который передает серверу информацию о размере пересылаемых ему данных. После всех заголовков можно увидеть следующую строку:

```
domain=rambler.ru&url=7&login=Name&passw=Pasword
```

Это тело запроса, в котором как раз и были переданы введенные ранее значения `Login` и `Password`. Данная строка состоит из 49 символов (49 байт), это точно столько, сколько было указано в заголовке `Content-Length`.

То есть несмотря на то, что вроде бы при использовании метода `POST` данные и отправляются в скрытом от пользователя виде, это нисколько не защищает их от просмотра, т. к. они достаточно элементарно могут быть переведены из разряда скрытых в открытые.

Глава 9
Сплошная практика

В данной главе будет рассмотрено создание самых распространенных скриптов, необходимость в которых возникает при реализации практически любого сайта. Основной упор при изложении материала сделан на практическую часть разработки PHP-программ, конечно же, будет много новой информации, как обычно, снабженной подробнейшими комментариями. Начнем с простого — с разработки формы обратной связи.

9.1. Форма обратной связи

Форма обратной связи представляет простейший механизм общения посетителя сайта с его администратором. Создайте файл backform.html (листинг 9.1).

Листинг 9.1. Файл backform.html

```
<FORM action="obrabotka.php" method="POST">

Ваше имя:<br>
<input type="text" name="person">
<br>
Ваш e-mail:<br>
<input type="text" name="email">
<br>
Вопрос администратору:<br>
<textarea name="question" cols=40 rows=5></textarea>
<br>
<input type="submit" name="okbutton" value="OK">

</FORM>
```

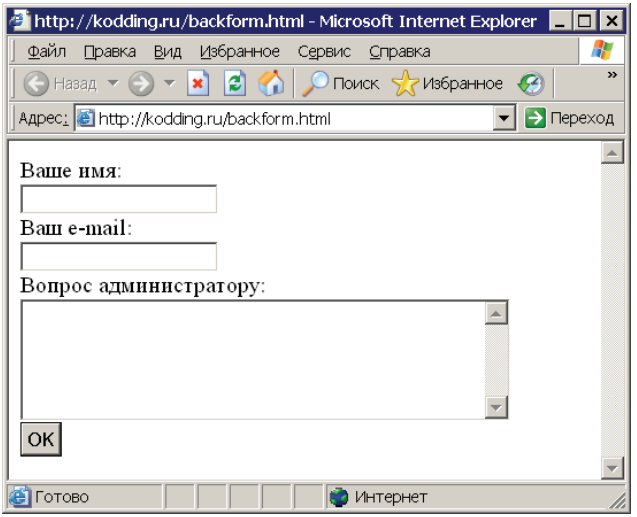


Рис. 9.1. Форма обратной связи

Как видите, форма состоит из четырех элементов (рис. 9.1):

- поля для ввода имени пользователя;
- поля для ввода e-mail пользователя;
- поля для ввода многострочного текста — сюда пользователь будет писать свой вопрос к администратору сайта;
- кнопки, после нажатия которой форма будет отправлена на сервер.

Теперь создайте скрипт obrabotka.php, который будет обрабатывать разработанную ранее форму (листинг 9.2).

Листинг 9.2. Файл obrabotka.php

```
<?php
//Проверяем, был ли скрипт вызван как обработчик формы
if (isset($_POST['okbutton']))
{
    //Куда будет отправлено письмо
    $komu="LittleBuddan@vr-online.ru";
    //Тема письма
    $tema="Вопрос от ".$_POST['person']." ".$_POST['email'];
    //Само письмо
    $text_pisma=$_POST['question'];

    //Отправляем письмо
```

```
13      mail($komu,$tema,$text_pisma);
14
15      echo "<br>Ваш вопрос был отправлен администратору";
16      echo "<br><a href=backform.html>Назад</a>";
17  }
18  ?>
```

В строке **3** используется новая конструкция, она называется *условной* и предназначена для проверки некоторого условия (логического выражения), при совпадении которого выполняется код, заключенный в фигурные скобки. Синтаксис условной конструкции:


```
If (условие)
{ код, который будет выполнен при совпадении условия }
```

Стоит отметить, что условие обязательно должно быть заключено в круглые скобки. В программе obrabotka.php было использовано следующее условие: `(isset($_POST['okbutton']))`. Таким образом проверяется наличие элемента с именем `okbutton` в массиве `$_POST` с помощью уже знакомой функции `isset()` — если он существует, то условие будет соблюдено (равно `TRUE`), в противном случае условие выполнено не будет (равно `FALSE`).

Зачем нужна такая проверка? Просто ничто не мешает кому-то запустить скрипт obrabotka.php напрямую — не важно, по ошибке или по злому умыслу. Тогда программа будет работать неправильно, т. к. у нее не будет необходимых данных (ни имени пользователя, ни его e-mail, ни текста письма), поэтому добавление условия позволяет добиться результата, при котором скрипт будет выполнен только в случае, когда пользователь заполнил форму и нажал кнопку **OK** — именно тогда в массиве `$_POST` будет создан элемент `okbutton` (рис. 9.2).

В строке **13** происходит отправка электронного письма с данными, которые ввел пользователь. Для этого используется стандартная функция `mail()`, ее синтаксис выглядит следующим образом:

```
mail(адрес получателя, тема, сообщение);
```

Если отправка письма успешна, то функция вернет `TRUE`, в противном случае — `FALSE`. Описание ее параметров:

- `адрес получателя` — куда будет отправлено письмо, в программе мы задаем его в переменной `$komu`;
- `тема` — тема письма, мы формируем его в переменной `$tema` как объединение фразы "Вопрос от", имени пользователя и его электронного адреса;
- `сообщение` — тело письма, мы формируем его в переменной `$text_pisma` равным значению элемента `question` формы.

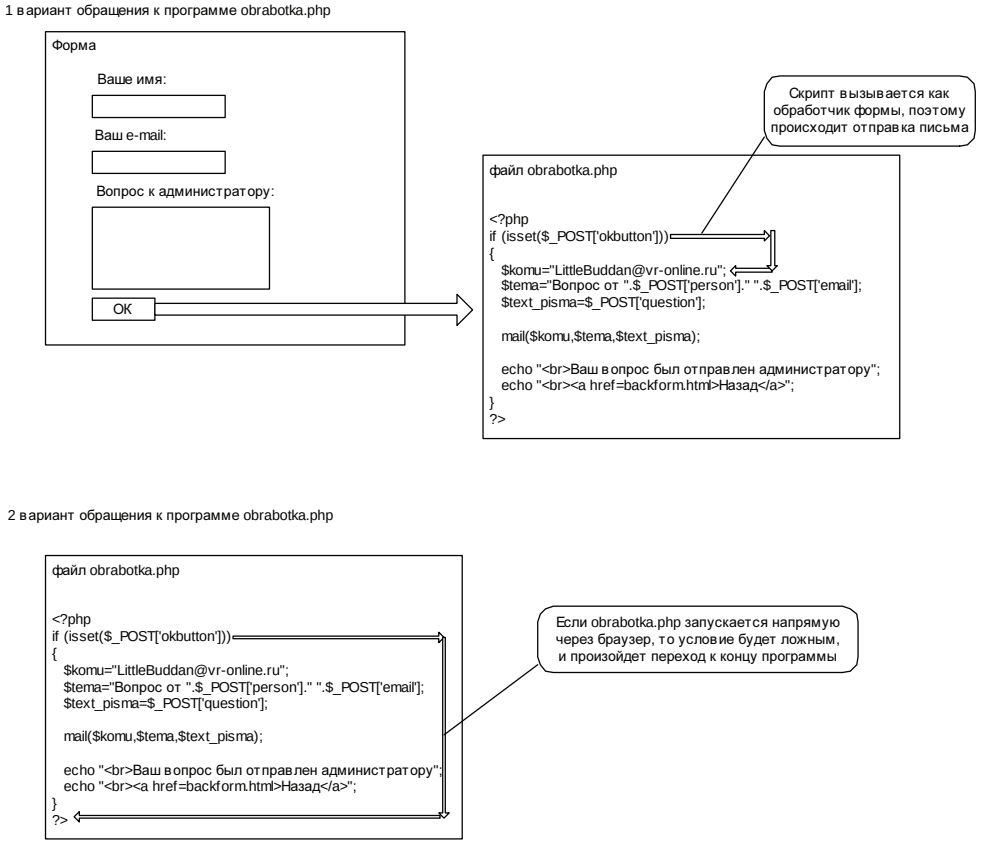


Рис. 9.2. Схема работы условной конструкции на примере программы obrabotka.php

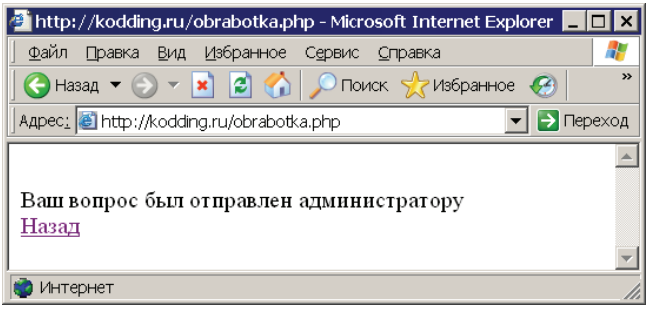


Рис. 9.3. Результат заполнения и отправки на сервер формы обратной связи

Запустите HTML-документ backform.php, в котором хранится форма обратной связи, введите данные и нажмите кнопку **OK**. Вы увидите результат, изображенный на рис. 9.3.

Когда разрабатываются небольшие по объему и несложные в реализации скрипты, то часто используют подход, при котором форма и ее обработчик хранятся в одном файле. В листинге 9.3 представлен код программы, реализующей тот же самый механизм обратной связи, рассмотренный ранее, только теперь он реализован с помощью одного файла, сохраните его под именем backform.php.

Листинг 9.3. Файл backform.php

```
<?php
//Если была нажата кнопка OK,
//то отправляем письмо и информируем об этом пользователя
if (isset($_POST['okbutton']))
{
   if ($_POST['person']=='' or $_POST['email']=='' or
$_POST['question']=='')
   {
      echo "<font color='red'>Вы ввели не все данные</font>";
      echo "<br><a href=backform.php>Назад</a>";
      exit;
   }

   //Куда будем отправлять письмо
   $komu="LittleBuddan@vr-online.ru";
   //Тема письма
   $tema="Вопрос от ".$_POST['person']." ".$_POST['email'];
   //Само письмо
   $text_pisma=$_POST['question'];

   //Отправляем письмо
   mail($komu,$tema,$text_pisma);

   echo "Ваш вопрос был отправлен администратору";
   echo "<br><a href=backform.php>Назад</a>";
   //Выполнять больше нечего, выходим из программы
   exit;
}
```

```
28 ?>
29
30 <FORM action="" method="post">
31 Ваше имя:<br>
32 <input type="text" name="person">
33 <br>
34 Ваш e-mail:<br>
35 <input type="text" name="email">
36 <br>
37 Вопрос администратору:<br>
38 <textarea name="question" cols=40 rows=5></textarea>
39 <br>
40 <input type="submit" name="okbutton" value="OK">
41 </FORM>
```

Образно говоря, теперь половина файла отвечает за обработку формы, половина — за ее вывод. В строке **4** осуществляется проверка, был ли скрипт вызван как обработчик `POST`-формы. Если нет (значит, условие `(isset($_POST['okbutton']))` ложно), то будет выполнен фрагмент кода, начинающийся со строки **30**, который отвечает за вывод пользователю формы обратной связи. Иначе, следующей будет выполнена строка **6**, где проверяется еще одно условие — все ли поля формы были заполнены пользователем, перед тем как он нажал **OK**.

Одно из главных правил при работе с формами — это проверка данных, которые ввел пользователь, и в первую очередь факта, ввел ли он их вообще. Делается это простой проверкой значений тех элементов формы, с данными которых будет осуществляться работа в программе. Реализуется это очень легко, т. к. независимо от того, какой метод передачи используется (`POST` или `GET`), для каждого элемента формы, при создании которого был указан параметр `name`, на сервер придет строка в виде `имя=значение`. Все что нам остается после этого — проверить, не содержит ли нужный нам элемент пустое значение, что мы и делаем с помощью следующего логического выражения:

```
($_POST['person']=="" or $_POST['email']=="" or $_POST['question']=="")
```

То есть условие в строке **6** будет верно, если поле `person` или поле `email` или поле для ввода многострочного текста `question` имеют пустое значение. В этом случае высветится предупреждение `"Вы ввели не все данные"` и работа программы завершится (рис. 9.4).

Таким образом в программу встроена защита от отправки пустых сообщений, а заодно создан более дружественный интерфейс, который выводит для пользователя поясняющее сообщение.

Особого внимания заслуживает строка **8**:

```
echo "<font color='red'>Вы ввели не все данные</font>";
```

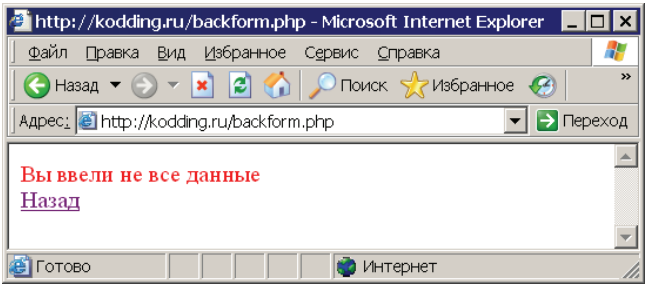


Рис. 9.4. Предупреждение, возникающее вследствие наличия на форме незаполненных данными элементов

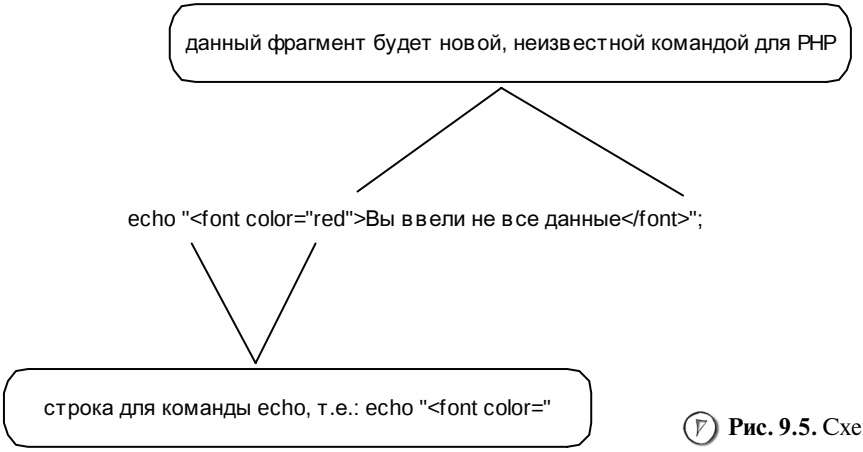


Рис. 9.5. Схема обработки препроцессором гипертекста фрагмента кода с неправильным использованием кавычек

Обратите внимание, в ней используется два вида кавычек. Это очень важный момент, т. к. если его не соблюсти, программа работать не будет. Первые двойные кавычки необходимы для строки, которая выводится с помощью команды `echo`, вторые — одинарные, для HTML-тега, находящегося внутри строки. Рассмотрим случай, при котором использовался только один вариант кавычек — двойных, при этом строка **8** приобрела бы следующий вид:

```
echo "<font color="red">Вы ввели не все данные</font>";
```

и была бы обработана препроцессором гипертекста по следующей схеме — рис. 9.5.

То есть выполнение программы backform.php закончилось бы ошибкой из-за незнакомой для PHP команды `ed">Вы ввели не все данные</font>"`.

В Денвере в качестве функции `mail()` используется отладочная заглушка, которая не отправляет письма, а складывает их в директорию tmp/!sendmail (рис. 9.6).

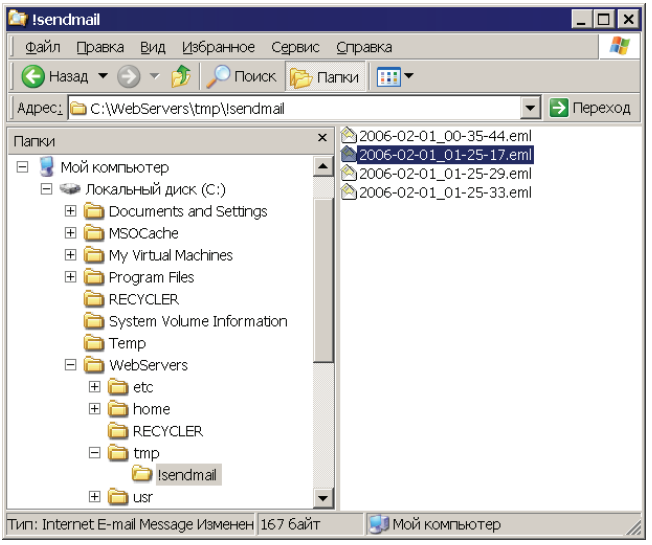


Рис. 9.6. Хранилище писем Денвера, отправленных с помощью функции `mail()`

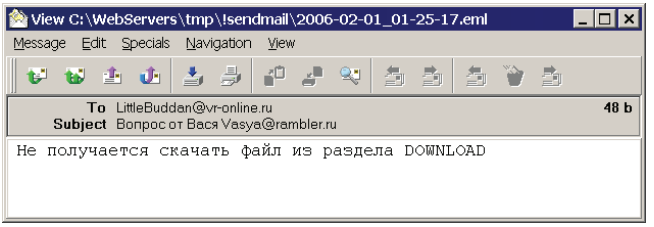


Рис. 9.7. Электронное письмо, сформированное функцией `mail()` и открытое в почтовом клиенте The Bat

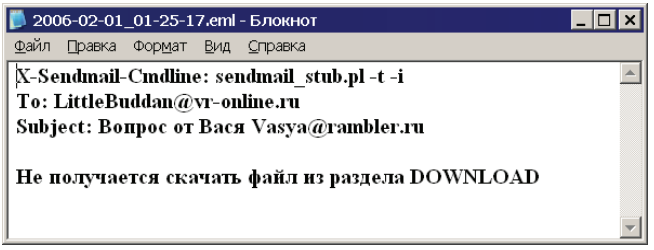


Рис. 9.8. Электронное письмо, сформированное функцией `mail()` и открытое с помощью Блокнота

Замечание

Параметры, влияющие на работу функции `mail()`, рассмотрены в приложении 1.

Вы можете открыть любое такое письмо с помощью двойного щелчка мыши, конечно, если у вас установлен почтовый клиент, например, Outlook, The Bat или другой (рис. 9.7).

Данное письмо можно также просмотреть в обычном Блокноте (рис. 9.8).

9.2. Гостевая книга

Конечно, гостевые книги уже не так популярны, как раньше — сейчас многие для этой цели просто создают специальный раздел на форуме. В то же время гостевая книга остается одним из самых простых и удобных способов оставить свое мнение или предложение на сайте. К тому же разработка гостевой книги интересна в плане изучения на практике нового материала, который вам понадобится в дальнейшем при реализации многих проектов. Трансформированный вариант гостевой книги очень часто можно увидеть на различных новостных лентах, он используется как средство для комментариев, которые оставляют люди, неравнодушные к тому или иному событию (рис. 9.9).

VR-online:\Новости

Процедура Разработка клиент-серверных приложений в Delphi на books.ru (VR-online:ТКатегория);
параметры
LittleBudda : **ТАвтор**;
2005-11-21 12:48:43 : **ТДата**;
Email : **ТИсточник**;
начало

Разработка клиент-серверных приложений в Delphi уже появилась на Books.ru. Заказываем ЗДЕСЬ.

ShellExecute(Печать);

конец;

// Коментарий № 1 (Автор: MaXL; Дата: 2005-11-21 14:35:44)

А содержания появиться или нет, и желательно как можно поскорее плиз, ну или вообще напиши что вней?

// Коментарий № 2 (Автор: LittleBudda; Дата: 2005-11-21 19:55:32)

Содержание будет в ближайшее время!

Рис. 9.9. Трансформированный вариант гостевой, который можно увидеть на сайте **www.vr-online.ru**

9.2.1. Приступаем к работе

Для начала создайте виртуальный хост **guestbook.ru**, в нем папку www, далее перезапустите Денвер, потому что только после этого новый хост станет доступен. Все файлы, разработанные в данном разделе, необходимо сохранять в guestbook.ru\www. Когда пользователь обратится к гостевой книге, ему будет предложена форма, состоящая из трех элементов:

- поля для ввода имени;
- поля для ввода текста;
- кнопки подтверждения.

Ниже формы будут располагаться сообщения, оставленные ранее, каждое из них будет выводиться следующим образом:

- дата сообщения;
- имя человека, который его оставил;
- текст сообщения.

На рис. 9.10 представлена схема внешнего вида гостевой книги (или, другими словами, ее интерфейса).

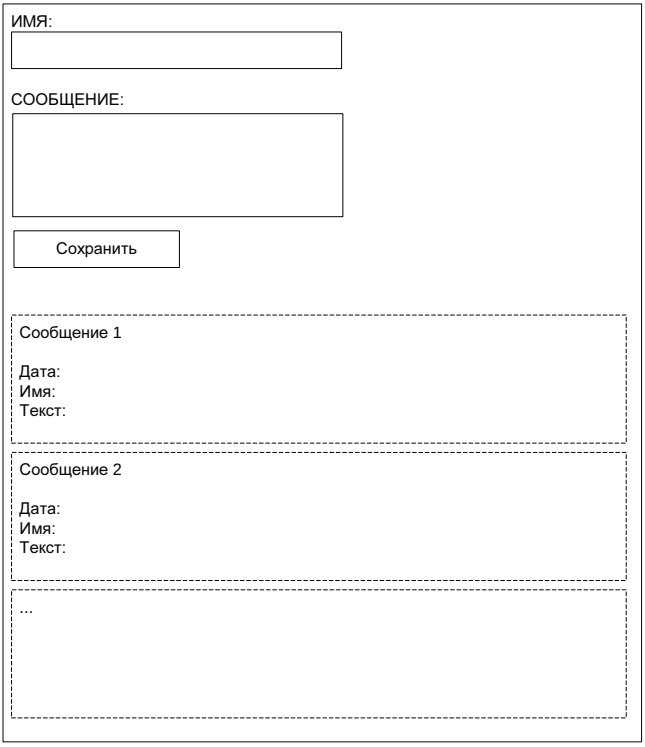


Рис. 9.10. Внешний вид будущей гостевой книги

Все сообщения будут храниться в одном текстовом файле. Каждое из них будет состоять из 5 строк:

- 1 строка представляет собой "----------", это разделитель, который устанавливается в начале каждого сообщения, он не носит информативного характера, а служит лишь для удобства. Например, в случае если нам нужно будет внести изменения вручную, разделитель поможет визуально отличить одно сообщение от другого;
- 2 строка предназначена для хранения IP-адреса посетителя, который оставил сообщение, эта информация не будет видна посетителям, потому что она предназначена для администратора;
- 3 строка — дата размещения сообщения, будет добавляться автоматически в момент сохранения пользователем сообщения;
- 4 строка — имя пользователя (то, что ввел пользователь в поле **Имя**);
- 5 строка — текст сообщения (то, что ввел пользователь в поле **Текст**).

Гостевая книга будет состоять из двух скриптов:

- guestbook.php — осуществляет вывод формы, изображенной на рис. 9.10, и всех сообщений, хранящихся в гостевой книге;
- add_message.php — обработчик формы, который осуществляет добавление нового сообщения в гостевую книгу.

После того как точно известно, что необходимо разработать, можно приступать к кодингу (от англ. *coding* — написание кода). В листинге 9.4 показан файл guestbook.php, который вам необходимо создать.

Листинг 9.4. Файл guestbook.php

```
<html>
<head>
  <title>Гостевая</title>
</head>
<body>

<!--Создаем форму, состоящую из поля, textarea и кнопки-->
<form action="add_message.php " method="POST">
Имя:<BR>
<input type="text" name="name_of_guest">
<BR>
Мысли:<BR>
<textarea name="message_of_guest" cols=40 rows=5></textarea>
<BR>
<input type="submit" name="okbutton" value="OK">
```

```
16 </form>
17
18 <?php
19 //Открываем файл gost.txt в режиме чтения
20 $f=fopen("gost.txt","rt") or die("Не могу открыть файл");
21 //Пока не конец файла сообщений
22 while (!feof($f))
23 {
24 //Читаем очередное сообщение
25         //Получаем строку "-----"
26         $hide_line=fgets($f);
27         //Получаем IP-адрес
28         $ip_=fgets($f);
29         //Читаем дату размещения сообщения
30         $data=fgets($f);
31                 //Выводим дату размещения сообщения
32                 echo "<small>Дата: </small>".$data."<br>";
33         //Читаем имя человека, который его написал
34         $data=fgets($f);
35                 //Выводим имя
36                 echo "<small>Имя: </small>".$data."<br>";
37         //Читаем текст сообщения
38         $data=fgets($f);
39                 //Выводим текст сообщения
40                 echo "<small>Сообщение: </small>".$data."<br>";
41         //Отделяем сообщение горизонтальной линией
42         echo "<HR>";
43 }
44 //Закрываем соединение с файлом
45 fclose($f);
46 ?>
```

Рассмотрим код написанной программы. В строках с **1** по **16** все должно быть понятно, т. к. там используются стандартные HTML-теги и происходит создание формы с полем ввода под названием `name_of_guest`, полем для ввода многострочного текста с именем `message_of_guest` и кнопкой добавления сообщения с именем `okbutton`. Данные формы будут передаваться методом `POST` и обрабатываться программой add_message.php.

В строке **20** используется уже знакомая вам функция `fopen()` для открытия файла message.txt, в котором хранятся сообщения пользователей (данного файла пока не существует, но чуть позже мы его создадим). Файл будет открыт как текстовый в режиме `r`, то есть только для чтения. В случае невозможности открыть файл, сработает защитная конструкция `die("Не могу открыть файл")`, принцип работы которой был рассмотрен ранее *(см. разд. 6.1)*.

В строке **22** вы можете увидеть новую конструкцию `while`. Она предназначена для организации цикла. Ее синтаксис выглядит следующим образом:


```
while (условие)
{
        Фрагмент кода
}
```

Назначение данной конструкции — выполнять код, заключенный в фигурные скобки, пока истинно условие.

Цикл, организованный с помощью while, называется циклом с предусловием.

Циклом такая организация выполнения называется, потому что один и тот же код может быть выполнен множество раз. Код, заключенный в фигурные скобки, называется *телом цикла*.

Когда в тексте программы встречается `while`, то обработка данной конструкции происходит следующим образом: сначала проверяется условие, которое обязательно должно быть заключено в круглые скобки, если оно ложно (`FALSE`), то код, заключенный в скобки, не будет выполнен ни разу, если условие верно (`TRUE`), код выполняется один раз, потом произойдет возврат к условию и его проверка еще раз. Если условие окажется ложным (`FALSE`), то тело цикла больше выполняться не будет, а осуществится переход на оператор, расположенный после закрывающей фигурной скобки (строка **44**), если же условие опять верно (`TRUE`), то тело цикла будет выполнено еще раз и далее все повторится. Отсюда следует главное *правило организации цикла*: условие должно быть таким, чтобы не произошло зацикливания программы. Рассмотрим следующий пример:

```
<?php
$a=0;
while ($a<1)
{
    echo "Привет";
}
?>
```

Если вы запустите его, то браузер повиснет, т. к. программа будет выполняться бесконечно по причине того, что условие всегда будет верным. Поэтому часто

в теле цикла размещается код, который будет изменять условие. Рассмотрим следующую программу:

```
<?php
$a=0;
while ($a<5)
{
    echo $a;
    $a=$a+1;
}
?>
```

В этом случае в теле цикла происходит увеличение значения переменной `$a` на единицу. Так как в условии проверяется именно значение переменной `$a`, цикл будет выполнен 5 раз.

Вернемся к программе guestbook.php. Цикл `while` в ней организован следующим образом:

```
Пока (не конец файла)
{
      читаем очередное сообщение и выводим его на экран
}
```

В условии используется новая функция `feof()`, которая определяет, достиг ли указатель текущей позиции конца файла. Если да, то функция возвращает `TRUE`, в противном случае — `FALSE`. Ее синтаксис:

```
feof(файловая переменная);
```

Принцип работы функции очень прост: каждый раз, когда что-то читается из файла, указатель текущей позиции смещается ровно на столько, сколько информации было прочитано, когда вся информация из файла прочитана, указатель будет указывать на конец файла, и в этом случае функция `feof` вернет `TRUE`.

Если мы просто напишем `while (feof($f))`, то уже при самом первом выполнении цикла при наличии хотя бы одного сообщения в gost.txt условие будет равняться `FALSE`. Поэтому используется отрицание "`не`", которое на PHP обозначается восклицательным знаком (`!`). Таким образом, запись `while (!feof($f))` означает "пока не конец файла, выполнять тело цикла". Когда выполнение программы дойдет до этого условия, то сначала будет исполнена функция `feof`, при этом если конец файла не достигнут, то она вернет `FALSE`, но знак `!`, стоящий перед `feof()`, перевернет результат (именно так работает отрицание) и получится, что условие цикла верно (`TRUE`), таким образом, тело цикла будет выполнено. После того как будет достигнут конец файла, функция `feof()` вернет `TRUE`, а отрицание перевернет этот результат на `FALSE`, соответственно работа цикла будет прекращена.

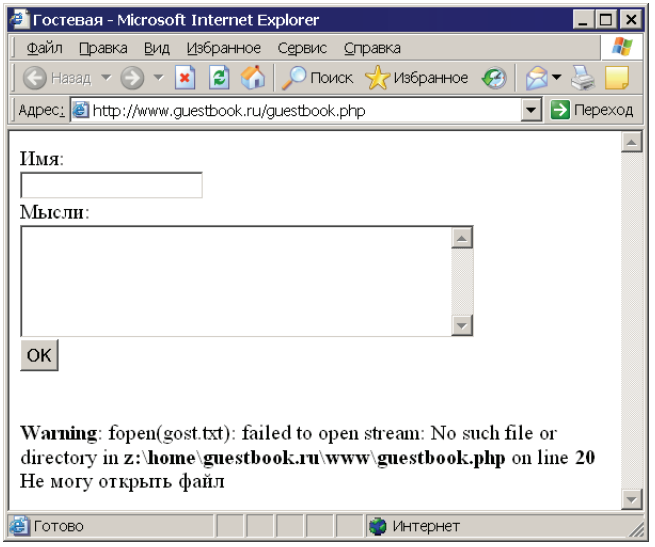


Рис. 9.11. Результат работы гостевой книги в случае отсутствия файла сообщений gost.txt

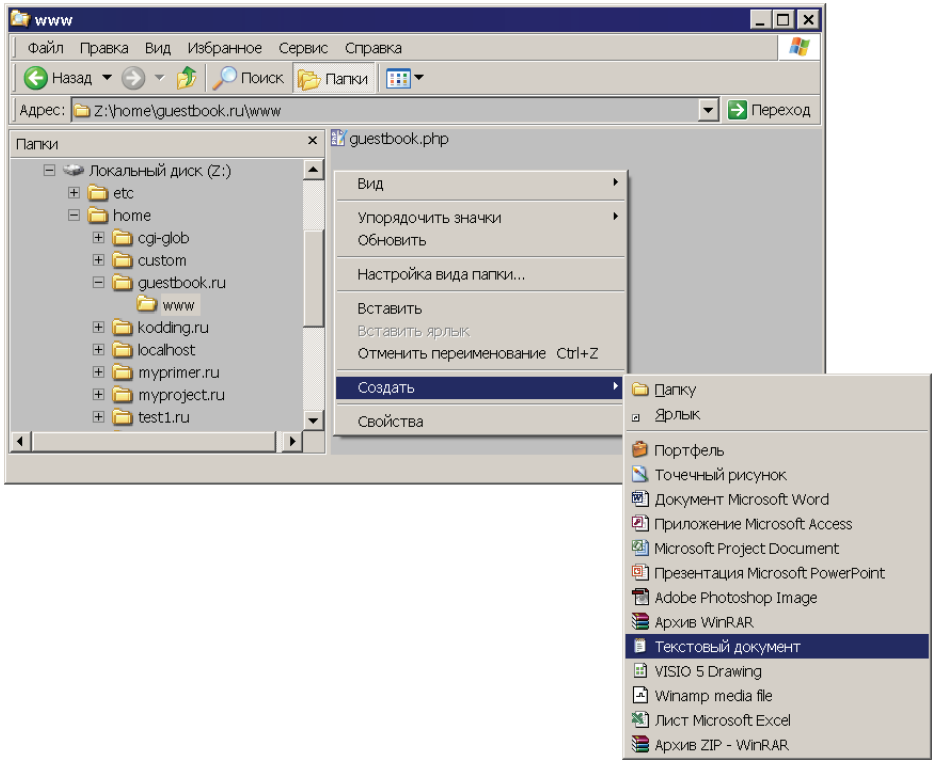


Рис. 9.12. Создание файла gost.txt, предназначенного для хранения сообщений

В теле цикла происходит чтение всех сообщений из файла gost.txt. Так как одно сообщение состоит из 5 строк (разделитель, дата, IP-адрес, имя, текст), то соответственно для вывода одного сообщения нужно прочитать эти 5 строк и вывести 3 из них на экран (разделитель и IP-адрес не выводятся). Каждое сообщение отделяется от предыдущего горизонтальной линией (тег `<HR>`, строка **42**).

Если сейчас запустить скрипт guestbook.php, то будет выведена форма и сообщение об ошибке (рис. 9.11).

Так происходит в силу того, что файла gost.txt пока не существует, поэтому при попытке выполнить функцию `fopen()` (строка **20**) произошла ошибка и сработала защитная конструкция `die()`, которая прекратила выполнение программы и вывела текст **Не могу открыть файл**.

Создайте файл gost.txt в guestbook.ru/www, для этого достаточно воспользоваться проводником, открыть нужный каталог, затем щелкнуть в правой части проводника правой кнопкой мыши и в выпадающем меню выбрать **Создать | Текстовый файл** (рис. 9.12).

После этого измените имя файла на gost.txt и запустите скрипт еще раз. Вы увидите, что сообщение об ошибке исчезло. Теперь создадим скрипт add_message.php, который будет добавлять новое сообщение в гостевую книгу (листинг 9.5).

Листинг 9.5. Файл add_message.php

```
1   <?php
2   //Скрипт будет работать только при условии, что он
3   //был запущен как обработчик формы, размещенной на guestbook.php
4   if (isset($_POST['okbutton']))
5   {
6      if ($_POST['name_of_guest']=='')
7         exit("Введите имя! <a href='guestbook.php'>Назад</a>");
8
9      if ($_POST['message_of_guest']=='')
10     {
11        //А тут более распространенный вариант в два оператора
12*       echo "Введите текст сообщения! <a href='
    guestbook.php'>Назад</a>";
13        exit;
14     }
15
16     //Получаем значения поля Имя
17     $name_of_guest=htmlspecialchars($_POST["name_of_guest"]);
18     //Получаем значения поля Мысли
19     $message_of_guest=htmlspecialchars($_POST["message_of_guest"]);
20
21     if (filesize("gost.txt")>0)
22           $first_message=FALSE;
23      else
24           $first_message=TRUE;
```

```
25
26      //Открываем файл gost.txt для записи,
27      //если файла нет, то он будет создан
28      $f=fopen("gost.txt","at") or die("Не могу открыть файл");
29
30      //Блокируем файл, чтобы никто не мог к нему обратиться,
31      //пока мы с ним не закончим работу
32      flock($f,2);
33
34      //Записываем строку "-----------------"
35      //Если это первое сообщение, то без \n в начале, иначе с ним
36      if ($first_message==TRUE)
37           fputs($f,"--------------------\n");
38       else
39           fputs($f,"\n--------------------\n");
40
41      //Записываем IP-адрес пользователя
42      fputs($f,$_SERVER['REMOTE_ADDR']."\n");
43      //Записываем дату размещения сообщения
44      fputs($f,date('d.m.y')."\n");
45      //Записываем имя
46      fputs($f,$name_of_guest."\n");
47      //Записываем текст сообщения
48      fputs($f,$message_of_guest);
49
50      //снимаем блокировку
51      flock($f,3);
52      //закрываем файл
53      fclose($f);
54  }
55
56  //Перенаправляем пользователя обратно на guestbook.php
57  header('location:guestbook.php');
58  ?>
```

Сначала происходит проверка, каким образом был запущен скрипт: если как обработчик формы, то сообщение будет добавлено, в противном случае сразу осуществится переход к строке **57**, в которой пользователь перенаправляется на скрипт guestbook.php, для этого используется функция `header()`, ее синтаксис выглядит следующим образом:

```
header(заголовок);
```

Данная функция отправляет браузеру пользователя заголовок, переданный в качестве параметра. Помните, в *главе 8* был рассмотрен процесс обмена HTTP-заголовками. Функция `header()` как раз посылает такого рода заголовок. В описанной ранее программе это заголовок с именем `location`, который осуществляет перенаправление. Здесь очень удобна аналогия с сотовыми телефонами, они тоже позволяют делать перенаправление вызова, когда вы звоните на один номер, а звонок незаметно для вас перенаправляется на другой, в данном случае в программе делается что-то подобное.

Вы можете отследить заголовок `location` описанными в главе 8 способами.

Как правило, заголовок состоит из трех частей:

- своего имени;
- двоеточия (которое выступает в роли разделителя);
- значения заголовка.

В строке **57** используется следующая запись:

```
header('location: guestbook.php')
```

То есть передается заголовок `location` со значением `guestbook.php`. Для браузера это будет означать: немедленно осуществить переход к программе guestbook.php.

Таким образом, если вы просто запустите скрипт add_message.php через браузер, то увидите, что выполнен у вас будет guestbook.php, потому что именно на него вы будете перенаправлены в этом случае.

Очень важно вызывать функцию `header()` до любых функций или команд вывода текстовых данных. К примеру, на рис. 9.13 показан результат выполнения следующей программы (prog1.php):

```
<?php
echo "Привет пользователь!";
header('location:guestbook.php');
?>
```

Как видите, надпись `"Привет пользователь!"` была выведена, но функция `header()` вернула ошибку. Текст `Cannot modify header information - headers already sent` говорит о том, что заголовки уже были отправлены. Это произошло, т. к. ранее был осуществлен вывод данных (`output started at z:\home\guestbook.ru\www\prog1.php:2`) в строке **2** (цифра 2, отделенная от имени программы двоеточием, как раз указывает на это).

С аналогичным результатом (рис. 9.14) будет выполнена и следующая программа (prog2.php):

```
<B>Здравствуй пользователь!</B>
<I>Сегодня чудесный день.</I>
<?php
header('location:guestbook.php');
?>
```

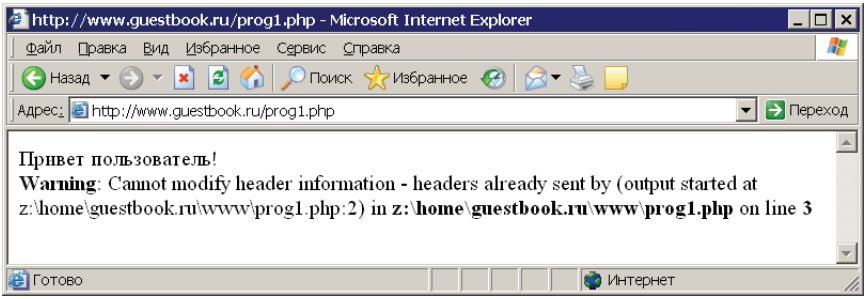


Рис. 9.13. Пример неправильного использования функции `header()`

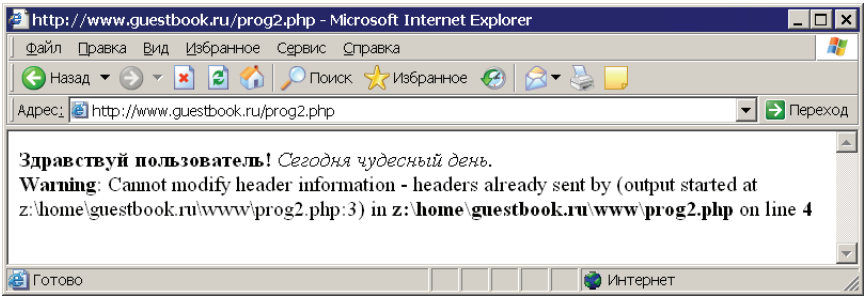


Рис. 9.14. Второй пример неправильного использования функции `header()`

Несмотря на то, что `header()` является первой PHP-функцией, которая используется в программе prog2.php, все равно возникает ошибка, т. к. до `header()` были использованы стандартные HTML-теги, которые тоже относятся к командам вывода. Поэтому всегда, когда вы хотите использовать функцию `header()` и сталкиваетесь с невозможностью отправки заголовка, обратите внимание, не выводите ли вы какую-то информацию пользователю ранее.

Возвращаемся к программе add_message.php. В строке **6** осуществляется проверка, ввел ли пользователь имя, если нет, то в строке **8**:

```
exit("Введите имя! <a href='guestbook.php'>Назад</a>");
```

используется команда выхода из программы `exit`. В качестве параметра, заключенного в круглые скобки и кавычки, передается строка текста, которая будет выведена пользователю при прекращении работы программы. У команды `exit` есть альтернативный вариант, когда она используется сама по себе, именно его вы можете наблюдать в строке **13** листинга 9.5 и в строках **10** и **26** листинга 9.3.

В строках **6** и **7** используется альтернативная запись условной конструкции `if` — без фигурных скобок. Данный вариант очень удобен, когда при выполнении условия должна выполняться всего одна команда (или функция). Рассмотрим 2 варианта кода, выполнение которого приведет к одному и тому же результату:

1 вариант кода

```
if  ($a==5) echo ("Переменная равна 5");
$a=$a+2;
```

2 вариант кода

```
if  ($a==5)
{
     echo ("Переменная равна 5");
}
$a=$a+2;
```

Оба варианта выполняют вывод строки "`Переменная равна 5`", если условие истинно (равно `TRUE`), далее вне зависимости от проверки условия к переменной `$a` будет прибавлена двойка. Но в данном случае первый вариант предпочтительнее, т. к. он более короткий.

В строке **9** программы add_message.php проверяется, ввел ли пользователь текст сообщения, но уже с помощью записи условной конструкции с фигурными скобками. В теле условия применяются команды `echo` и `exit`, что является альтернативным вариантом записи `exit("Текст")`, который был использован в предыдущем условии. Если пользователь при попытке сохранить сообщение, например, не ввел имя, то он увидит подсказку со ссылкой **Назад**, указывающей на guestbook.php (рис. 9.15).

В строках **17** и **19** имя пользователя и текст сообщения, введенные посетителем, передаются в качестве параметров функции `htmlspecialchars()`, результат ее выполнения сохраняется в переменных `$name_of_guest` и `$message_of_guest`. Функция `htmlspecialchars()` преобразует специальные символы, которые используются в HTML-тегах в их безопасные эквиваленты. Ее синтаксис:

```
htmlspecialchars(строка для преобразования);
```

Рассмотрим пример (prog3.php), результат выполнения которого изображен на рис. 9.16:

```
<?php
$stroka="<I>Прекрасный день!</I><B> И хорошая погода.</B>";
```

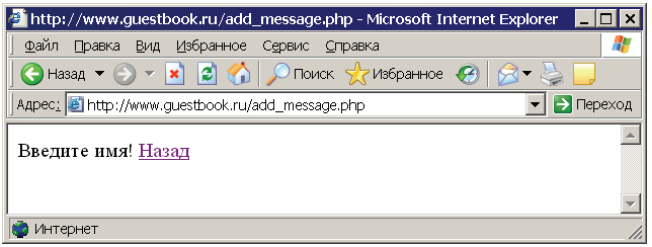


Рис. 9.15. Если пользователь хочет разместить сообщение, но не ввел имя, то он увидит данную подсказку

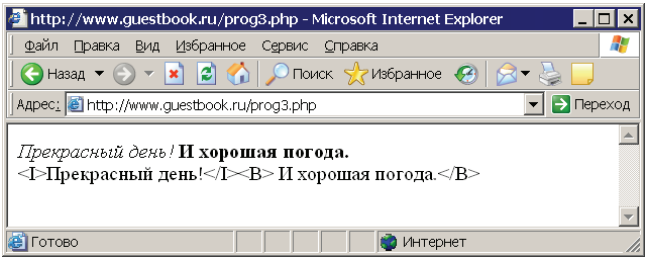


Рис. 9.16. Пример использования функции `htmlspecialchars()`

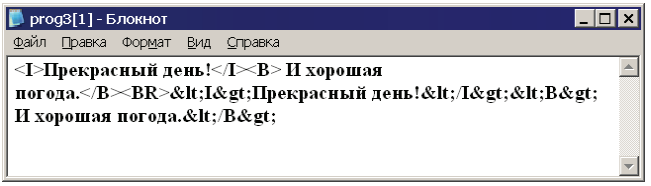


Рис. 9.17. Функция `htmlspecialchars()` в действии

```
echo $stroka."<BR>";
$new_stroka=htmlspecialchars($stroka);
echo $new_stroka
?>
```

Сначала мы выводим строку, содержащую HTML-теги, с помощью `echo`, а потом эту же строку перед выводом обрабатываем функцией `htmlspecialchars()`, что приводит к выводу HTML-тегов как обычного текста, потому что теперь они лишены своей функциональности. Если вы посмотрите HTML-код, присланный сервером браузеру, то увидите следующий результат — рис. 9.17.

Как видите, после обработки функцией `htmlspecialchars()` символ `<` заменен на `<`, а символ `>` заменен на `>`, теперь использование HTML-тегов в гостевой книге запрещено.

В строках с **21** по **24** вы можете встретить такую конструкцию:

```
if (filesize("gost.txt")>0)
        $first_message=FALSE;
    else
        $first_message=TRUE;
```

Ее можно описать следующим образом:

```
Если (размер файла gost.txt больше 0)
      то присваиваем переменной $first_message значение FALSE
  иначе
     присваиваем переменной $first_message значение TRUE
```

Таким способом реализуется защита от пустого сообщения в гостевой книге. О том, почему оно может появиться, мы поговорим чуть позже.

Функция `filesize()` возвращает размер в байтах указанного в качестве параметра файла, если при ее выполнении произошла ошибка, то будет возвращено `FALSE`. Синтаксис функции выглядит следующим образом: `filesize(имя файла)`.

В условии происходит сравнение результата выполнения функции `filesize()` с нулем. Если результат больше, то значит gost.txt уже содержит хотя бы одно сообщение, и переменной `$first_message` присваивается `FALSE`, в противном случае записываемое сообщение является первым, и переменной `$first_message` присваивается `TRUE`.

В строке **28** происходит открытие файла gost.txt в режиме `a` (только для записи). И опять используется защитная конструкция `die()`. Далее осуществляется запись информации с помощью знакомой вам функции `put()`, т. е. записывается сообщение, состоящее из 5 строк. Здесь используется символ окончания строки `\n`, давайте рассмотрим его назначение на следующем фрагменте кода:

```
fputs($f,'Первая строка');
fputs($f,'Вторая строка');
```

После его выполнения в файле окажется следующая информация:

```
Первая строкаВторая строка
```

Теперь рассмотрим следующий фрагмент:

```
Fputs($f,'Первая строка\n');
Fputs($f,'Вторая строка');
```

После выполнения данного кода в файле окажется следующая информация:

```
Первая строка
Вторая строка
```

Символ `\n` является системным и предназначен для перевода строки, это аналогично тому, когда вы пишете текст в Word, а затем нажимаете клавишу <Enter> для создания нового абзаца. Так же и с записью в текстовый файл средствами PHP: символ `\n` обозначает конец строки текста, поэтому при записи следующей строки с помощью функции `fputs()` мы можем быть уверены, что она будет располагаться на новой строчке.

В качестве первой строки записывается разделитель. Если это разделитель для самого первого сообщения, он будет записан в виде:

```
"——————\n"
```

иначе как:

```
"\n----------\n"
```

Первая ли это строка сообщения, выясняется с помощью условия:

```
if ($first_message==TRUE)
        fputs($f,"----------\n");
    else
        fputs($f,"\n----------\n");
```

где как раз и используется значение переменной `$first_message`, рассмотренной чуть раньше. Таким образом, если происходит запись второго сообщения, то первый `\n` осуществит перевод строки, это необходимо, чтобы первая строка следующего сообщения была записана с новой строчки.

В качестве второй строки записывается IP-адрес посетителя, который извлекается из переменной окружения `$_SERVER['REMOTE_ADDR']`. В третьей строке записывается дата размещения сообщения. Для чего используется функция `date()`, которая возвращает системную дату или время — это зависит от значения, переданного функции в качестве параметра. Синтаксис функции:

```
date(что выводить)
```

Если передать ей в качестве параметра значение `'d.m.y.'`, то функция вернет текущую дату в формате `дд.мм.гг`, то есть все элементы даты будут состоять из двух символов (пример: 10.12.05). Если передать в качестве параметра `'d.m.Y.'`, то функция вернет текущую дату в формате `дд.мм.гггг`, то есть под день и месяц будет отведено два символа, а под год — четыре (пример: 10.12.2005). А если передать в качестве параметра `'d/m/Y/'`, то функция вернет текущую дату в формате `дд/мм/гггг`.

В четвертой строке сообщения будет сохранено имя пользователя, которое находится в переменной `$name_of_guest`. В пятой строке записывается текст сообщения, взятый из переменной `$message_of_guest`. Далее представлена схема записи сообщения в файл gost.txt, для случаев, когда оно является первым (рис. 9.18), и когда ранее уже было добавлено одно сообщение (рис. 9.19).

Запустите скрипт guestbook.php и добавьте несколько сообщений. Теперь если вы откроете файл gost.txt, то увидите следующее — рис. 9.20.

Предлагаю провести вам небольшой эксперимент. В строке **17** скрипта add_message.php замените `$_POST["name_of_guest"]` на `$_POST["name"]`, в результате у вас должно получиться:

```
$name_of_guest=htmlspecialchars($_POST["name"]);
```

После этого сохраните изменения и запустите gostbook.php, далее введите имя, текст сообщения и нажмите **OK**, в результате вы увидите следующее (рис. 9.21).

При использовании Блокнота убедитесь, что режим переноса (пункт меню ***Формат | Перенос по словам)*** *выключен, иначе может показаться, что для хранения текста сообщения используется несколько строк.*

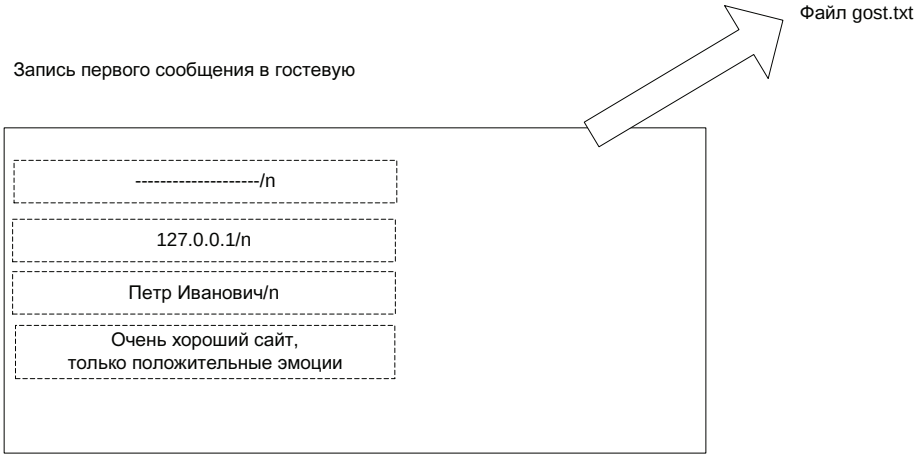


Рис. 9.18. Схема записи в файл gost.txt первого сообщения

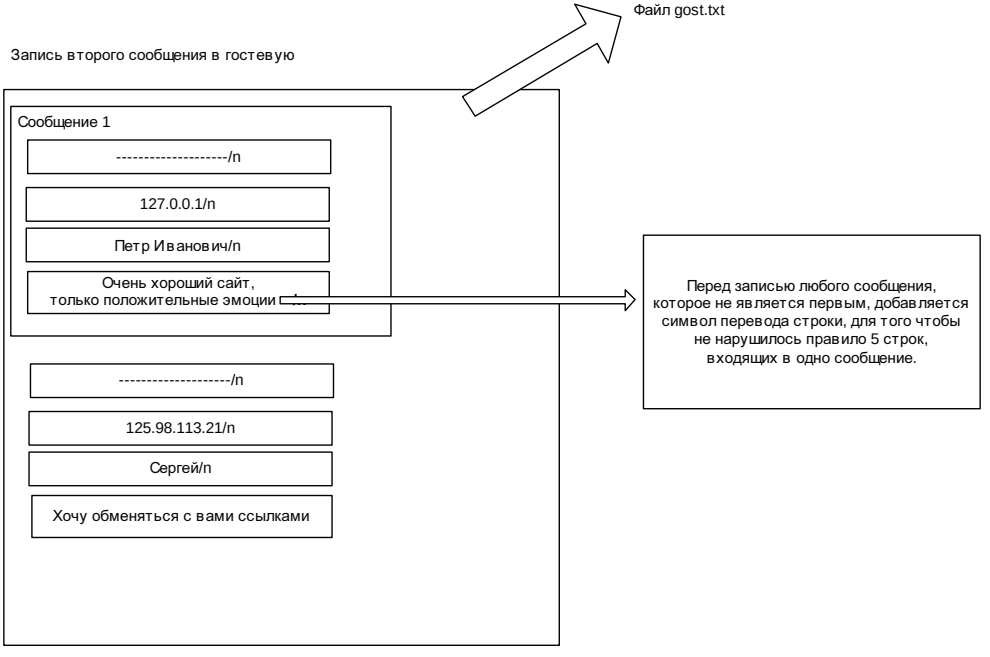


Рис. 9.19. Схема записи в файл gost.txt второго сообщения

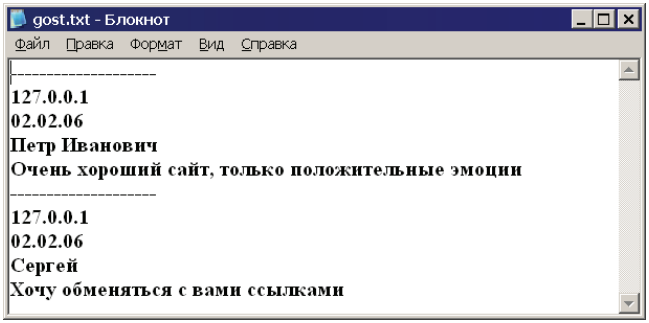


Рис. 9.20. Файл gost.txt, открытый с помощью Блокнота, после добавления двух сообщений

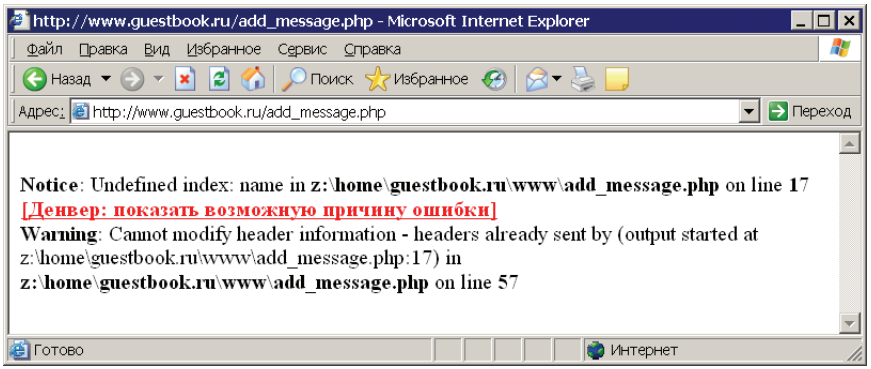


Рис. 9.21. Результат выполнения измененного файла add_message.php

Как видите, нам сообщается о том, что не существует элемента `name` массива `$_POST` (**Undefined index: name**), к которому происходит обращение в строке **17** программы. Поэтому если вы встречаете подобного рода ошибку, внимательно проверьте код программы на наличие обращения к несуществующим переменным или элементам массивов, т. к. это одна из самых распространенных ошибок начинающих программистов. Верните строку **17** к первоначальному виду и сохраните изменения.

9.2.2. Дорабатываем гостевую книгу

Рассмотрим случай, когда пользователь при вводе текста сообщения будет использовать клавишу <Enter>. Тогда каждое ее нажатие будет равнозначно наличию в тексте символа перевода строки `/n`, и такое сообщение будет записано в несколько строк.

Важно не только уметь разрабатывать программы, но и грамотно тестировать ее на различные действия со стороны пользователя, именно в этом заключается успех проектирования качественных программ, и это отличает новичка от опытного программиста, т. к. последний всегда предусматривает возможность возникновения нештатных ситуаций, вызванных либо действиями пользователя, либо сбоем в системе.

Но самое интересное произойдет при выводе всех сообщений из файла gost.txt. Предположим, пользователь ввел текст "Я хочу <здесь он нажал Enter> узнать ваш e-mail". При выводе всех сообщений, который происходит в файле guestbook.php, для данного сообщения в качестве текста будет выведено только "Я хочу", а следующая строка "узнать ваш e-mail" будет прочитана как разделитель. Таким образом структура файла нарушится, и соответственно на экране пользователь увидит не то, что нужно.

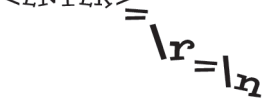
Исправить данный недостаток поможет функция `str_replace()`. Ее синтаксис:

```
str_replace(что_ищем,
на_что_заменяем,
где_ищем);
```

Данная функция ищет подстроку, переданную в параметре `что_ищем`, в строке, переданной в качестве параметра `где_ищем`. Если подстрока найдена, функция заменяет ее на значение, переданное в параметре `на_что_заменяем`. Результатом выполнения функции является измененная строка. Рассмотрим следующий пример:

```
<?php
$a="Я увидел красный автомобиль на дороге";
$b=str_replace('красный','зеленый',$a);
echo $b;
?>
```

В этом небольшом фрагменте кода осуществляется поиск в переменной `$a` слова `красный` и замена его на `зеленый`, при этом измененный текст сохраняется в переменной `$b` как результат работы функции `str_replace`. После выполнения этой небольшой программы на экран будет выведен текст `Я увидел зеленый автомобиль на дороге`.



Вернемся к нежелательному переводу строки, от которого необходимо защитить гостевую книгу. Перевод строки в Windows осуществляется следующими символами:

- `\r`
- `\n`

Каждый раз, когда вы нажимаете в текстовом редакторе клавишу <Enter>, происходит вставка именно этих символов. Хотя вы этого и не видите на экране, но они реально существуют, поэтому с ними можно работать так же, как с обычными символами. Может возникнуть вопрос: "Почему же когда мы осуществляем запись,

мы указываем только символ \n?". Дело в том, что PHP берет эту работу на себя. Если работа с файлом осуществляется в операционной системе Windows, он автоматически подставит недостающий символ \r. Если же работа осуществляется в UNIX, то никакой подстановки не будет, т. к. в этой ОС конец строки обозначается всего лишь одним символом \n. Поэтому наша программа будет корректно работать в обоих ОС.

Чтобы реализовать защиту от лишних пользовательских <Enter>, нам всего лишь нужно воспользоваться функцией str_replace(): с ее помощью найти скрытые символы перевода строки и заменить их на простой пробел. Данный алгоритм реализуется следующими двумя строчками кода:

```
$message_of_guest=str_replace(chr(10),' ',$message_of_guest);
$message_of_guest=str_replace(chr(13),' ',$message_of_guest);
```

которые необходимо вставить после кода:

```
//Получаем значения поля Имя
$name_of_guest=htmlspecialchars($_POST["name_of_guest"]);
//Получаем значения поля Текст
$message_of_guest=htmlspecialchars($_POST["message_of_guest"]);
```

Функция chr() возвращает символ по его коду. Рассмотрим работу с этой функцией более подробно. Каждый символ в операционной системе представлен его числовым эквивалентом — кодом. Это касается как видимых символов, так и скрытых. Например, выполнение следующей программы (prog4.php):

```
<?php
echo chr(110);
echo chr(115);
echo chr(119);
?>
```

приведет к результату, показанному на рис. 9.22.

В этом случае намного удобнее просто написать echo "nsw" — результат будет тем же самым. Но со скрытыми символами дело обстоит иначе, их нельзя просто

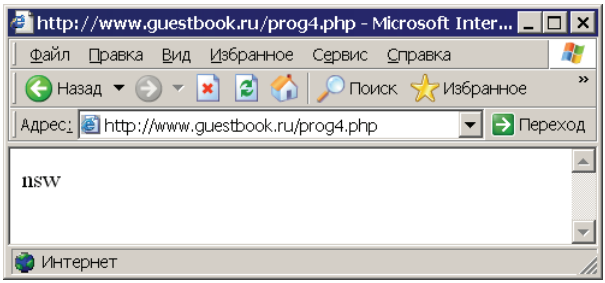


Рис. 9.22. Вывод строки nsw с помощью функции chr()

набрать на клавиатуре, соответственно, чтобы передать их в качестве параметров функции `str_replace()`, нужно использовать функцию, которая осуществляет вывод этих символов по их коду, что как раз и делает функция `chr()`. Конечно, очень было бы удобно написать следующим образом:

```
$message_of_guest=str_replace('/r',' ',$message_of_guest);
$message_of_guest=str_replace('/n',' ',$message_of_guest);
```

Но, к сожалению, данный код работать не будет.

Есть еще один нежелательный момент, от которого пока не защищена наша программа — это ограничение на количество сохраняемой информации для каждого сообщения. Например, 100 символов для имени посетителя и 1000 символов для сообщения будет вполне достаточно. Воспользуемся функцией `substr()`. Ее синтаксис таков:

```
substr(строка, откуда_начать, количество_символов)
```

Данная функция производит над строкой, переданной в первом параметре, следующее действие: пропускает количество символов, переданных во втором параметре относительно начала, и возвращает подстроку длиной равной числу, переданному в третьем параметре. Например, имеется строка: "`На небе сегодня тучи`", если выполнить следующий код:

```
$s=substr("На небе сегодня тучи", 3, 4);
echo $s;
```

то результатом функции будет строка "`небе`". Нумерация строки начинается с 0. Отсчитываем 3 символа — это будет буква "`н`", далее возвращаем следующие 4 символа, соответственно получается слово "`небе`", вот таким образом работает функция `substr()`. Следующие две строчки необходимо добавить после использования `htmlspecialchars()`, но до использования `str_replace()`:

```
//Если пользователь ввел больше символов в имени, чем нужно,
//то удаляем все лишнее
$name_of_guest=substr($name_of_guest,0,100);
//аналогично и с текстом сообщения
$message_of_guest=substr($message_of_guest,0,1000);
```

Таким образом осуществляется сохранение в переменной `$name_of_guest` первых 100 символов имени и в `$message_of_guest` первых 1000 символов сообщения. Поэтому, если пользователь введет имя, которое будет состоять из 150 символов, то последние 50 символов будут урезаны.

Так как было введено ограничение, то неплохо было бы в файле guestbook.php перед выводом элементов для ввода добавить информацию о максимально возможном количестве символов. Это можно сделать следующим образом:
строку **9**:

```
Имя<BR>
```

заменить на:

```
Имя <font size=-2>(не более 100 символов)</font>:<BR>
```

строку **12**:

```
Мысли<BR>
```

заменить на:

```
Мысли <font size=-2>(не более 1000 символов)</font>:<BR>
```

9.2.3. Цензура не дремлет

В любой системе, связанной с общением — будь это форум, чат, или конференция, — желательно иметь механизм цензуры. Конечно же, он будет разработан и для гостевой книги. Как вариант, можно использовать механизм премодерации — когда сообщение, прежде чем станет доступным для всеобщего просмотра, сначала попадает в специальное хранилище (в роли которого может выступать отдельный файл). И до того времени, пока администратор не просмотрит его и не разрешит ему быть видным для общественного просмотра, никто другой не сможет его увидеть. Но как быть в случаях, когда у вас не хватает времени? Вы заняты всю неделю, а сообщений очень много — они просто будут копиться, а сайт будет выглядеть одиноко, т. к. создастся ощущение, что никто его не посещает. В решении данной проблемы поможет цензура — конечно, ее, как и любую систему проверки, можно при желании обойти, но все равно она будет достойным оружием в борьбе за красивую речь.

Код, отвечающий за реализацию цензуры, вынесем в отдельный модуль. Технология его работы очень проста — у нас будет массив, в котором хранится набор запрещенных цензурой слов. Данные, введенные пользователем, будут проверяться на наличие в них совпадений из массива, если они будут, то произойдет замена этих слов на пробелы. Создайте файл censura.php (листинг 9.6).

Листинг 9.6. Файл censura.php

```
1   <?php
2*  $mas=array('Плохое слово1','Плохое слово2','Плохое слово3',
    'Плохое слово4');
3
4   //обрабатываем цензурой имя и сообщение
5   foreach ($mas as $line)
6   {
7      $name_of_guest=str_replace($line,' ',$name_of_guest);
8      $message_of_guest=str_replace($line,' ',$message_of_guest);
9   }
10  ?>
```

Проверка наличия "плохих слов" в строке текста (имени и сообщении) осуществляется в два этапа. Сначала организуется цикл, перебирающий все элементы массива, в цикле используется уже знакомая функция `str_replace()`, которая ищет значение очередного элемента массива в тексте, если оно будет найдено, то произойдет замена на пробел, иначе просто осуществится переход к следующему элементу массива, и поиск повторится уже по его значению. Цикл организуется с помощью новой конструкции `foreach`, которая имеет следующий синтаксис:

```
foreach (массив as значение_очередного_элемента)
{
    тело_цикла
}
```

В качестве параметра `массив` передается имя массива, значения которого необходимо перебрать, а в качестве параметра `значение_очередного_элемента` выступает переменная, в которой будет сохраняться содержимое очередного элемента массива. На рис. 9.23 представлена схема, иллюстрирующая принцип работы цикла `foreach`.

Рассмотрим пример (prog5.php):

```
<?php
$arr_colors = array("Красный", "Зеленый", "Синий", "Желтый");
foreach ($arr_colors as $value)
{
   echo  $value."<BR>";
}
?>
```

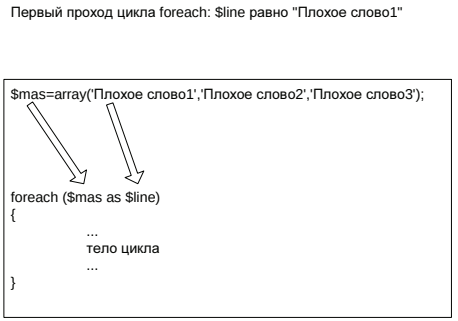


Рис. 9.23. Схема, иллюстрирующая принцип работы цикла `foreach`

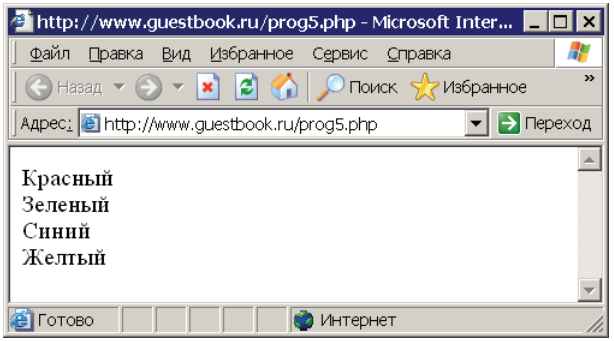


Рис. 9.24. Пример работы с циклом `foreach`

Его результат работы изображен на рис. 9.24.

Конструкция `foreach` организует цикл, который будет выполнен столько раз, сколько элементов в массиве, имя которого было передано в конструкцию в качестве параметра. Причем на каждом шаге цикла в переменной (в примере с цветами это `$value`), переданной в качестве второго параметра, будет содержаться значение очередного элемента массива. Слово `as` является системным и используется как разделитель параметров конструкции `foreach`.

Теперь осталось подключить модуль цензуры к файлу add_message.php, для этого откройте его и сразу после использования функции `htmlspecialchars()` вставьте следующую строку:

```
require_once('censura.php');
```

В листинге 9.7 представлен полный файл add_message.php.

Листинг 9.7. Окончательный вариант файла add_message.php

```
<?php
//Скрипт будет работать только при условии, что он был запущен
//как обработчик формы, размещенной на guestbook.php
if (isset($_POST['okbutton']))
{
   if ($_POST['name_of_guest']=='')
      exit("Введите имя! <a href='guestbook.php'>Назад</a>");

   if ($_POST['message_of_guest']=='')
   {
      //А тут более распространенный вариант в два оператора
      echo "Введите текст сообщения! <a href='
guestbook.php'>Назад</a>";
```

```
13        exit;
14      }
15
16      //Получаем значения поля Имя
17      $name_of_guest=htmlspecialchars($_POST["name_of_guest"]);
18      //Получаем значения поля Мысли
19      $message_of_guest=htmlspecialchars($_POST["message_of_guest"]);
20
21      //Подключаем цензуру
22      require_once('censura.php');
23
24      //Если пользователь ввел больше символов в имени, чем нужно,
25      //то удаляем все лишнее
26      $name_of_guest=substr($name_of_guest,0,100);
27      //аналогично и с текстом сообщения
28      $message_of_guest=substr($message_of_guest,0,1000);
29
30      //Удаляем переводы строки
31      $message_of_guest=str_replace(chr(10),' ',$message_of_guest);
32      $message_of_guest=str_replace(chr(13),' ',$message_of_guest);
33
34      if (filesize("gost.txt")>0)
35            $first_message=FALSE;
36       else
37            $first_message=TRUE;
38
39      //Открываем файл gost.txt для записи,
40      //если файла нет, то он будет создан
41      $f=fopen("gost.txt","at") or die("Не могу открыть файл");
42
```

```
43      //Блокируем файл, чтобы никто не мог к нему обратиться,
44      //пока мы с ним не закончим работу
45      flock($f,2);
46
47      //Записываем строку "-----------------"
48      //Если это первое сообщение, то без \n в начале, иначе с ним
49      if ($first_message==TRUE)
50           fputs($f,"--------------------\n");
51       else
52           fputs($f,"\n--------------------\n");
53
54      //Записываем IP-адрес пользователя
55      fputs($f,$_SERVER['REMOTE_ADDR']."\n");
56      //Записываем дату сообщения
57      fputs($f,date('d.m.y')."\n");
58      //Записываем имя
59      fputs($f,$name_of_guest."\n");
60      //Записываем текст
61      fputs($f,$message_of_guest);
62
63      //снимаем блокировку
64      flock($f,3);
65      //закрываем файл
66      fclose($f);
67  }
68
69  //Перенаправляем пользователя обратно на guestbook.php
70  header('location:guestbook.php');
71  ?>
```

9.2.4. Управляем с удобством — админка

Ни один более менее серьезный сайт не обходится без его центра — панели администратора или, как его часто называют, админки. Для гостевой книги было бы очень удобно иметь админку, которая позволит вести работу в привилегированном режиме — то есть с возможностью просматривать сообщения вместе с IP-адресом человека, который их разместил, а также удалять их, например, если модуль цензуры не справился с чем-то. Конечно, можно вручную открыть файл gost.txt, но намного удобней будет работать с сообщениями посредством специального инструмента.

Создайте в guestbook.ru/www папку adminka — именно там будет располагаться админка. Администратор будет заходить под именем admin и паролем, который будет храниться в файле pas.txt. В настоящее время никто не хранит пароль в открытом виде — используется его хеш, который представляет собой специальный набор символов, получаемый в результате шифрования пароля по алгоритму MD5, реализованному в PHP в виде функции `md5()`, ее синтаксис:

```
md5(строка);
```

Функция зашифровывает строку, переданную в качестве параметра, по алгоритму MD5 и возвращает результат этого действия. Рассмотрим небольшой пример (prog6.php):

```
<?php
$our_password="12345";
$res=md5($our_password);
echo $res;
?>
```

Результат его выполнения показан на рис. 9.25.

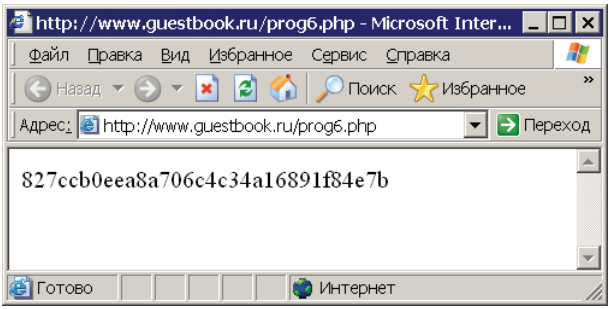


Рис. 9.25. Пример использования функции шифрования `md5()`

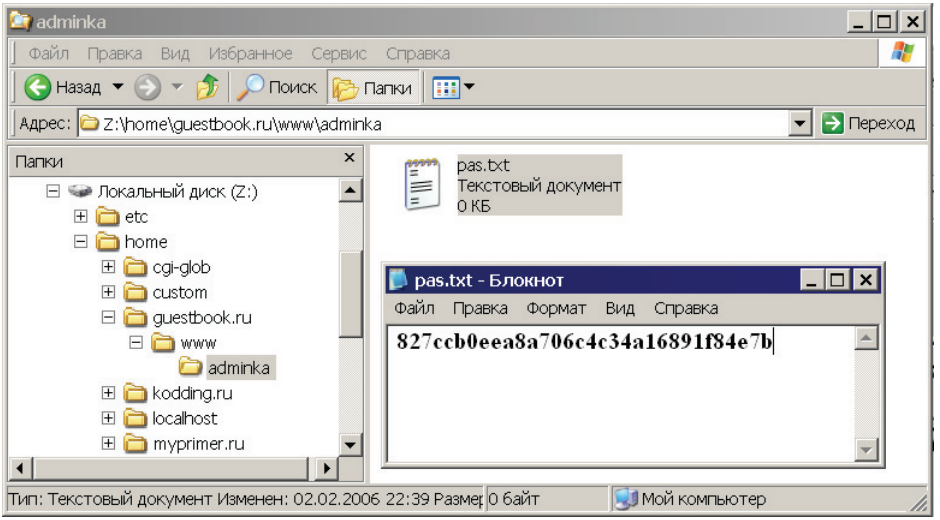


Рис. 9.26. Создание файла pas.txt, предназначенного для хранения пароля администратора

Скопируйте полученное значение и сохраните его в файле pas.txt, который нужно предварительно создать в папке adminka (рис. 9.26).

Именно со значением файла pas.txt будет происхо-дить сравнение при аутентификации пользователя, пытающегося зайти в админку.

Приступаем к разработке формы авторизации. Создайте файл index.php в папке adminka, таким образом, если вы наберете в браузере **guestbook.ru/adminka**, сразу будет запущен именно этот файл. Его содержание представлено в листинге 9.8.

Таким образом, если кто-то получит файл pas.txt, то сам пароль администратора от этого известен не станет.

Листинг 9.8. Файл adminka/index.php

```
1  <?php
2  //Если нажата кнопка "Войти"
3  if (isset($_POST['enter']))
4  {
5      //Читаем хеш из файла pas.txt
6      $s=file('pas.txt');
```

```
7       //Получаем хеш от пароля, введенного пользователем
8       $hash=md5($_POST['passwd']);
9       //Сравниваем хеш пароля и логин с теми,
10      // которые ввел пользователь
11      if (($s[0]==$hash) and ($_POST['login'] =='admin'))
12      {
13         //Подключаем модуль, выводящий все сообщения
14         require_once("list_message.php");
15         //Больше ничего делать не надо, поэтому выходим
16         exit;
17      }
18      else
19      {
20        //Если пароль или имя пользователя неправильные,
21        //выводим информацию об этом
22        echo "Логин или пароль неверные ";
23        echo "<a href='index.php'>Назад</a>";
24        exit;
25      }
26  }//end - if (isset($_POST['enter']))
27
28  ?>
29  <!--форма-->
30  <FORM action='' method='POST'>
31  Логин:<input type='text' name='login'>
32  <BR><BR>
33  Пароль:<input type='password' name='passwd'>
34  <BR><BR>
35  <input type='submit' name='enter' value='Войти'>
36  </FORM>
```

Сначала производится проверка того, как запущен скрипт index.php. Если как обработчик формы авторизации, то проверяется корректность логина и пароля, введенных пользователем. Если они верны, то происходит подключение модуля list_message.php, назначение которого будет рассмотрено чуть позже. Если index.php запущен в первый раз, то пользователю выводится форма для авторизации (см. строки с **30** по **36**). На рис. 9.27 в виде схемы изображен процесс аутентификации.

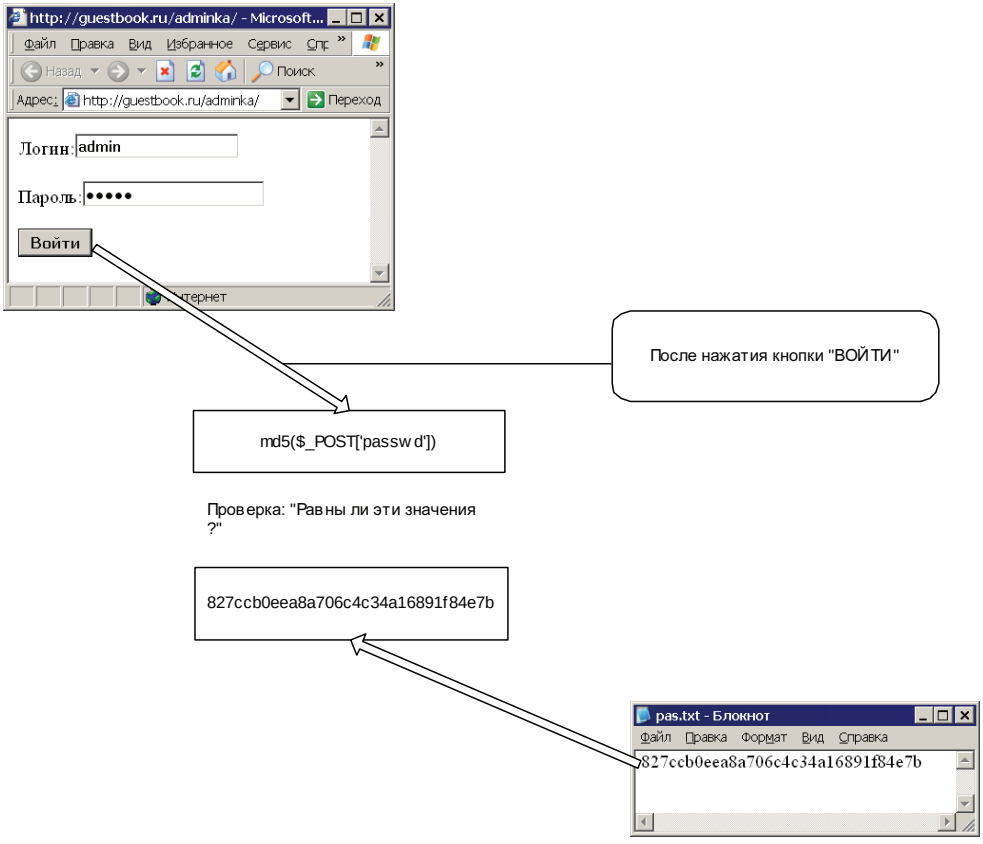


Рис. 9.27. Процесс проверки пароля пользователя, пытающегося зайти в администраторскую часть гостевой книги

В листинге 9.9 представлен модуль list_message.php, его также необходимо создать в папке adminka. Данный модуль дополнительно выводит для каждого сообщения его номер, IP-адрес пользователя, который его разместил, а также кнопку **Удалить**, нажатие которой осуществляет удаление выбранного сообщения из гостевой.

Листинг 9.9. Файл adminka/list_message.php

```
1  <?php
2  $f=fopen("..\gost.txt","rt") or die("Не могу открыть файл");
3
```

```
4  //Переменная, отвечающая за номер сообщения
5  $i=1;
6  while (!feof($f))
7  {
8          //Выводим номер сообщения
9          echo "Сообщение №".$i."<br>";
10         //Читаем разделитель "--------------------"
11         $hide_line=fgets($f);
12         //Получаем IP-адрес
13         $ip_=fgets($f);
14                 //Выводим IP-адрес
15                 echo "IP-адрес: ".$ip_."<br>";
16         //Получаем дату сообщения
17         $data=fgets($f);
18                  //Выводим Дату сообщения
19                  echo "Дата: ".$data."<br>";
20         //Читаем имя человека, который его написал
21         $data=fgets($f);
22                 //Выводим имя
23                 echo "Имя: ".$data."<br>";
24         //Читаем текст сообщения
25         $data=fgets($f);
26                 //Выводим текст сообщения
27                 echo "Сообщение: ".$data."<br>";
28
29         //Выводим для очередного сообщения кнопку УДАЛИТЬ в форме
30         ?>
31         <form action="delete_message.php?num=<?=$i?>" method="POST">
32         <input type="submit" name="delete_button" value="Удалить">
33         </form>
34         <?php
35         //Отделяем сообщение горизонтальной линией
36         echo "<HR>";
37         //Увеличиваем количество выведенных сообщений на 1
38         $i++;
39 }
40 ?>
```

В строке **2** происходит открытие файла, в режиме `r`. Далее объявляется переменная `$i`, в которой будет храниться номер текущего сообщения. Затем следует цикл `while`, имеющий следующую логику работы:

```
Пока (не прочитаны все сообщения)
{
       выводим очередное сообщение и кнопку "Удалить"
       переходим к следующему сообщению
}
```

При нажатии кнопки **Удалить** будет вызван скрипт delete_button.php, ему будет передан методом `GET` параметр `num` с номером сообщения, которое необходимо удалить. Внимания заслуживает строка **31**:

```
<form action="delete_message.php?num=<?=$i?>" method="GET">
```

Посмотрите на `<?=$i?>`, это краткая запись следующего кода:

```
<?php
echo $i
?>
```

Данный вариант использования `echo` очень часто практикуется при вставке в HTML-тег PHP-кода, т. к. он очень компактный.

Теперь создайте в папке adminka файл с именем delete_message.php, который будет производить удаление выбранного администратором сообщения (листинг 9.10).

Листинг 9.10. Файл adminka/delete_message.php

```
1  <?php
2  //Если скрипт был вызван для удаления
3  //выбранного администратором сообщения, то
4  if (isset($_POST['delete_button']))
5  {
6    //Получаем содержимое файла
7    $lines = file('..\gost.txt');
8
9    //Открываем файл в режиме r+
10   $f=fopen("..\gost.txt","r+t") or die("Не могу открыть файл");
11   flock($f,2);
12   ftruncate($f,0);
13
14   //Количество выведенных сообщений
15   $a=0;
16   //Текущий элемент массива
17   $s=0;
18
```

```
19    //Пока не обработаны все сообщения, выполнять тело цикла
20    while (!(count($lines)==$s))
21    {
22      //Если следующее сообщение равно тому, которое нужно удалить,
23      //то его просто пропускаем
24      if (!($a+1==$_GET['num']))
25      {
26         //Записываем строку "-----------------"
27         fputs($f,$lines[$s]);
28         $s++;
29         //Записываем IP-адрес пользователя
30         fputs($f,$lines[$s]);
31         $s++;
32         //Записываем дату сообщения
33         fputs($f,$lines[$s]);
34         $s++;
35         //Записываем имя
36         fputs($f,$lines[$s]);
37         $s++;
38         //Записываем текст
39         fputs($f,$lines[$s]);
40         $s++;
41      }
42      else
43      {
44         //Сразу переходим на 5 элементов вперед
45         //Для того чтобы пропустить удаляемое сообщение
46         $s=$s+5;
47      }//end - if (!($a+1==$_GET['num']))
48
49      //Прибавляем на 1 количество выведенных сообщений
50      $a++;
51    }//end - while
52
53    //Снимаем блокировку
54    flock($f,3);
55    //Закрываем файл
56    fclose($f);
57  }//end if (isset($_POST['delete_button']))
58
59  //Перенаправляем обратно на list_message.php
60  header("location: list_message.php");
61  ?>
```

В строке **4** осуществляется проверка, как запущен скрипт, если для удаления сообщения, т. е. нажатием кнопки **Удалить** в панели администратора, то начнется

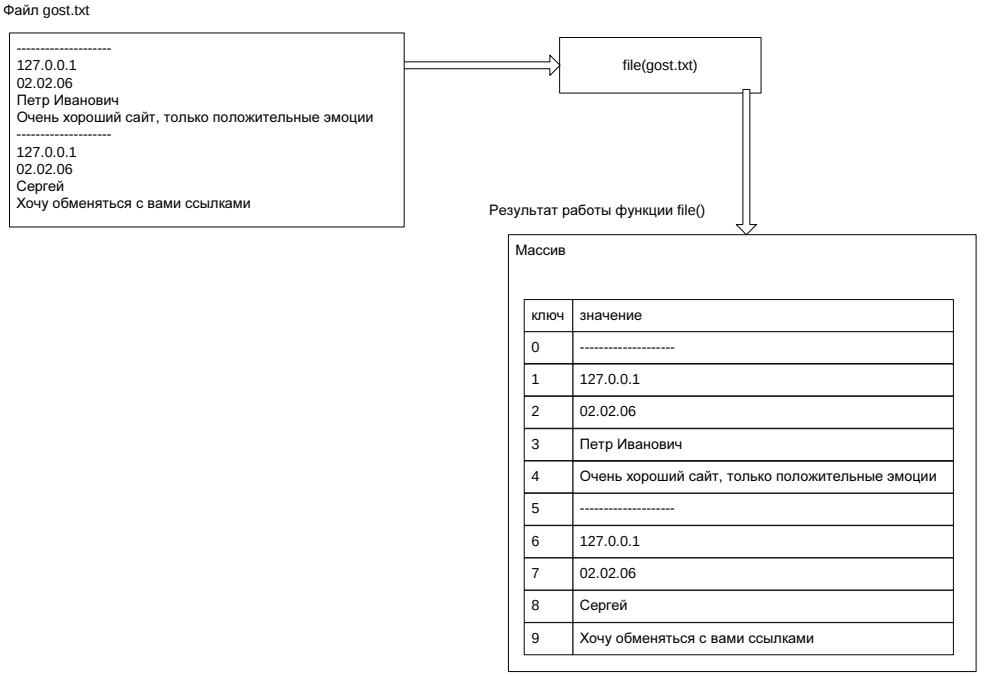


Рис. 9.28. Схема работы функции `file()`

выполняться строка **7**, в которой происходит считывание содержимого файла gost.txt в массив `$line`, делается это с помощью функции `file()`, ее синтаксис:

```
file(имя_файла)
```

Данная функция делит содержимое файла, переданного в качестве параметра, на отдельные строки, которые затем возвращает как элементы массива. Схема работы функции изображена на рис. 9.28.

Стоит отметить, что в качестве параметра функции `file()` передается значение `'..\gost.txt'`, где две точки указывают на родительский каталог. Это необходимо, т. к. при работе с админкой текущим каталогом является adminka — именно там находятся ее файлы, и если просто указать `'gost.txt'`, то файл найден не будет. Вы можете провести эксперимент, для этого в строках **7** и **10** замените `'..\gost.txt'` на `'gost.txt'`, после чего можно будет наблюдать следующее сообщение об ошибке — рис. 9.29.

В результате работы функции `file()` каждая строка файла gost.txt будет представлена как отдельный элемент массива `$line`.

Далее происходит открытие файла gost.txt в режиме `r+t`, после этого он блокируется и все его содержимое удаляется с помощью функции `ftruncate()`. Логика

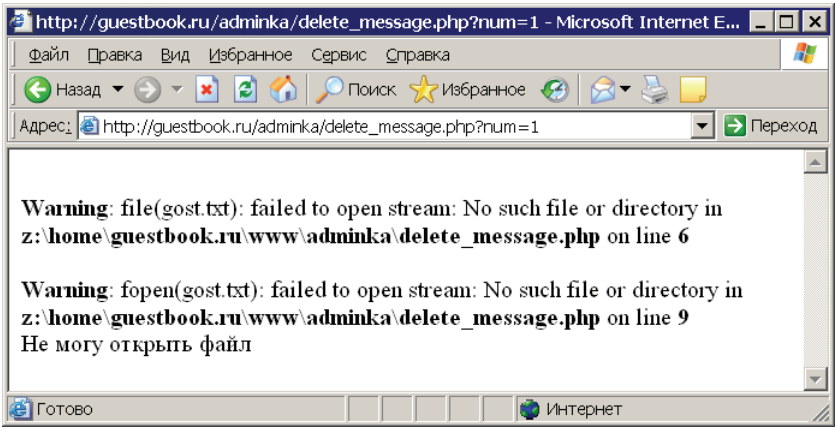


Рис. 9.29. Ошибка, возникающая вследствие неправильного указания пути файла

остальной части программы очень проста, в файл gost.txt происходит запись по одному сообщению, которые берутся из массива `$line`, при этом осуществляется проверка, не является ли это сообщение тем, которое удалил администратор гостевой (номер удаляемого сообщения хранится в `$_GET['num']`). Если это так, то оно просто пропускается и не попадает в файл, где хранятся сообщения.

В условии цикла `while` (строка **20**) используется новая функция `count()`, которая возвращает количество элементов в массиве, в качестве параметра передается имя массива. Таким образом, `count($lines)` вернет количество элементов в массиве `$line`.

9.2.5. Использование сессий

У разработанной админки есть один существенный минус, который заключается в том, что шаг авторизации можно обойти, т. к. ничто не мешает любому пользователю запустить в браузере скрипт list_message.php и спокойно с ним работать (предлагаю провести вам такой эксперимент). Выходом из этой ситуации является использование сессий.

Сессия — это механизм, позволяющий серверу идентифицировать пользователя посредством уникального номера, который назначается сеансу работы пользователя с сервером.

Для того чтобы инициализировать сессию, необходимо использовать функцию `session_start()`, после чего посетителю будет присвоен уникальный номер — идентификатор, который сохранится как на сервере, так и на компьютере пользователя.

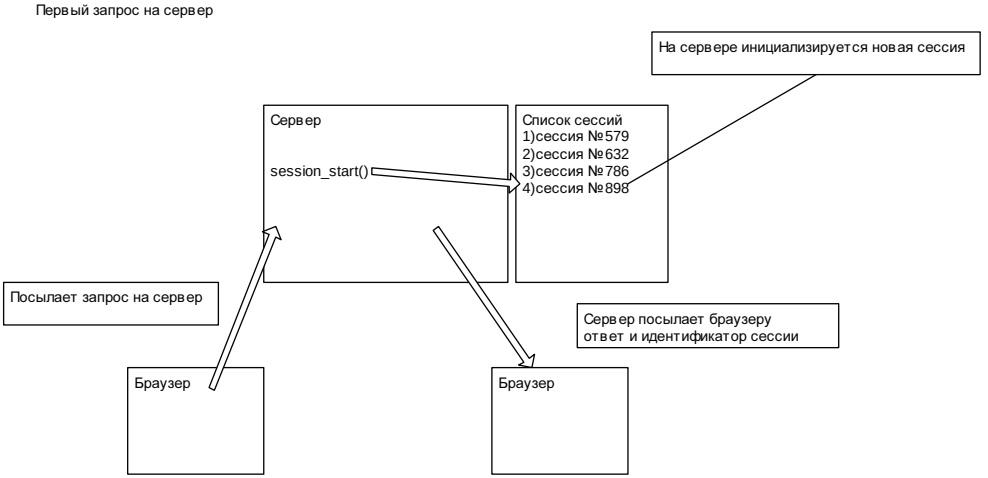


Рис. 9.30. Создание сессии при первом обращении браузера к серверу

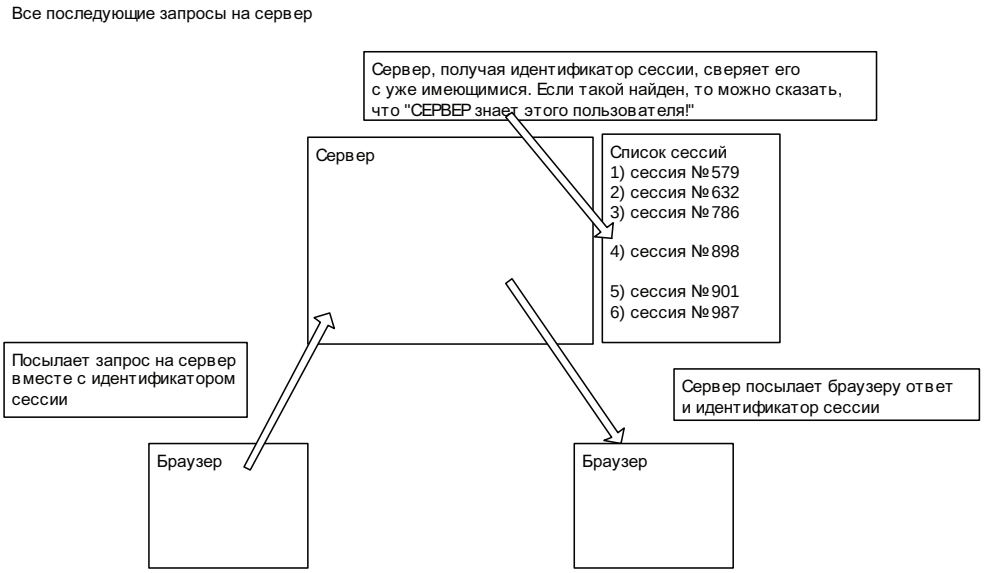


Рис. 9.31. Обработка сервером ранее созданной сессии при последующем обращении браузера к серверу

При следующем обращении пользователя к тому же серверу незаметно для него вместе с его запросом будет отправлен и идентификатор сессии. Сервер, получив его, сверяет с теми, которые у него имеются, и соответственно может определить, обращался этот пользователь к серверу раньше или нет (рис. 9.30 и 9.31).

Самое важное свойство сессий заключается в том, что они могут хранить данные на сервере, время жизни которых равно времени жизни сессии. Для этого используется специальный массив `$_SESSION`, если необходимо что-то сохранить, достаточно объявить элемент этого массива с нужным именем и присвоить ему сохраняемое значение. При следующем обращении пользователя к серверу ваш скрипт запросто может обратиться к этому элементу массива `$_SESSION`.

Идея авторизации, которая лишена описанного ранее недостатка, состоит в том, чтобы после авторизации пользователя создать элемент `$_SESSION['autorized']` и присвоить ему значение `TRUE`. А дальше в каждом скрипте, который входит в админку, просто проверять наличие `$_SESSION['autorized']`. Если данный элемент массива создан и равен `TRUE`, то значит это авторизованный администратор, в противном случае это просто шутник, который хочет обойти авторизацию.

Как уже упоминалось ранее, для инициализации сессии используется функция `session_start()`, которую обязательно нужно использовать в самом начале программы. Рассмотрим следующий пример:

```
<html>
<head>
  <title>Работа с сессиями</title>
</head>
<?php
session_start();
?>
```

и еще один:

```
<?php
echo "Привет";
session_start();
?>
```

Результат выполнения обоих примеров приведет к ошибке, т. к. объявление сессии идет не в самом начале программы. Правильным вариантом будет следующий:

```
<?php
session_start();
?>
<html>
<head>
  <title>Работа с сессиями</title>
```

```
</head>
```

и

```
<?php
session_start();
echo "Привет";
?>
```

После того как сессия объявлена, в ней можно сохранять данные. Рассмотрим следующий пример (prog7.php):

```
<?php
//Объявляем сессию
session_start();
//Сохраняем в сессии значение
$_SESSION['dat']='a';
//Выводим сохраненное значение на экран
echo $_SESSION['dat'];
?>
```

Результат его работы представлен на рис. 9.32.

После того как вы выполните приведенный ранее пример, не торопитесь закрывать браузер, а откройте с помощью проводника папку tmp, где как раз и хранится информация о сессиях (рис. 9.33).

Найдите самый поздний по времени создания файл — он будет хранить информацию для сессии, которая была создана с помощью прошлого примера (prog7.php). Откройте этот файл, вы увидите следующий результат — рис. 9.34.

Можно увидеть, что значение `dat` равно `a` — это то, что было сохранено в сессии.

Теперь немного усложним пример prog7.php:

```
<?php
//Объявляем сессию
session_start();
//Если пользователь обращается к программе повторно,
//то прибавляем к элементу dat букву "a"
```

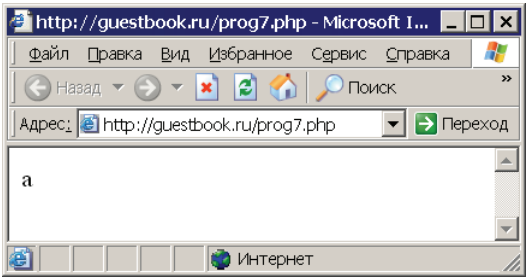


Рис. 9.32. Простой пример использования сессии

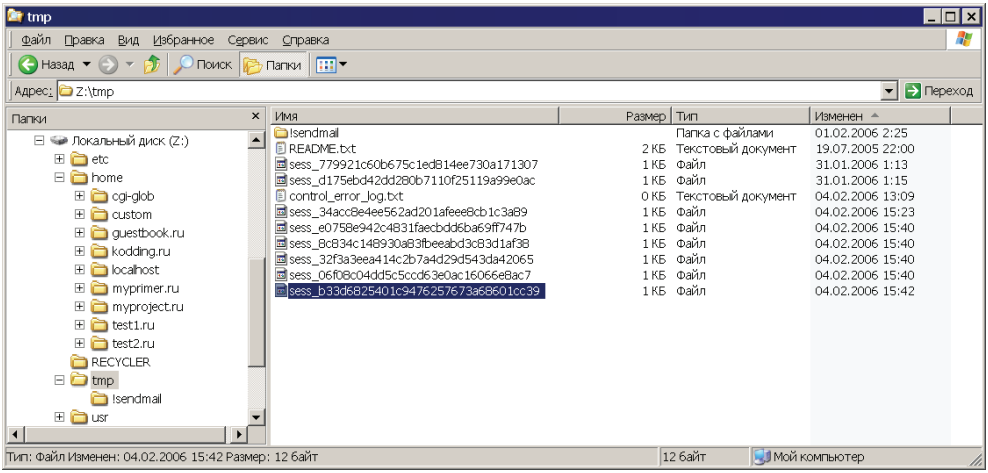


Рис. 9.33. Информация о сессиях, хранимая на сервере

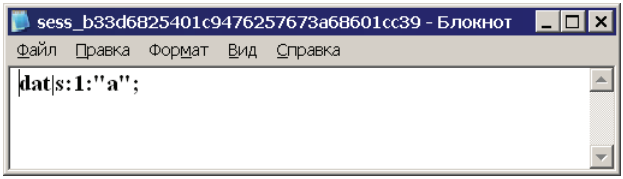


Рис. 9.34. Содержимое файла, который хранит информацию об определенной сессии

```
if (isset($_SESSION['dat']))
      $_SESSION['dat']=$_SESSION['dat'].'a';
//Если пользователь запускает скрипт первый раз,
//то создаем элемент "a"
else
      $_SESSION['dat']='a';

//Выводим значение элемента dat
echo $_SESSION['dat'];
?>
```

Как видите, здесь осуществляется проверка: если элемент `dat`, который должен храниться в сессии, уже создан, то к его содержимому прибавляется буква "a", иначе просто создается элемент `dat`. Запускайте программу, вы увидите такой же результат, как и на рис. 9.32. А теперь несколько раз нажмите в браузере кнопку **Обновить** — вы увидите, что количество букв "a" растет, т. е. изменяется значение элемента `dat`, хранящегося в сессии (рис. 9.35).

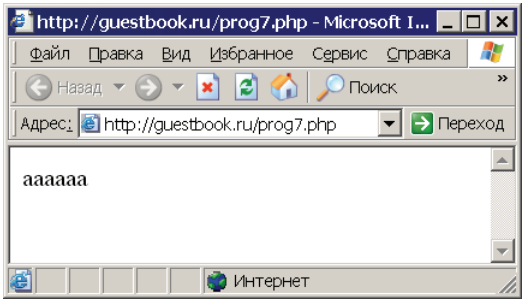


Рис. 9.35. Результат выполнения программы prog7.php после 5 нажатий кнопки **Обновить**

А теперь закройте браузер и запустите этот же пример еще раз, в результате вы увидите всего лишь одну букву "a". Пусть вас это не удивляет, дело в том, что информация о сессии уничтожилась, т. к. сессия живет только до закрытия браузера. Запуск скрипта во вновь открытом окне браузера приведет к созданию новой сессии, что и произошло в данном случае. Уничтожить сессию можно также с помощью функции `session_destroy()`.

Есть еще один очень важный момент, который нужно учитывать при работе с сессиями — объявлять сессию, т. е. использовать функцию `start_session()` надо в каждом скрипте, в котором необходима сессия и ее данные. Рассмотрим это на примере. Создайте файл first_prog.php следующего содержания:

```
<?php
session_start();
$_SESSION['dat']='a';
?>
<a href='view.php'>Распечатать значение элемента dat сессии</a>
```

Также необходимо создать второй файл с именем view.php:

```
<?php
echo $_SESSION['dat'];
?>
```

Теперь запустите first_prog.php и нажмите на ссылку, в результате вы увидите следующее — рис. 9.36.

Как видите, выполнение view.php вызвало ошибку, которая возникла из-за того, что в строке **2** происходит обращение к неизвестной программе переменной, но

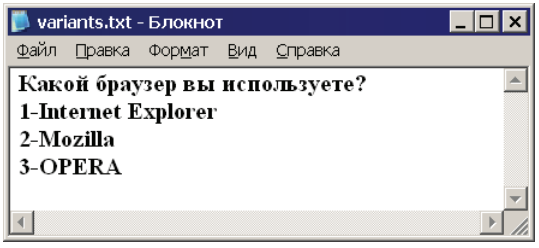


Рис. 9.36. Пример неправильного использования сессии

в этой строке содержится только `echo $_SESSION['dat'];`. Таким образом, перед нами явная ситуация потери сессии при переходе к другому скрипту. Теперь измените view.php следующим образом:

```
<?php
session_start();
echo $_SESSION['dat'];
?>
```

Параметры, влияющие на работу сессии, рассмотрены в приложении 1.

И запустите first_prog.php еще раз, далее опять нажмите на ссылку, вы увидите, что сессия сохранилась, и значение элемента `dat` без ошибок вывелось на экран.

Для того чтобы защитить админку гостевой книги, разработанную ранее, от несанкционированного доступа, необходимо сделать следующее:

В index.php добавьте в самое начало, сразу после `<?php`, код:

```
session_start();
if(isset($_SESSION['autorized']))
{
    require_once("list_message.php");
    exit;
}
```

В index.php перед строкой

```
require_once("list_message.php");
```

добавьте код:

```
$_SESSION['autorized']=true;
```

В list_message.php добавьте в самое начало, сразу после `<?php`, код:

```
if (!isset($_SESSION['autorized'])) exit;
```

В delete_message.php добавьте в самое начало, сразу после `<?php`, код:

```
session_start();
if (!isset($_SESSION ['autorized'])) exit;
```

В delete_message.php замените строку:

```
header("location:list_message.php");
```

на

```
header("location:index.php");
```

9.3. Голосование

Голосование — незаменимый способ обратной связи, оно позволяет узнавать мнение посетителей сайта по тому или иному вопросу. Например, создав голосование "Нравится ли Вам сайт?", вы сможете узнать, как пользователи оценивают ваше творение. Голосование очень действенно. Оно не отнимает много времени у пользователя, и поэтому ему не составляет труда отдать свой голос, что он часто и делает, даже если оказался на вашем сайте совершенно случайно.

Для начала создайте виртуальный хост **opros.ru**, и в нем папку www, далее перезапустите Денвер — только после этого новый хост станет доступен. Все файлы, разработанные в данном разделе, необходимо сохранять в opros.ru\www.

9.3.1. Приступаем к работе

Создайте файл variants.txt, который будет хранить информацию по голосованию, его первая строка будет содержать тему голосования, каждая последующая будет характеризовать вариант ответа. Заполните variants.txt аналогично рис. 9.37.

Очень важно после ввода последнего варианта не делать перевода строки (не нажимать клавишу <Enter>), иначе тем самым вы создадите пустой вариант.

Теперь создайте файл result.txt, который будет предназначен для хранения информации о количестве проголосовавших за тот или иной вариант. В его первой строке будет находиться информация о предназначении файла в виде текста `"Результаты голосования"`. Начиная со следующей строки, для каждого варианта голосования в results.txt будет храниться число проголосовавших, т. е. для первого

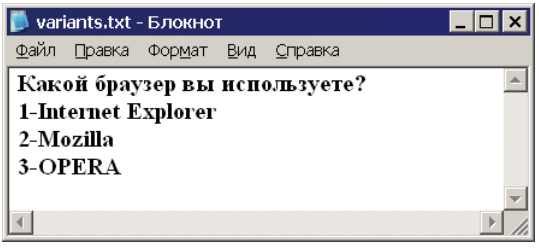


Рис. 9.37. Файл variants.txt, содержащий вопрос и 3 варианта ответа

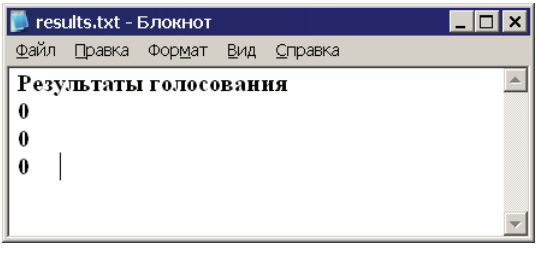


Рис. 9.38. Файл results.txt, содержащий результаты голосования

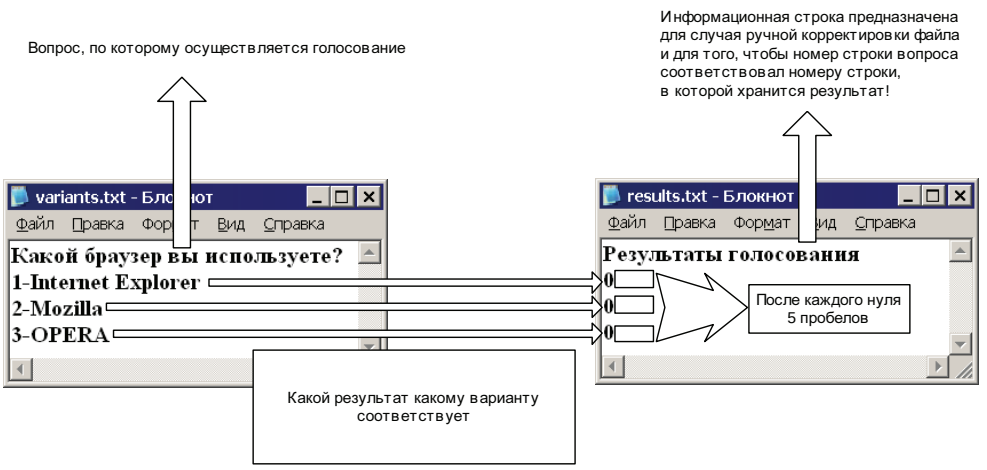


Рис. 9.39. Схема взаимосвязи между файлами variants.txt и results.txt

варианта во второй строке, для второго в третьей и т. д. Заполните result.txt аналогично рис. 9.38, после каждого 0 добавьте по 5 пробелов, зачем это нужно — я объясню чуть позже.

На рис. 9.39 изображена взаимосвязь между файлами variants.txt и results.txt.

Приступаем к кодингу (от англ. "coding"). Создайте файл index.php следующего содержания — листинг 9.11.

Листинг 9.11. Файл index.php

```
<?php
//Читаем файл с вопросом и вариантами ответов
$variants = file('variants.txt');
            //Текущий элемент массива $variants
            $current_variants=0;
//Читаем файл с результатами голосования
$results = file('results.txt');
           //Текущий элемент массива $results равен 1, потому что
           //0 элемент - это фраза "Результаты голосования"
            $current_results=1;
            //Всего проголосовало человек
            $itog=0;
```

```
13  ?>
14  <form action="add_golos.php" method="post">
15  <?php
16  //Выводим тему голосования
17  echo $variants[$current_variants]."<BR>";
18  //Увеличиваем текущий элемент массива $variants на 1
19  $current_variants++;
20
21  //Пока есть варианты для голосования, выводим их
22  while (count($variants)!=$current_variants)
23  {
24    //Выводим вариант голосования в виде переключателя
25*   echo '<input name="first" type=radio
    value="'.$current_variants.'">'.$variants[$current_variants];
26    //также выводим результаты для этого варианта,
27    //он будет заключен в скобки
28    echo ' ('.$results[$current_results].')';
29    //Прибавляем к общему числу проголосовавших значение
30    //по текущему варианту
31    $itog=$itog+$results[$current_results];
32    //Перевод строки
33    echo '<BR>';
34    //Увеличиваем текущий элемент массива $variants на 1
35    $current_variants++;
36    //Увеличиваем текущий элемент массива $results на 1
37    $current_results++;
38  }//end - while
39
40  //Выводим общее число проголосовавших
41  echo '<BR> Всего проголосовало: '.$itog.' человек <BR>';
42  ?>
43  <input name='add_golos' type="submit" value="Проголосовать">
44  </form>
```

В самом начале программы — в строках **3** и **7** происходит создание массива вопросов с именем `$variants` и массива результатов с именем `$results`. Это действие осуществляется функцией `file()`, которая полностью читает содержимое файлов variants.txt и results.txt в описанные массивы. Сразу после этого в переменные

`$current_variants` и `$current_results` устанавливаются значения текущих элементов созданных массивов. В дальнейшем можно будет писать следующим образом:

- `$variants[$current_variants]` — и обратиться к нужному элементу массива `$variants`;
- `$results[$current_results]` — и обратиться к нужному элементу массива `$results`.

Переменная `$current_variant` объявляется равной нулю, чтобы можно было обратиться к элементам массива `$variant`, начиная с первого, т. к. там будет храниться тема голосования. Переменная `$current_result` специально объявляется равной 1, потому что в массиве `$current_result` нас будут интересовать все элементы, кроме нулевого, содержащего текст "`Результаты голосования`".

В строке **14** объявляется `POST`-форма, обработчиком которой будет программа add_golos.php . В строке **17** выводится тема голосования, а в строке **19** происходит увеличение значения переменной `$current_variants` на 1 (или, другими словами, осуществляется переход к следующему элементу массива `$variants`). Далее с помощью конструкции `while` организуется цикл (строки с **22** по **38**), в котором происходит вывод всех вариантов голосования как переключателей. Цикл работает по следующему алгоритму:

```
Пока (количество элементов массива $variants не равно
$variants_current)
{
    вывести очередной вариант голосования
    и количество уже отданных за него голосов
}
```

В условии цикла используется функция `count()`, которая, как вы знаете, возвращает количество элементов в массиве. Таким образом, `count($variants)` вернет количество элементов в массиве `$variants`. Далее это значение сравнивается с номером текущего элемента массива (строка **22**), если они не равны, то условие будет верным (равно `TRUE`), и тело цикла выполнится. В условии цикла используется отрицание, которое стоит перед единственным знаком равно. Это не ошибка, просто чтобы реализовать отрицание в выражении, в котором происходит сравнение между собой значений левой и правой части, необходимо один знак равно заменить на восклицательный знак.

В теле цикла создается элемент "переключатель", это происходит следующим образом:

```
echo '<input name="first" type=radio
value="'.$current_variants.'">'.$variants[$current_variants];
```

То есть в качестве параметра `value` для переключателя будет выступать номер элемента массива `$variants` (или, другими словами, номер варианта ответа), значение которого выводится в данный момент. Таким образом, несмотря на то, что нумерация массива начинается с нуля, порядковый номер варианта голосования равен номеру элемента массива, в котором он расположен.

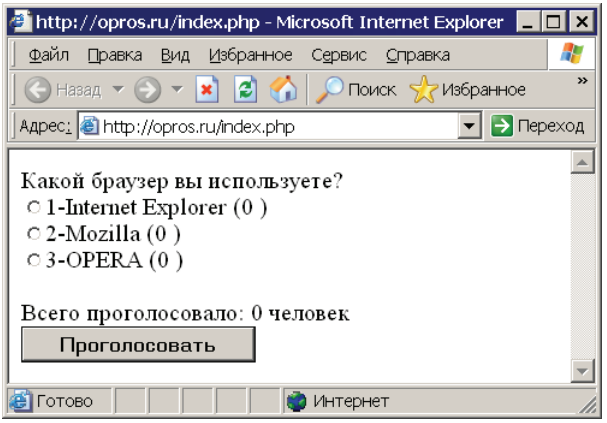


Рис. 9.40. Форма для голосования

Продолжаем рассматривать тело цикла. В строке **28** используется запись

```
echo ' ('.$results[$current_results].')';
```

предназначенная для вывода количества проголосовавших за текущий вариант (используется массив `$results`). Далее это значение прибавляется к переменной `$itog`, таким образом в конце выполнения цикла в этой переменной будет храниться общее число всех голосов.

В строках **35** и **37** происходит переход к следующему элементу для массивов `$variants` и `$results` (прибавление по 1 к переменным, в которых хранятся текущие элементы массивов). Далее тело цикла заканчивается и при выполнении программы будет произведен возврат к условию (строка **22**). Если оно верно, то цикл будет выполнен еще раз и так до тех пор, пока не будут выведены все варианты голосования (или, другими словами, цикл не дойдет до последнего элемента массива `$variants`).

В строке **41** выводится общее число проголосовавших с помощью следующего кода:

```
echo '<BR> Всего проголосовало: '.$itog.' человек <BR>';
```

Ну и затем, после окончания кода на PHP, в строке **43** создается кнопка **Проголосовать**.

Если вы сейчас наберете в браузере **opros.ru**, то увидите следующий результат — рис. 9.40.

Теперь создайте файл add_golos.php, который будет осуществлять прием голоса пользователя за выбранный им вариант (листинг 9.12).

Листинг 9.12. Файл add_golos.php

```
1  <?php
2  //Проверка на то, что скрипт запущен как обработчик
3  if (isset($_POST['add_golos']))
4  {
```

```
5    //Проверка на то, что вариант был выбран
6    if (isset($_POST['first']))
7    {
8       //Открываем файл результатов в бинарном режиме
9       $f=fopen("results.txt","r+b") or die("Не могу открыть файл");
10      //Блокируем файл, чтобы никто не мог к нему обратиться,
11      //пока мы с ним не закончим работу
12      flock($f,2);
13      //Читаем первую строку - "Результаты голосования"
14      $stroka=fgets($f);
15      //Количество прочитанных вариантов голосования из results.txt
16      $i=0;
17      //Читаем результаты для очередного варианта,
18      //пока не дойдем до варианта, за который отдал голос посетитель
19      while ($_POST['first']!=$i)
20      {
21        //Запоминаем текущую позицию указателя
22        $position=ftell($f);
23        //Читаем строку с результатами
24        $stroka=fgets($f);
25        //Увеличиваем количество прочитанных строк на 1
26        $i++;
27      }
28      //Переводим указатель в последнюю запомненную позицию
29      fseek($f,$position,SEEK_SET);
30      //Увеличиваем количество проголосовавших на 1
31      $new_value=$stroka+1;
32      //Записываем новое значение
33      fputs($f,$new_value);
34      //Снимаем блокировку
35      flock($f,3);
36      //Закрываем файл, теперь с ним могут работать другие
37      fclose($f);
38   }
39 }
40 //Возвращаем пользователя назад к форме голосования,
41 //теперь он увидит, что его голос принят
42 header('location:index.php');
43 ?>
```

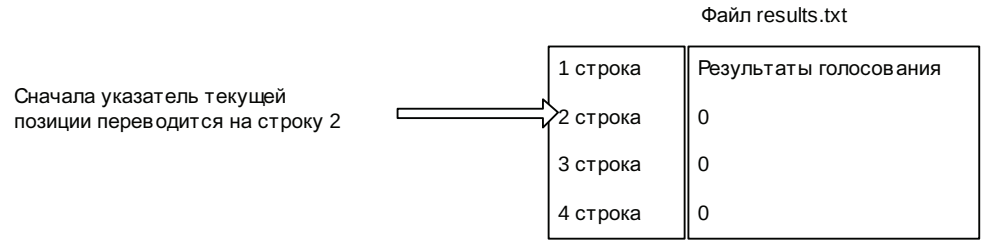


Рис. 9.41. Шаг 1 — перевод указателя текущей позиции на вторую строку

Рис. 9.42. Шаг 2 — поиск результата для варианта, выбранного пользователем

Сначала осуществляется проверка, каким образом запущен скрипт, если как обработчик формы для голосования, то он будет работать, иначе сразу выполнится строка **42**, которая вернет пользователя назад к форме для голосования. Далее в строке **6** проверяется, выбрал ли пользователь вариант для голосования, если да, то будет доступен элемент `first` массива `$_POST`, иначе обрабатывать будет нечего и будет осуществлен переход на строку **42**.

В строке **9** происходит открытие файла результатов results.txt в бинарном режиме, потому что он более корректно работает при записи информации. Далее файл блокируется и происходит чтение его первой строки, в которой содержится фраза `"Результаты голосования"`. Это нужно, чтобы просто перевести указатель текущей позиции на одну строку вперед, т. к. сами результаты начинаются только со второй строки (рис. 9.41).

Далее логика программы строится следующим образом:

1. В results.txt ищется результат для варианта голосования, который выбрал пользователь (рис. 9.42).

Новое значение, увеличенное на 1, записывается поверх старого

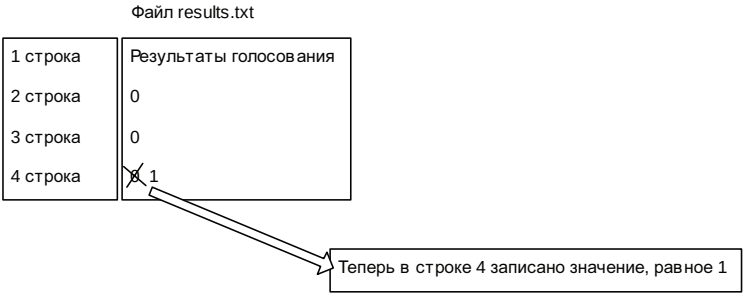


Рис. 9.43. Шаг 4 — запись нового значения в файл results.txt поверх старого

2. Затем этот результат увеличивается на единицу.

3. В result.txt происходит замена старого результата на новый, увеличенный на 1 (рис. 9.43).

В строке **16** создается переменная `$i` со значением 0, в ней будет храниться номер текущего варианта голосования, с которым происходит работа в данный момент. А в строке **19** организуется цикл, который последовательно читает по одной строке из файла результатов (другими словами, самого результата) и сохраняет ее значение в `$stroka`, так вот когда значение `$i` будет равно `$_POST['first']`, соответственно результат для выбранного пользователем варианта голосования будет найден, и его значение будет содержаться в `$stroka`. После каждого прочтения строки из results.txt значение переменной `$i` увеличивается на 1.

Можно сказать, что весь цикл работает следующим образом:

```
Пока не найдена строка с результатом для варианта, за который
голосует пользователь
{
  Запоминаем позицию указателя
  Читаем очередную строку результата
}
```

В строке **22** используется новая функция `ftell()`, которая запоминает текущее положение указателя файла, в качестве параметра ей передается файловый указатель. Использование данной функции вызвано тем, что после того как был прочитан нужный результат, указатель будет сдвинут относительно него, и при

попытке сохранить новое значение будет затерта абсолютно другая строка.

Переход в файле на нужную позицию осуществля-ется с помощью функции `fseek()`, ее синтаксис:

```
fseek(файловый_указатель, позиция,
дополнительный_параметр)
```

Данная функция осуществляет перемещение указателя текущей позиции для файла, файловый указатель которого передан в качестве первого параметра на позицию, переданную в качестве второго параметра. Третий параметр характеризует, относительно какого места файла будет осуществлено перемещение, для него возможны следующие значения:

- `SEEK_SET` — относительно начала файла;
- `SEEK_CUR` — относительно текущей позиции;
- `SEEK_END` — относительно конца файла.

Теперь ничто не мешает увеличить число проголосовавших для выбранного пользователем варианта на 1, т. е. принять голос и сохранить новое значение поверх старого. Помните, в начале раздела было сказано, что в файле результатов после каждого результата нужно добавить по 5 пробелов? Дело в том, что запись будет происходить в режиме замены. Таким образом, если бы лишних пробелов оставлено не было, то после того как число знаков в одном из результатов увеличилось до 2, например, переход с 9 до 10, то 0 затер бы символ (в случае Windows один из символов) перевода строки и нижестоящая строчка поднялась бы на одну позицию вверх (для Windows это тоже будет справедливым).

9.3.2. Использование cookie

Главный недостаток у только что реализованной системы голосования — это то, что ее можно запросто "накрутить". Это делается следующим образом:

Наберите в браузере **opros.ru**, будет запущен файл index.php.

Теперь выберите любой вариант и нажмите кнопку **Голосовать**, после этого проголосуйте аналогичным способом еще несколько раз.

Как видите, таким образом можно голосовать за понравившийся вам вариант любое количество раз. Частично решить эту проблему поможет cookie. Почему не полностью? Дело в том, что пользователь может отключить cookie в браузере, например, в Internet Explorer, для этого достаточно выбрать пункт **Сервис | Свойства обозревателя**, затем перейти на вкладку **Конфиденциальность**, нажать кнопку **Дополнительно** и в появившемся окне **Дополнительные параметры конфиденциальности** настроить параметры работы с cookie (рис. 9.44).

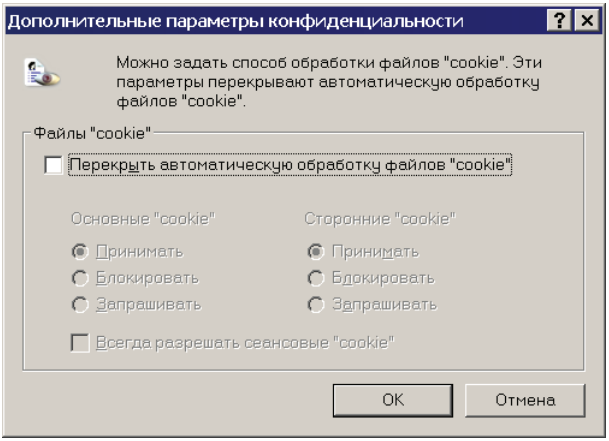


Рис. 9.44. Изменение настроек работы с cookie в браузере **Internet Explorer**

Замечание

Когда вы устанавливаете на каком-то сайте флаг "сохранить пароль" и при следующем посещении вы не вводите его, то можете быть уверены, что эта информация просто храниться у вас на компьютере в cookie и автоматически отсылается серверу при обращении к нему.

Cookie предоставляет механизм, посредством которого скрипт может сохранять данные (переменные, массивы) на компьютере пользователя. Причем для различных Web-сайтов данные хранятся отдельно друг от друга, т. е. браузеру известно, какому серверу какой cookie принадлежит. После сохранения cookie повторное обращение к серверу, который сделал это, приведет к тому, что сохраненная в cookie информация будет передана серверу.

Для того чтобы создать cookie, используется функция `setcookie()`, ее синтаксис:

```
setcookie(имя_cookie,
значение_cookie, время_жизни_cookie);
```

Например, следующая запись:

```
setcookie('my_cookie','Понедельник',time()+60*60*24*3)
```

приведет к созданию cookie с именем `my_cookie`, значением `Понедельник` и временем жизни, равным 3 дням, по истечении которого cookie будет удален из системы. Время жизни cookie устанавливается в секундах по следующей формуле:

```
Текущее_время + время_жизни_в_секундах
```

где `Текущее_время` — результат выполнения функции `time()`, которая возвращает количество секунд, прошедших с 1 января 1970 года.

Замечание

Можно сказать, что cookie похожи на сессии, только первые хранят информацию на компьютере пользователя, и срок жизни этих данных может быть намного продолжительнее, чем у сессий.

Время_жизни_в_секундах — для трех дней будет равно 60 (число секунд в одной минуте), умноженному на 60 (число минут в одном часе), умноженному на 24 (количество часов в одном дне) и умноженному на 3 (три дня).

В табл. 9.1 рассмотрим несколько альтернативных вариантов использования функции setcookie().

Таблица 9.1. *Использование функции* `setcookie()`

Вариант использования	**Синтаксис**	**Пример**
Создание cookie на одну сессию, т. е. он будет действовать до того момента, пока пользователь не закроет браузер	`setcookie('имя_cookie', Значение)`	`setcookie('my_cookie', 'Пользователь очень капризен')`
Изменение существующего cookie	`setcookie('имя_существующего_cookie', новое_значение, новое_время_жизни)`	`setcookie('my_cookie', 'Пользователь совсем не капризен', time()+60*60)`
Удаление существующего cookie	`setcookie('имя_существующего_cookie')` или `setcookie('имя_существующего_cookie','')`	`setcookie('my_cookie')` или `setcookie('my_cookie','')`

Все cookie, пришедшие от пользователя, хранятся в массиве $_COOKIE. Поэтому, для того чтобы получить нужный cookie, достаточно указать его имя как элемент массива $_COOKIE. Рассмотрим небольшой пример (prog8.php):

```
<?php
if (!isset($_COOKIE['my_cookie']))
{
   setcookie('my_cookie','Пользователь совсем не капризен',
time()+30);
   echo "В вашей системе установлен cookie";
}
echo "Значение my_cookie равно - ".$_COOKIE['my_cookie'];
?>
```

Результат его выполнения представлен на рис. 9.45.

Как видите, скрипт выполнился с ошибкой. Все дело в том, что после установки cookie он хранится на компьютере пользователя, и сервер ничего о нем не знает до того момента, пока пользователь заново не обратится к серверу, который его установил, именно после этого массив `$_COOKIE` будет заполнен. Измените prog8.php следующим образом:

```
<?php
if (!isset($_COOKIE['my_cookie']))
{
   setcookie('my_cookie','Пользователь совсем не капризен',
time()+30);
   echo "В вашей системе установлен cookie";
}
else echo "Значение my_cookie равно - ".$_COOKIE['my_cookie'];
?>
```

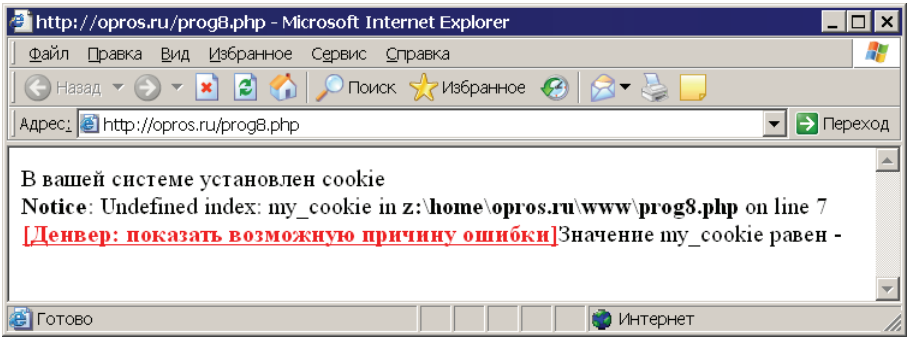


Рис. 9.45. Пример неправильной работы с cookie

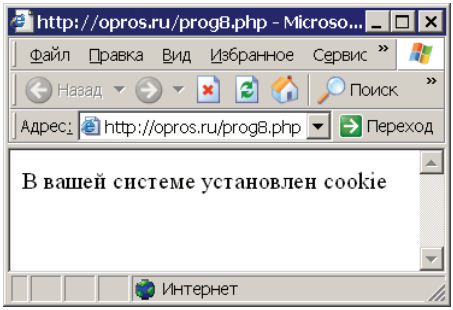


Рис. 9.46. Установка cookie

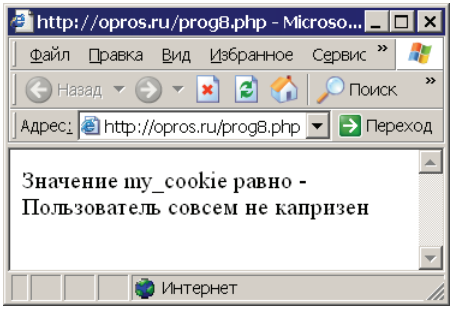


Рис. 9.47. Вывод значения cookie на экран

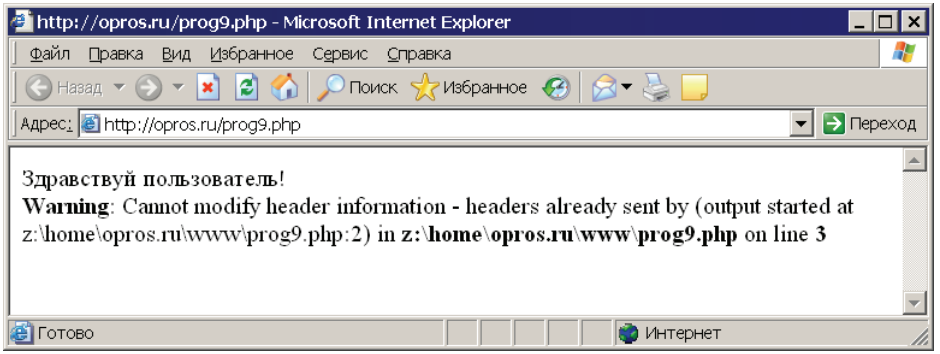


Рис. 9.48. Пример неправильного использования функции `setcookie()`

Теперь закройте браузер, подождите 30 с, именно такое время жизни у cookie с именем `my_cookie`, и запустите опять prog8.php, вы увидите надпись **В вашей системе установлен cookie**, это произошло потому, что по истечении установленного времени в 30 с cookie был удален.

Cookie необходимо устанавливать до команд вывода, рассмотрим следующий пример (prog9.php):

```
<?php
echo "Здравствуй пользователь!";
setcookie('my_cookie','Пользователь совсем не капризен', time()+30);
?>
```

Результатом его выполнения будет ошибка, т. к. перед функцией `setcookie` используется команда вывода `echo` (рис. 9.48).

Поговорим о том, почему так происходит. Дело в том, что использование функции `setcookie()` вызывает процесс отправки браузеру пользователя заголовка с именем `Set-Cookie`, а как вы знаете, заголовки должны передаваться до команд вывода.

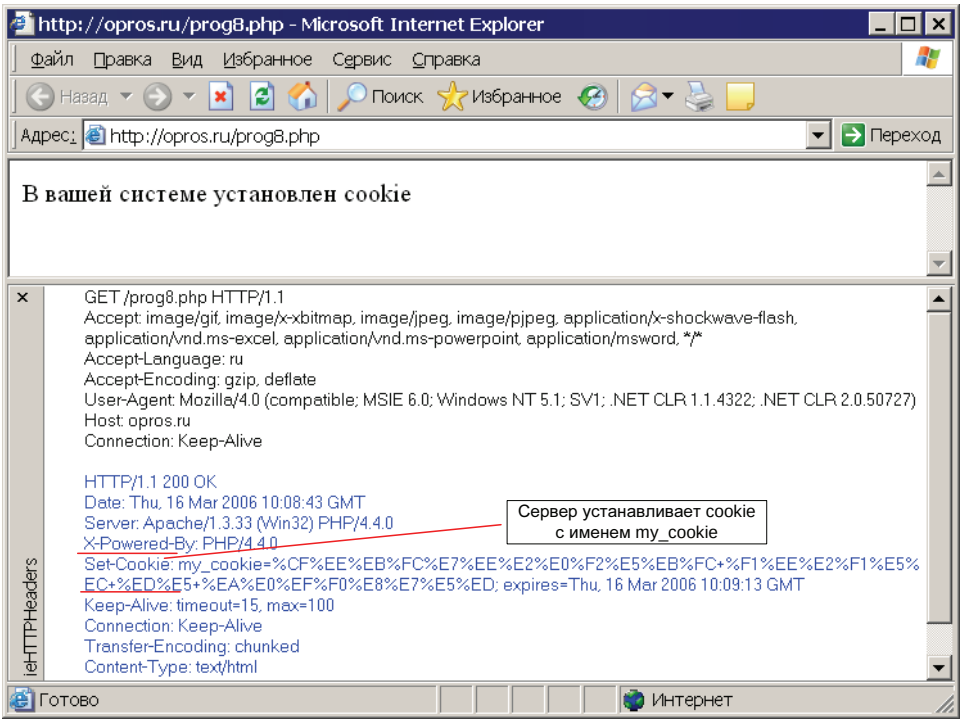


Рис. 9.49. Сервер устанавливает cookie с именем `my_cookie` с помощью специального заголовка `Set-Cookie`

Запустите Internet Explorer с включенным инструментом просмотра скрытой информации **ieHTTPHeaders** и выполните программу prog8.php. Вы увидите следующее — рис. 9.49.

Теперь нажмите кнопку **Обновить**, и вы сможете наблюдать следующий результат — рис. 9.50.

Теперь вы обладаете достаточными знаниями, для того чтобы встроить в систему голосования простейшую защиту от накрутки. Для этого в файл add_golos.php сразу после условия `if (isset($_POST['first']))` и открывающейся фигурной скобки добавьте следующий код:

```
//Если пользователь еще не голосовал,
//то устанавливаем cookie со временем жизни 30 дней,
//иначе не принимаем голос и переводим пользователя на форму голосования
    if (!(isset($_COOKIE['golos'])))
```

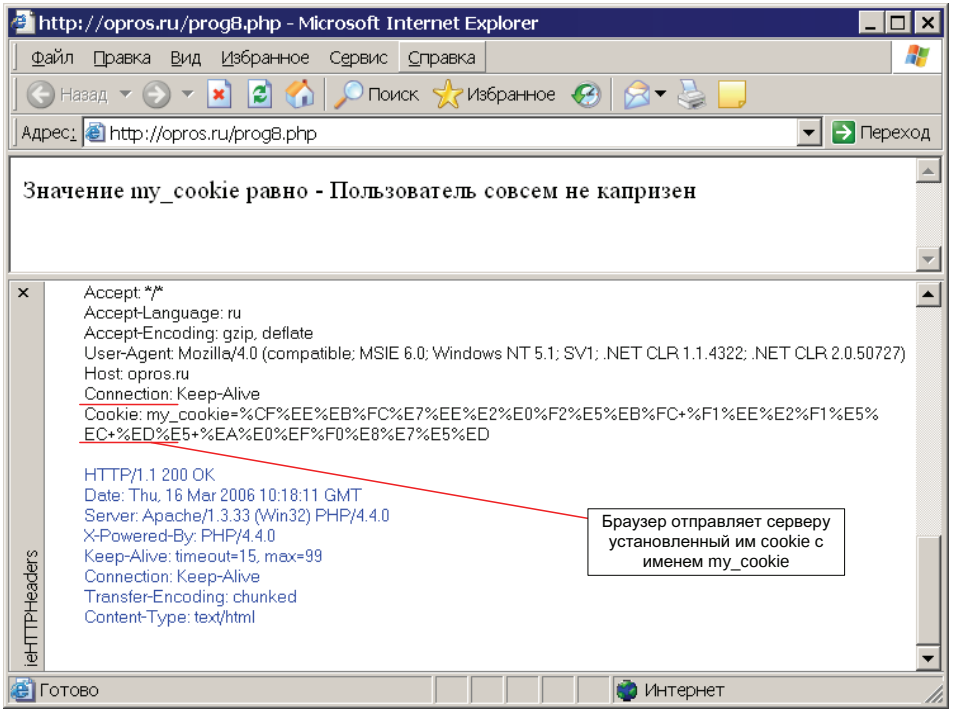


Рис. 9.50. Браузер отправляет серверу установленный им cookie с именем `my_cookie`

```
        setcookie('golos','yes',time()+60*60*24*30);
    else
    {
        header('location:index.php');
        exit;
    }
```

После этого при повторном голосовании ваш голос принят не будет.

Вы можете посмотреть cookie, установленные на вашем компьютере, например, Internet Explorer хранит их в папке Document and Settings*Имя учетной записи пользователя*\Cookies (рис. 9.51).

Конечно, здесь не вся информация интуитивно понятна, но, например, значение `yes` cookie с именем `golos` отчетливо можно увидеть. Вы удивитесь, но если удалить файл, в котором хранится cookie, то система голосования будет считать, что мы не голосовали, а это опять открывает доступ к накручиванию.

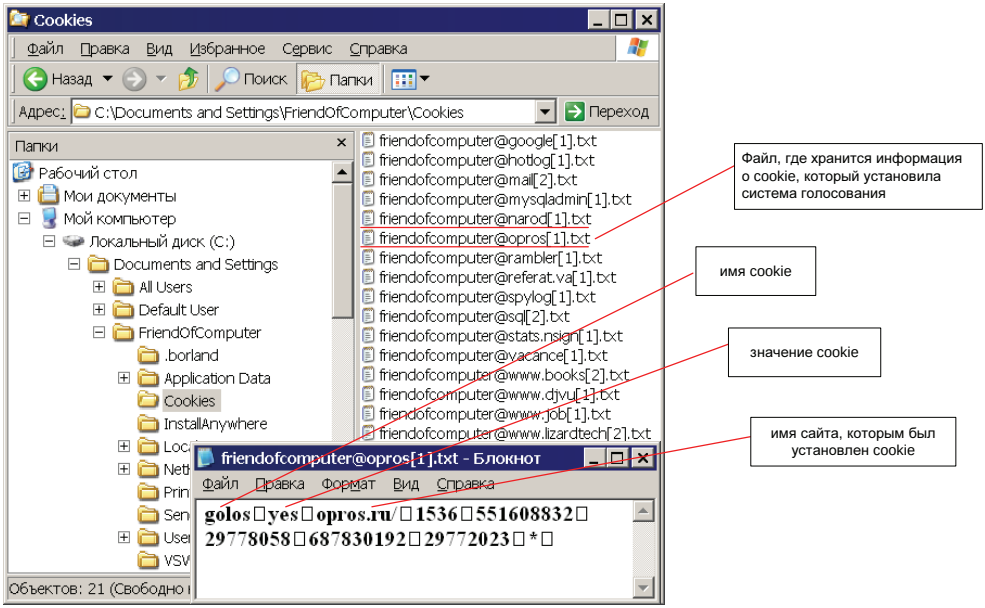


Рис. 9.51. Местонахождение cookie, установленного системой голосования

Замечание

Один из самых надежных вариантов защиты от накручивания заключается в ведении авторизации, т. е. чтобы только авторизированные пользователи могли голосовать.

9.3.3. Админка для системы голосования

Для более удобной работы с системой голосования разработаем панель администратора, которая будет позволять задавать новые темы для голосования (рис. 9.52).

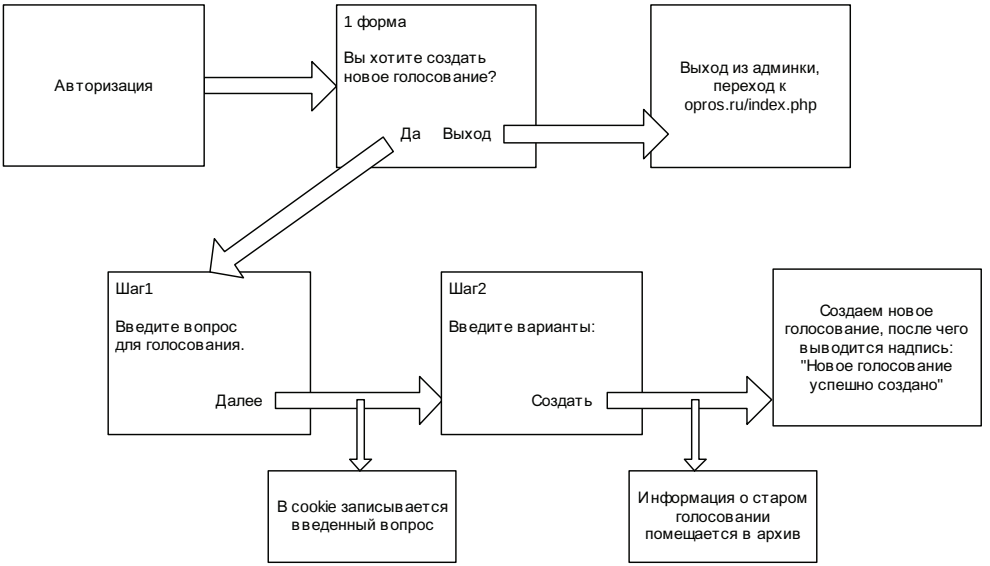


Рис. 9.52. Схема функционирования админки для системы голосования

То есть, чтобы попасть в админку, пользователю нужно будет авторизоваться. Модуль авторизации будет практически тот же самый, что был разработан для гостевой книги. Создайте папку adminka, в ней файл pas.txt, сохраните в файле хеш пароля администратора (аналогично действиям для гостевой книги). Теперь создайте в папке adminka файл index.php (листинг 9.13).

Листинг 9.13. Файл adminka/index.php

```
<?php
//Запускаем сессию
session_start();
//Если нажата кнопка "Войти"
if (isset($_POST['enter']))
{
    //Читаем хеш из файла pas.txt
    $s=file('pas.txt');
    //Получаем хеш от пароля, введенного пользователем
```

```
    $hash=md5($_POST['passwd']);
    //Сравниваем хеш пароля и логин с теми,
    //которые ввел пользователь
    if (($s[0]==$hash) and ($_POST['login']=='admin'))
    {
       //Записываем в сессию информацию о том,
       // что пользователь успешно авторизовался
       $_SESSION['autorized']=true;
       //Подключаем модуль, выводящий меню пользователю
       include("menu.php");
       //Больше делать ничего не надо, выходим
       exit;
    }
    else
    {
      //Если пароль или имя пользователя неправильные,
      //выводим информацию об этом
      echo "Логин или пароль неверные ";
      echo "<a href='index.php'>Назад</a>";
      exit;
    }
}//end - if (isset($_POST['enter']))
?>
<!--форма авторизации-->
<FORM action='' method='POST'>
Логин:<input type='text' name='login'>
<BR><BR>
Пароль:<input type='password' name='passwd'>
<BR><BR>
<input type='submit' name='enter' value='Войти'>
</FORM>
```

Если пользователь прошел авторизацию, то подключается модуль menu.php, который выводит меню админки, состоящее из двух кнопок: **Создать новое голосование** и **Выход**. Создайте в папке adminka файл menu.php (листинг 9.14).

Если пользователь нажмет кнопку **Выход**, то произойдет выход из админки, и далее он будет перенаправлен на **opros.ru**. Создайте файл logout.php (листинг 9.15).

Листинг 9.14. Файл adminka/menu.php

```
<?php
//Если этот скрипт пытаются запустить в обход авторизации,
//то осуществляем выход
if (!isset($_SESSION['autorized'])) exit;
?>
<!--Выводим меню, состоящее из двух кнопок-->
<!--Каждая кнопка находится в своей форме-->
<H1>Панель администрирования системы голосования</H1>
<form name="MenuForm" action="master.php?step=1" method="POST">
<input type="submit" value="Создать новое голосование">
</form>
<form name="MenuForm2" action="logout.php" method="POST">
<input type="submit" value="      Выход            ">
</form>
```

Листинг 9.15. Файл adminka/logout.php

```
<?php
//Уничтожаем сессию
session_start();
session_unset();
session_destroy();
//Перенаправляем пользователя на  opros.ru
header("location: ../index.php");
?>
```

Если в меню пользователь нажмет кнопку **Создать новое голосование**, то ему необходимо будет пройти 2 шага:

1. Ввод темы голосования.
2. Ввод вариантов для голосования.

После создания нового голосования информация по старому будет изменена: файлы variants.txt и result.txt будут перемещены в архив, который хранится в каталоге arhiv. В начало этих файлов будет добавлена дата в формате `ддммгггг`, где `дд` — это день, `мм` — месяц, `гггг` — год, которые являются датой переноса этих файлов в архив. Создайте в папке adminka подкаталог arhiv и файл master.php (листинг 9.16).

Листинг 9.16. Файл adminka/master.php

```
<?php
//Запускаем сессию
session_start();
//Если этот скрипт пытаются запустить в обход авторизации,
//то осуществляем выход
if (!isset($_SESSION['autorized'])) exit;

//Если первый шаг мастера, то
if ($_GET['step']==1)
{
   echo "<H1>Панель администрирования системы голосования</H1>";
   echo "<H2>Шаг1: </H2> Введите вопрос для голосования.";
   ?>
   <!--форма ввода вопроса-->
   <form name="Step1" action="master.php?step=2" method="POST">
   <input name="question" type="text" value="">
   <input type="submit" value="Далее>>">
   </form>
   <?php
}

//Если второй шаг мастера, то
if ($_GET['step']==2)
{
   //Сохраняем в cookie вопрос, введенный пользователем
   setcookie('question',$_POST['question']);
   echo "<H1>Панель администрирования системы голосования</H1>";
   echo "<H2>Шаг2: </H2>";
   echo "Введите варианты голосования, максимум 4, <BR>";
   echo "если необходимо меньше, просто оставьте лишние поля
   пустыми.";
   ?>
   <!--форма ввода вариантов голосования-->
   <form name="Step2" action="master.php?step=3" method="POST">
   вариант №1 <input name="variant1" type="text" value=""><BR>
   вариант №2 <input name="variant2" type="text" value=""><BR>
   вариант №3 <input name="variant3" type="text" value=""><BR>
   вариант №4 <input name="variant4" type="text" value=""><BR><BR>
   <input type="submit" value="Далее>>">
   </form>
```

```
<?php
}
//Если третий шаг мастера, то
if ($_GET['step']==3)
{
   //файлы variants.txt и results.txt перемещаются в папку arhiv
   rename("../variants.txt","arhiv/".date('dmY')."variants.txt");
   rename("../results.txt","arhiv/".date('dmY')."results.txt");

   //Создаем новый файл вариантов - variants.txt
   $f=fopen("../variants.txt","at");
   //Блокируем файл
   flock($f,2);
     //Записываем вопрос для голосования
     fputs($f,$_COOKIE['question']);
     //Записываем вариант №1, если он задан
     if ($_POST['variant1']!='') fputs($f,"\n".$_POST['variant1']);
     //Записываем вариант №2, если он задан
     if ($_POST['variant2']!='') fputs($f,"\n".$_POST['variant2']);
     //Записываем вариант №3, если он задан
     if ($_POST['variant3']!='') fputs($f,"\n".$_POST['variant3']);
     //Записываем вариант №4, если он задан
     if ($_POST['variant4']!='') fputs($f,"\n".$_POST['variant4']);
   //Снимаем блокировку с файла variants.txt
   flock($f,3);
   //Закрываем файл variantx.txt
   fclose($f);

   //Создаем новый файл результатов - results.txt
   $f=fopen("../results.txt","at");
   //Блокируем файл
   flock($f,2);
     //Записываем текст "Результаты голосования"
     fputs($f,"Результаты голосования");
     //Записываем 0 и 5 пробелов для первого варианта,
     //если администратор его ввел
     if ($_POST['variant1']!='') fputs($f,"\n"."0     ");
     //Записываем 0 и 5 пробелов для второго варианта,
     // если администратор его ввел
     if ($_POST['variant2']!='') fputs($f,"\n"."0     ");
     //Записываем 0 и 5 пробелов для третьего варианта,
     // если администратор его ввел
     if ($_POST['variant3']!='') fputs($f,"\n"."0     ");
```

```
83          //Записываем 0 и 5 пробелов для четвертого варианта,
84          // если администратор его ввел
85          if ($_POST['variant4']!='') fputs($f,"\n"."0     ");
86      //Снимаем блокировку с файла results.txt
87      flock($f,3);
88      //Закрываем файл results.txt
89      fclose($f);
90
91      //Выводим сообщение о том, что новое голосование создано
92      echo "<H1>Панель администрирования системы голосования</H1>";
93      echo "Новое голосование успешно создано!<BR>";
94      //ссылка на выход
95      echo "<a href='logout.php'>Выход</a>";
96  }
97  ?>
```

Работа мастера состоит из трех этапов: на первых двух пользователь вводит информацию, на третьем создается новое голосование и пользователю выводится информация об этом. После того как администратор ввел вопрос для нового голосования, он переходит ко второму этапу, где необходимо ввести варианты ответов, при этом сам вопрос сохраняется в cookie с именем `question`. После того как варианты введены и нажата кнопка **Далее**, начинает работу третий этап, при реализации которого была использована новая функция `rename()`, синтаксис ее следующий:

```
rename(имя_файла, новое_имя_файла);
```

Функция перемещает файл, переданный в качестве первого параметра, в соответствии с путем, указанном во втором параметре, например:

```
rename(c:\1.txt, c:\windos\hhh.txt);
```

Осуществит перемещение файла 1.txt, находящегося на диске C: в папку Windows с новым именем hhh.txt.

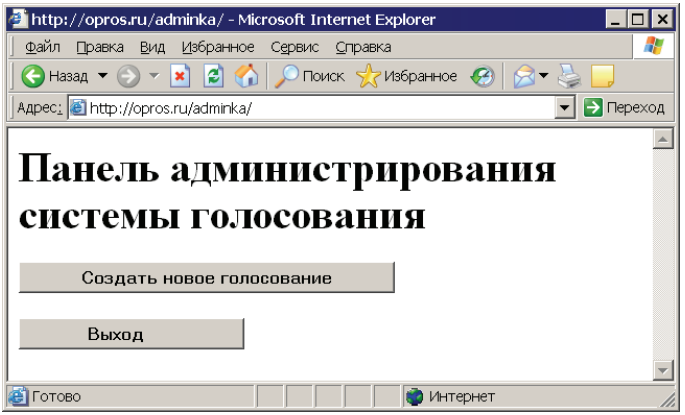


Рис. 9.53. Меню панели администратора

На рис. 9.53—9.56 вы можете увидеть окна разработанного мастера создания нового голосования.

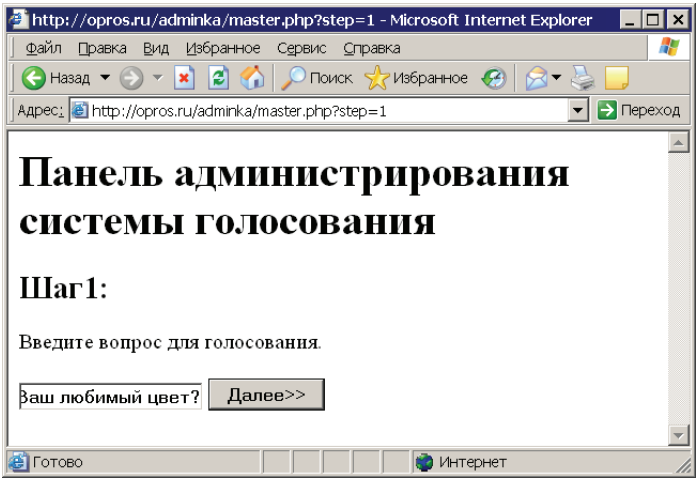


Рис. 9.54. Первый шаг работы мастера: ввод вопроса для голосования

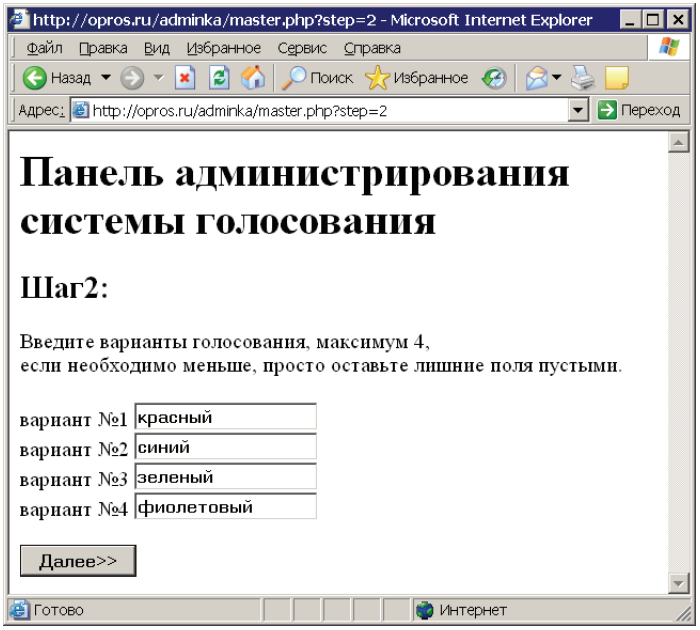


Рис. 9.55. Второй шаг работы мастера: ввод вариантов для голосования

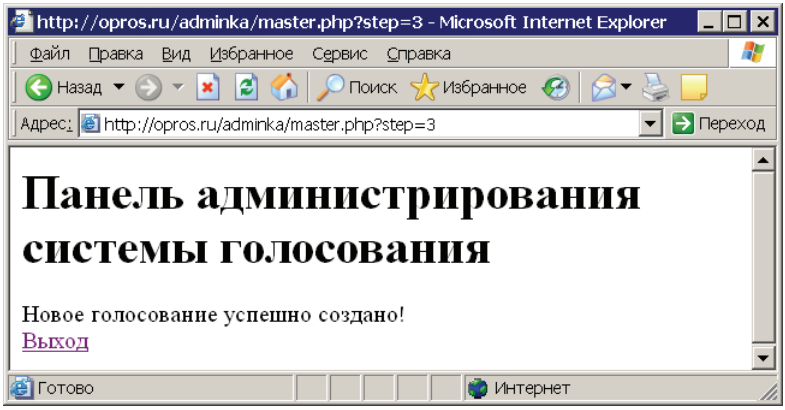


Рис. 9.56.
Заключительный этап работы мастера

9.4. Загрузка файлов

Практически ни один современный сайт не обходится без функции загрузки файлов, в частности рисунков. Но реализация этой возможности обладает некоторыми тонкостями, которые будут рассмотрены в данной части книги. Для работы с примерами этого раздела создайте виртуальный хост **upload.ru**, в нем папку www, далее перезапустите Денвер — только после этого новый хост станет доступен.

9.4.1. Основы

Для реализации загрузки файла предусмотрена специальная форма, она создается следующим образом:

```
<form action="обработчик" method="POST" enctype="multipart/form-data">
специальный элемент
обычная кнопка
</form>
```

Специальный элемент объявляется как:

```
<input type="file" name="имя элемента" >
```

Следующий пример (листинг 9.17) приведет к созданию формы для загрузки файла (рис. 9.57).

Обратите внимание, что нужно создавать `POST`*-форму.*

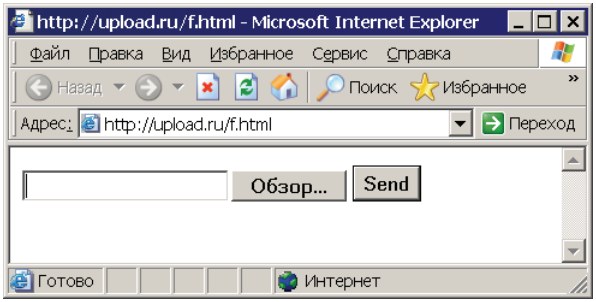


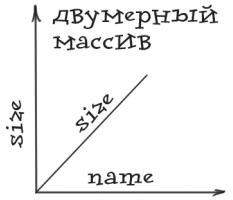
Рис. 9.57. Форма, предназначенная для загрузки файлов

Листинг 9.17. Форма для загрузки файла

```
<form action="obrabotka.php" method="POST" enctype="multipart/form-data">
<input type="file" name="upload">
<input type="submit" value="Send">
</form>
```

Как видите, специальный элемент типа `file` создает поле ввода и кнопку **Обзор**, нажатие которой вызывает стандартное окно выбора файла (рис. 9.58).

После нажатия кнопки **Send** будет произведена отправка выбранного файла на сервер. Как правило, файл отправляется не целиком, а предварительно делится на несколько частей, которые называют пакетами, а затем происходит отправка каждого из них по отдельности. На сервере выполняется обратный процесс — все пакеты собираются в один файл, это возможно благодаря тому, что каждый из них пронумерован, и сервер легко определяет, в каком порядке их соединить. На рис. 9.59 изображена схема, иллюстрирующая описанный процесс.



Вместе с самим файлом на сервер также поступает дополнительная информация о нем, а именно: размер файла, имя файла и тип файла. Эти данные доступны через специальный двумерный массив `$_FILES`. Работа с *двумерным массивом* строится следующим образом:

```
Имя_массива[имя_элемента][имя_элемента2]
```

В данном случае имя первого элемента будет совпадать с именем элемента для выбора файла (`<input type="file" name="имя элемента">`), а имя второго элемента будет характеризовать то значение, которое необходимо получить. Например, чтобы вывести имя файла, присланного на сервер из формы, созданной в листинге 9.17, можно использовать следующий код:

```
echo $_FILES['upload']['name']
```

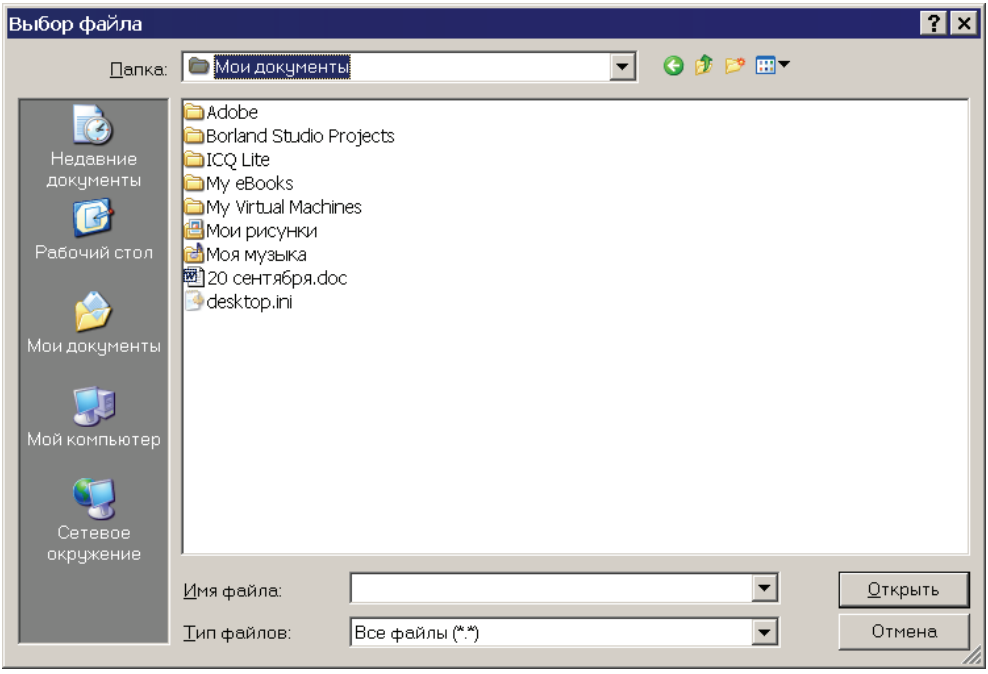


Рис. 9.58. Окно **Выбор файла**, которое появляется после нажатия кнопки **Обзор**

Далее представлена информация о том, какие данные можно получить из массива `$_FILES`, для удобства именем первого элемента массива выбрано `upload`:

- `$_FILES['upload']['name']` — имя файла на компьютере клиента;
- `$_FILES['upload']['type']` — тип файла;
- `$_FILES['upload']['size']` — размер файла в байтах;
- `$_FILES['upload']['tmp_name']` — временное имя, с которым принятый файл был сохранен на сервере;
- `$_FILES['upload']['error']` — если при загрузке файла возникла ошибка, то здесь будет содержаться ее номер.

Давайте рассмотрим пример. Создайте файл upload_form.html следующего содержания:

```
<form action="upload_action.php" method="POST" enctype="multipart/
form-data">
<input type="file" name="upload" >
<input type="submit" value="Send">
</form>
```

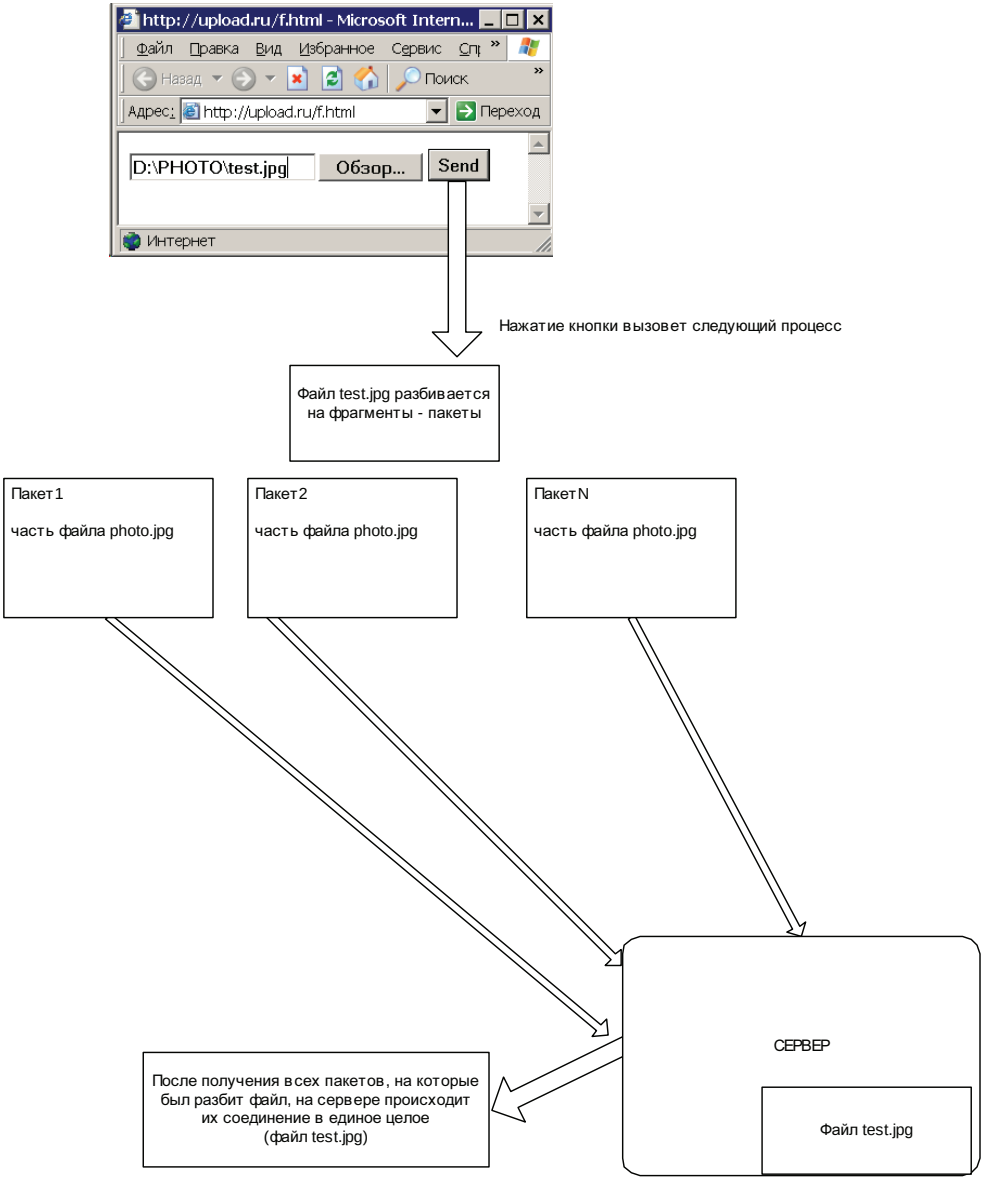


Рис. 9.59. Получение сервером файла, отправленного пользователем через браузер

Теперь создайте файл-обработчик upload_action.php, его содержимое:

```
<pre>
<?php
print_r($_FILES)
?>
</pre>
```

Здесь используется новая функция `print_r()`, которой в качестве параметра передается имя массива, в результате ее выполнения все содержимое массива будет выведено на экран — вместе с именами элементов и их значениями.

Запустите upload_form.html, выберите какой-нибудь файл, например любой рисунок, и нажмите **Send** — в результате вы увидите следующее — рис. 9.60.

То есть у массива `$_FILE` есть элемент `upload`, который в свою очередь является также массивом (об этом говорит запись `=> Array`), состоящим из элементов `name`, `type`, `tmp_name`, `error`, `size`, каждый из которых имеет свое значение.

Если вы нажмете кнопку **Send**, не выбрав файла, то увидите следующее — рис. 9.61.

В данном случае `$_FILES['upload']['size']=0` говорит о том, что ничего не было загружено на сервер.

Замечание

Функцию `print_r()` часто используют в отладочных целях, т. к. она помогает распечатать все содержимое массива с помощью одной строчки кода — не нужно вводить в программу множество операторов `echo` с указанием точного имени каждого элемента.

А теперь предлагаю изменить файл upload_action.php следующим образом (добавить скрытое поле с именем `MAX_FILE_SIZE`):

```
<form action="upload_action.php" method="POST" enctype="multipart/
form-data">
```

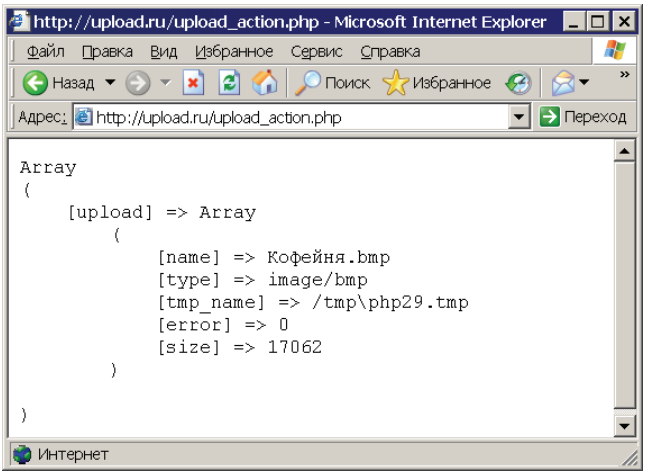


Рис. 9.60. Результат работы функции `print_r()`

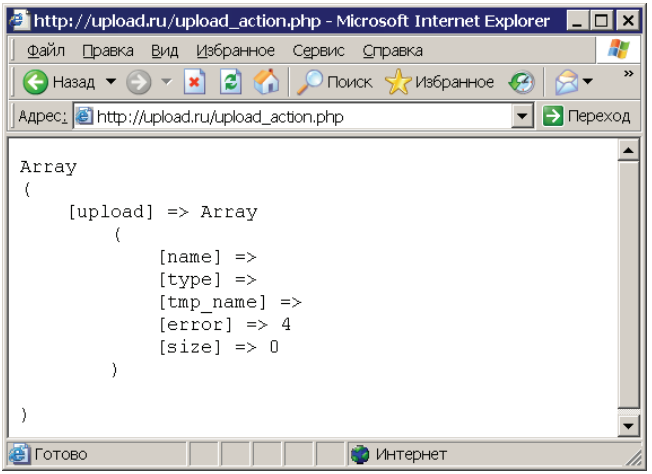


Рис. 9.61. Содержание массива `$_FILES` в случае, когда пользователь не выбрал файл

```
<input type="hidden" name="MAX_FILE_SIZE" value="10000">
<input type="file" name="upload" >
<input type="submit" value="Send">
</form>
```

Скрытое поле `MAX_FILE_SIZE` устанавливает ограничения с помощью параметра `value` на максимально возможный размер закачиваемого файла. Таким образом, сам браузер заблокирует отправку файла, если его размер превышает допустимый. Ограничение указывается в байтах, в данном случае оно равно около 10 Кбайт (1 Кбайт равен 1024 байтам). Давайте проверим это. Запустите измененный upload_form.php и попробуйте загрузить любой файл, размер которого больше 10 Кбайт, вы увидите следующий результат — рис. 9.62.

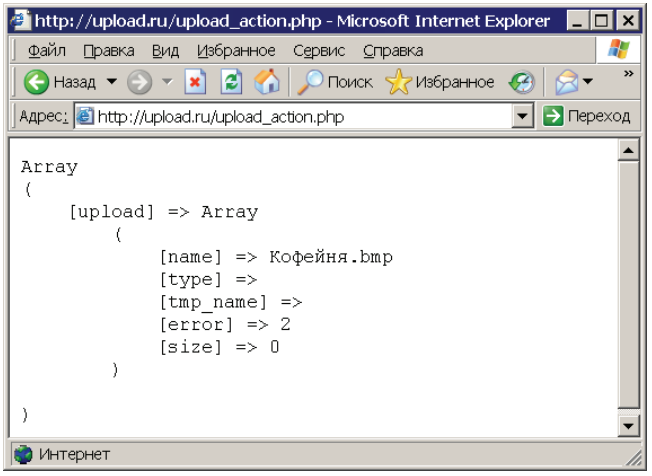


Рис. 9.62. Результат загрузки файла, размер которого превышает установленное ограничение

Ошибка с номером 2, которая была получена в последний раз (рис. 9.62), говорит о том, что браузер заблокировал отправку файла превышающего ограничение, установленное с помощью скрытого поля `MAX_FILE_SIZE`.

Как видите, размер принятого файла равен 0, при загрузке произошла ошибка с кодом 2 (см. элемент `error`), но зато известно имя файла, который хотел загрузить пользователь. Код ошибки помогает понять, что именно послужило причиной неудачи. В случае закачки файла в PHP предусмотрены несколько вариантов ошибок, которые описаны в табл. 9.2, причем каждую из них можно использовать в двух вариантах: в виде числа или в виде константы.

Таблица 9.2. *Описание ошибок, возникающих при загрузке файла на сервер*

Код ошибки	Константа	Описание
0	UPLOAD_ERR_OK	Файл был загружен без ошибок
1	UPLOAD_ERR_INI_SIZE	Размер загружаемого файла превысил ограничение, установленное в php.ini
2	UPLOAD_ERR_FORM_SIZE	Размер загружаемого файла, превысил значение скрытого поля MAX_FILE_SIZE, установленное в форме для загрузки файла
3	UPLOAD_ERR_PARTIAL	Серверу удалось получить загружаемый файл только частично
4	UPLOAD_ERR_NO_FILE	Файл не был загружен

Теперь измените файл action_upload.php (листинг 9.18).

Листинг 9.18. Измененный файл action_upload.php

```
<?php
switch ($_FILES['upload']['error'])
{
case 0:
    echo "Файл был успешно загружен<BR>";
    break;
case 1:
    echo "Размер принимаемого файла превысил ограничение,
установленное в php.ini<BR>";
```

```
9       break;
10      case 2:
11*     echo "Размер загружаемого файла превысил значение, установленное
        в MAX_FILE_SIZE<BR>";
12      break;
13      case 3:
14      echo "Загружаемый файл был получен только частично<BR>";
15      break;
16      case 4:
17      echo "Файл не был загружен<BR>";
18      break;
19      }//end - switch
20      ?>
21      <pre>
22      <?php
23      print_r($_FILES)
24      ?>
25      </pre>
```

В строке со **2** по **18** была добавлена проверка на наличие ошибок, возникших при загрузке файла. В зависимости от ее номера выводится поясняющая надпись, для чего используется специальная конструкция `switch`, ее синтаксис:

```
switch (Что_будем_сравнивать)
{
case Значение1:
       код1
       break;

case Значение2:
       код2
       break;

case ЗначениеN:
       кодN
       break;
}
```

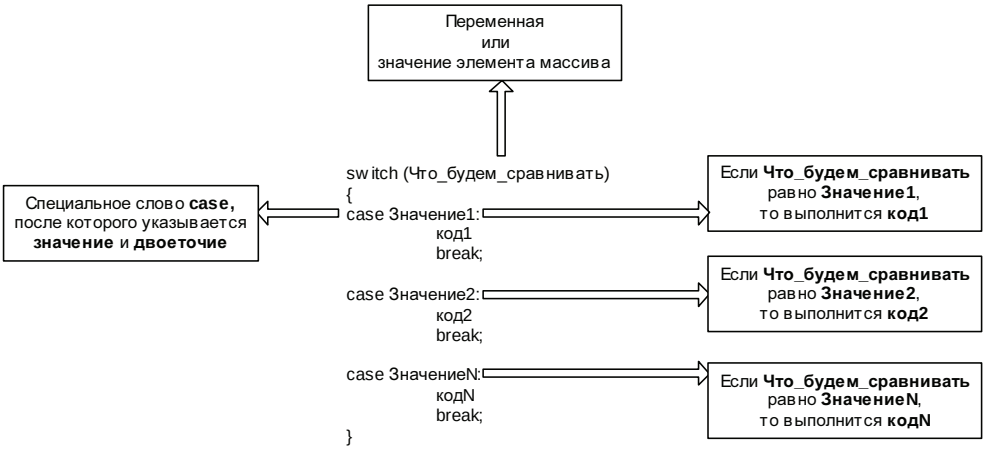


Рис. 9.63. Схема работы конструкции `switch`

Данная конструкция сравнивает переменную, элемент массива или выражение с различными значениями и в случае совпадения выполняет участок кода, принадлежащий значению, сравнение с которым дало положительный результат (рис. 9.63).

Вернемся к программе action_upload.php. Когда ее выполнение дойдет до строки **2**, начнется сравнение `$_FILES['upload']['error']` со значением, равным 0, если они равны, то будет выполнен код:

```
echo "Файл был успешно загружен<BR>";
break;
```

первая строка которого выведет текст, а вторая — break (своеобразный `exit`, только для конструкции) осуществит выход из конструкции `switch`, т. к. дальнейшего смысла искать совпадений нет (рис. 9.64). Если окажется, что `$_FILES['upload']['error']` не равно 0, то далее будет осуществлено сравнение с 1 и т. д.

Замечание

Выражением можно назвать любую строку с использованием переменных, элементов массивов, математических операций или функций. Например: `$s+2` или `$mas['color']`.

А теперь изменим action_upload.php (листинг 9.19).

В данном листинге вместо обычных числовых значений используются константы. Логика работа программы от этого нисколько не изменится, но зато код станет более понятным.

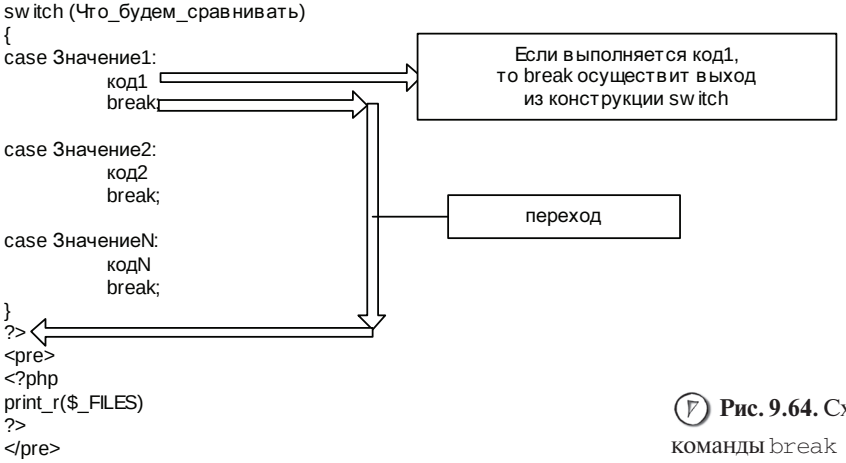


Рис. 9.64. Схема работы команды `break`

Листинг 9.19. Еще раз измененный файл action_upload.php

```
<?php
switch ($_FILES['upload']['error'])
{
case UPLOAD_ERR_OK:
    echo "Файл был успешно загружен<BR>";
    break;
case UPLOAD_ERR_INI_SIZE:
    echo "Размер принимаемого файла превысил ограничение,
установленное в php.ini<BR>";
    break;
case UPLOAD_ERR_FORM_SIZE:
    echo "Размер загружаемого файла превысил значение,
установленное в MAX_FILE_SIZE<BR>";
    break;
case UPLOAD_ERR_PARTIAL:
    echo "Загружаемый файл был получен только частично<BR>";
    break;
case UPLOAD_ERR_NO_FILE:
    echo "Файл не был загружен<BR>";
    break;
}//end - switch
```

```
?>
<pre>
<?php
print_r($_FILES)
?>
</pre>
```

9.4.2. Познаем тонкости

После того как пользователь выбрал файл и отправил его на сервер, этот файл будет сохранен в специальной папке (которая задается в параметре `upload_tmp_dir` файла php.ini, более подробную информацию см. в *приложении 1*). Если с файлом не будет произведено никаких операций (например, перемещение его из временного каталога в постоянный), то через некоторый промежуток времени закаченный файл будет автоматически удален. Также стоит отметить про ограничение на размер файла, устанавливаемого самим сервером, за это значение отвечает параметр `upload_max_filesize`, находящийся в php.ini (более подробную информацию об этом можно найти в *приложении 1*).

На некоторых хостах может быть установлена версия PHP ниже 4.1.0, а в ней нет массива `$_FILES`, и вместо него нужно использовать массив `$HTTP_POST_FILES`. Поэтому, прежде чем приступать к программированию скрипта по закачке файла, уточните версию PHP на хосте, где в дальнейшем будет располагаться ваша программа, чтобы не попасть в ситуацию, когда вроде бы все сделано правильно, но почему-то скрипт не работает. В версиях, начиная с 4.1.0, массив `$HTTP_POST_FILES` тоже можно использовать (но в данном случае массив `$_FILES`

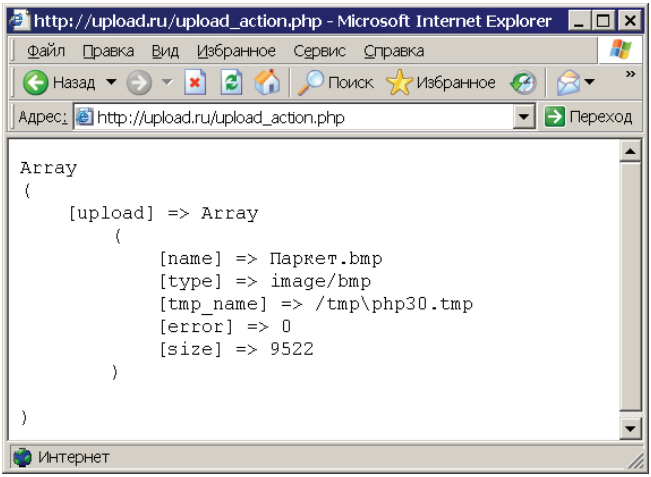


Рис. 9.65. Использование массива `$HTTP_POST_FILES`

более предпочтителен), поэтому если в качестве upload_action.php создать файл следующего содержания:

```
<pre>
<?php
print_r($HTTP_POST_FILES)
?>
</pre>
```

то все будет работать без проблем (рис. 9.65).

9.4.3. Полноценный скрипт для закачки файла

Рассмотрим более практичный пример, разрешающий пользователю загружать на сервер только картинки с расширением jpg, gif, png. Создайте файл index.html (листинг 9.20).

Листинг 9.20. Файл index.html

```
1*  <form action="upload_action.php" method="POST"
    enctype="multipart/form-data">
2   <!--Ставим ограничение в 600 кбайт-->
3   <input type="hidden" name="MAX_FILE_SIZE" value="614400">
4   <input type="file" name="upload" >
5   <input type="submit" name="upload_button" value="Закачать">
6   </form>
```

Теперь создайте файл upload_action.php (листинг 9.21) и каталог PICTURE — именно там будут храниться картинки, загруженные пользователем.

Листинг 9.21. Файл upload_action.php

```
1   <?php
2   //Проверка на то, что обработка формы
3   if (isset($_POST["upload_button"]))
4   {
5     //Проверка на то, что нет ошибок
6     if ($_FILES['upload']['error']==0)
7     {
8       //Получаем имя файла, который хочет закачать пользователь
9       $upload_file = $_FILES['upload']['name'];
10      //Задаем каталог, куда будут грузиться картинки для пункта
11      $upload_dir = 'PICTURE/';
12
```

```
13      //Массив расширений графических файлов разрешенных для закачки
14      $images_allow =array ("JPG", "GIF", "PNG","jpg", "gif", "png");
15      //Получаем информацию о файле
16      $file_info = pathinfo($upload_file);
17      //Проверка на то, что расширение файла разрешено для загрузки
18      if (!(in_array($file_info['extension'],$images_allow)))
19      {
20*         echo "Расширение ". $file_info['extension']." Является
    недопустимым, ";
21          echo "вы можете загружать только файлы JPG, GIF и PGN!<BR>";
22          echo "<a href='index.html'>НАЗАД</a>";
23          exit;
24      }
25
26       //Файл, который будем копировать, он закачан на сервер
27       $tmp_path=$_FILES['upload']['tmp_name'];
28       //Путь+имя файла, куда будем копировать
29       $full_path_for_copy=$upload_dir.$upload_file;
30
31       //Пытаемся переместить загруженный файл из временного хранилища
32       if (move_uploaded_file($tmp_path,$full_path_for_copy))
33       {
34          echo 'Все ОК';
35*         echo "Картинка успешно загружена, вы можете видеть ее
    ниже.<BR>";
36          echo "<img src=".$full_path_for_copy.">";
37          exit;
38       }
39       else
40       {
41*         echo 'Ошибка, не удалось скопировать файл из временного
    каталога.';
42          echo "<br><a href='index.html'>НАЗАД</a>";
43          exit;
44       }
45    }
46    else
47    {
48*     echo "Невозможно загрузить файл из-за возникновения ошибки
    с кодом ";
```

```
48      echo $_FILES['upload']['error'];
49      echo "<br><a href='index.html'>НАЗАД</a>";
50    }
51  }
52  ?>
```

Сначала осуществляется проверка на то, запущен ли скрипт как обработчик. Затем еще одна проверка, не было ли ошибок при закачке файлов — если да, то осуществляется переход на строку **47**, в результате чего будет выведен текст `"Невозможно загрузить файл из-за возникновения ошибки с кодом"`, код ошибки и ссылка **Назад**, нажатие которой вернет пользователя к форме загрузки файла.

В строке **14** в массиве `$images_allow[]` задаются расширения файлов, разрешенные к закачке. В строке **16** используется новая функция `pathinfo()`, ее синтаксис:

```
pathinfo(имя_файла);
```

Функция `pathinfo()` возвращает в виде массива информацию о файле, переданном в качестве параметра, массив состоит из трех элементов:

- `dirname` — путь к файлу;
- `basename` — имя файла;
- `extension` — расширение, именно этот параметр нас интересует, поэтому в следующей строке (в условии) мы обращаемся к этому параметру.

В качестве параметра функции `pathinfo()` передается переменная `$upload_file`, в которую в строке **9** было сохранено имя закачанного на сервер файла. Результат работы функции сохраняется в переменной `$file_info`. Для определения, является ли допустимым расширение закачанного пользователем файла, используется следующий код:

```
if (!(in_array($file_info['extension'],$images_allow)))
```

В нем вы можете видеть новую функцию `in_array()`, ее синтаксис:

```
in_array(искомое_значение, массив);
```

Данная функция осуществляет поиск значения, переданного в качестве первого параметра в массиве, имя которого передано в качестве второго параметра, если значение найдено, то функция вернет `TRUE`, в противном случае — `FALSE`.

Если все проверки пройдены успешно, то далее в строке **27** в переменной `$tmp_path` происходит сохранение пути временного хранения закачанного файла, затем в строке **29** формируется путь, по которому будет перемещен файл. В строке **32** происходит перемещение файла, для этого используется новая функция `move_uploaded_file()`, ее синтаксис:

```
move_uploaded_file(путь_к_закачанному_файлу,
путь_куда_его_переместить);
```

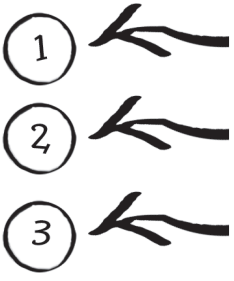
Функция осуществляет перемещение закачанного на сервер файла, путь к которому передан в качестве первого параметра, в новое место, указанное во втором параметре. Если с помощью данной функции осуществляется попытка переместить какой-то другой файл (не закачанный с помощью специально предназначенной для этого формой), то функция просто вернет `FALSE`. Если функция была выполнена успешно, то она вернет `TRUE`. В программе upload_action.php в строке **32** проверяется результат работы функции `move_uploaded_file()`, если он положителен (равен `TRUE`), то выводится текст `"Картинка успешно загружена, вы можете видеть ее ниже"`, под которым пользователь сможет увидеть закачанную им картинку. В противном случае выводится сообщение об ошибке: `"Ошибка, не удалось скопировать файл из временного каталога"`.

Параметры, связанные с загрузкой файлов на сервер, рассмотрены в приложении 1.

9.5. Определяем быстродействие скрипта

Одним из самых важных свойств скрипта является время его выполнения. Потому что именно оно напрямую характеризует как долго придется ждать посетителю при обращении к сайту. Мало разработать программу, нужно попытаться ее сделать быстрой, особенно это актуально для Web-ресурсов, на которых ожидается большое количество посетителей.

Принцип оценки быстродействия сводится к следующему:



Перед началом выполнения скрипта определяется текущее время.

В конце выполнения скрипта опять определяется текущее время.

Затем из второго значения вычитается первое и получается время работы скрипта.

Для реализации первых двух пунктов используется функция повышенной точности `microtime()`, которая возвращает результат в формате `"msec sec"`, где

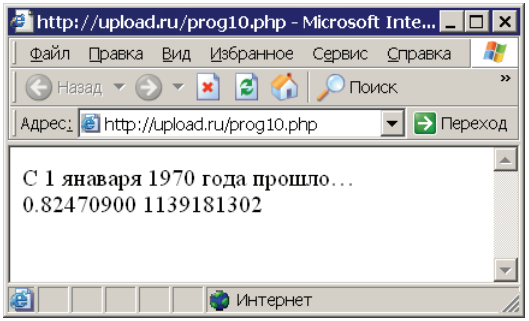


Рис. 9.66. Результаты использования функции `microtime()`

`sec` — количество секунд, прошедших с 1 января 1970 года, а `msec` — дробная часть. Рассмотрим пример (prog10.php):

```
<?php
echo "С 1 января 1970 года прошло…<BR>";
echo microtime();
?>
```

результат выполнения которого изображен на рис. 9.66.

Так как функция `microtime()` возвращает два значения, то для дальнейшей работы с ними их нужно сохранить отдельно друг от друга. Для этого используется функция `explode()`, ее синтаксис:

```
explode("разделитель","строка")
```

Данная функция делит строку на фрагменты в соответствии с заданным разделителем. В результате выполнения возвращается массив, каждый элемент которого равен отдельному фрагменту, например, следующий пример (prog11.php):

```
<?php
$stroka="красный,синий,голубой";
//Делим строку на фрагменты, используя в качестве разделителя запятую
$mas=explode(',',$stroka);
//Распечатываем полученный массив
print_r($mas);
?>
```

вернет такой результат — рис. 9.67.

А вот в результате выполнения этой программы (prog12.php):

```
<?php
$stroka=" машина|танк|самолет ";
//Делим строку на фрагменты, используя в качестве разделителя запятую
$mas=explode('|',$stroka);
//Распечатываем полученный массив
```

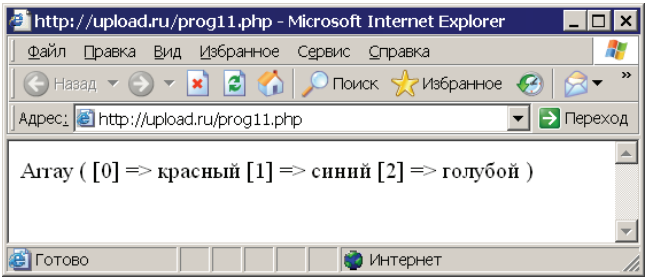


Рис. 9.67. Пример использования функции `explode()`

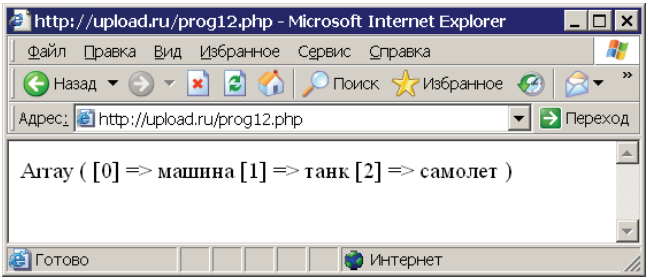


Рис. 9.68. Еще один пример использования функции `explode()`

```
print_r($mas);
?>
```

вы увидите следующее — рис. 9.68.

Остается дело за малым — всего лишь разработать функцию, которая будет возвращать в виде массива результат работы `microtime()`. После чего можно будет измерять время выполнения любой программы на PHP. Для *создания функции* используется следующая запись:

```
function имя_функции(параметры_которые_необходимо_передать_в_функцию)
{
    тело_функции
    return значение_которое_будет_возвращено_функцией
}
```

где `параметры_которые_необходимо_передать_в_функцию` — это переменные, разделенные запятой. Параметр может быть один или вообще отсутствовать, в последнем случае после названия функции ставятся пустые скобки. Под `телом_функции` подразумевается обычный код, который вы используете в программе на PHP. Элемент `return` является обязательным и размещается в самом конце тела функции, он обозначает результат, который функция вернет в вызывающую ее программу. На рис. 9.69 изображены составные части функции.

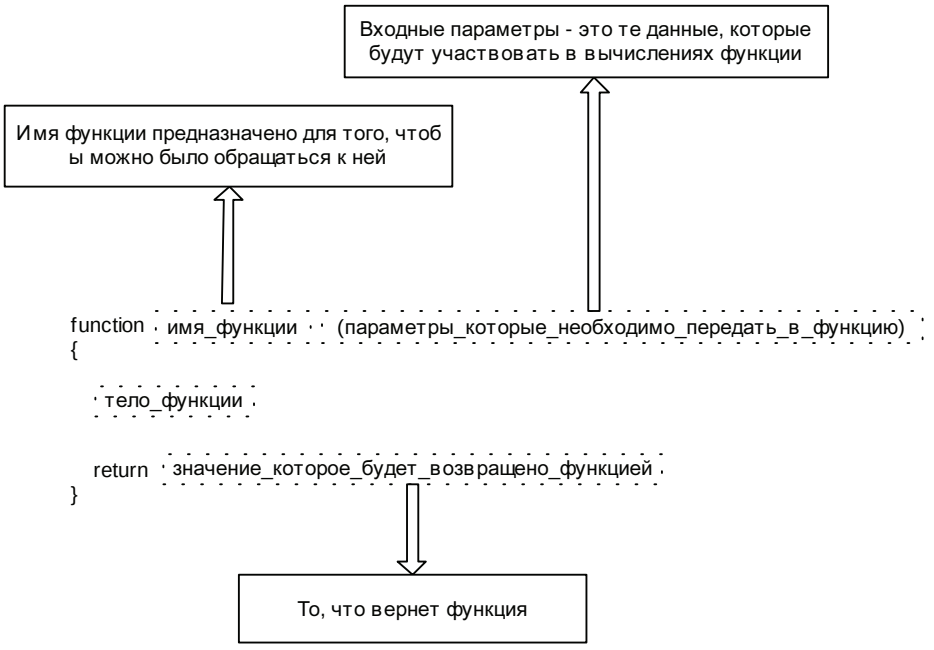


Рис. 9.69. Составные части функции

Предлагаю рассмотреть пример функции, возвращающей квадрат переданного ей числа в качестве единственного параметра:

```
<?php
//Объявляем функцию
function double_number($n)
{
   return $n*$n;
}
```

Далее представлен пример программы, работающей с только что разработанной функцией расчета квадрата числа (prog13.php):

```
<?php
//Объявляем функцию
function double_number($n)
{
   return $n*$n;
```

Замечание

Когда предполагается многократное использование одного и того же фрагмента кода, его, как правило, делают функцией, что позволит обращаться к нему из различных мест программы и даже модулей.

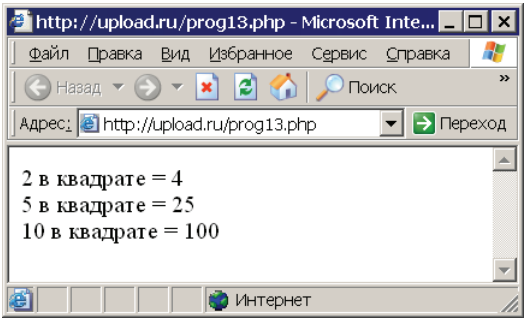


Рис. 9.70. Пример работы с функцией `double_number()`, созданной самостоятельно

```
}

echo "2 в квадрате = ".double_number(2)."<BR>";
echo "5 в квадрате = ".double_number(5)."<BR>";
echo "10 в квадрате = ".double_number(10)."<BR>";
?>
```

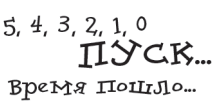
Результат ее работы представлен на рис. 9.70.

Давайте рассмотрим еще одну программу, в которой объявлена функция, складывающая три числа, переданных в качестве параметра (prog14.php):

```
<?php
//Объявляем функцию
function summ($first_num,$second_num,$third_num)
{
    $result=$first_num+$second_num+$third_num;
    return $result;
}

echo "10+45+78 = ".summ(10,45,78)."<BR>";
?>
```

Результат ее работы представлен на рис. 9.71.



Вернемся к теме оценки быстродействия скрипта. Создайте отдельный модуль с именем time_of_speed.php следующего содержания:

```
<?php
function our_time()
{
    //Получаем время
    $time_mas=explode(' ',microtime());
```

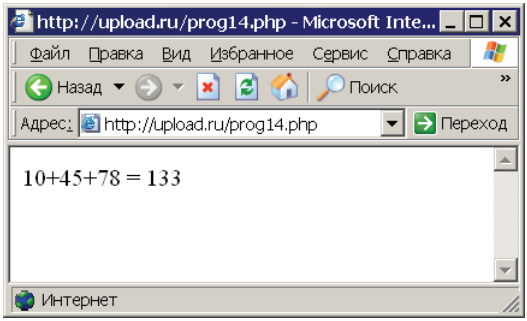


Рис. 9.71. Пример работы с функцией `summ()`, созданной самостоятельно

```
    //Складываем две составляющие результата функции microtime() в
одну
    $res=$time_mas[0]+$time_mas[1];
    //Возвращаем результат выполнения функции our_time()
    return $res;
}
?>
```

Давайте оценим время формирования скрипта системы голосования, скопируйте time_of_speed.php в каталог opros.ru/www, теперь откройте файл opros.ru/www/index.php и измените его следующим образом (обратите внимание на строки с **3** по **5** и строки с **51** по **55**) — листинг 9.22

Листинг 9.22. Файл opros.ru/www/index.php со встроенной системой определения быстродействия

```
1  <?php
2  //Подключаем модуль оценки быстродействия
3  require_once('time_of_speed.php');
4  //Засекаем начальное время
5  $begin_time=our_time();
6
7  //Читаем файл с вопросом и вариантами ответов
8  $variants = file('variants.txt');
9              //Текущий элемент массива $variants
10             $current_variants=0;
11 //Читаем файл с результатами голосования
12 $results = file('results.txt');
13             //Текущий элемент массива $results равен 1, потому что
14             //0 элемент - это фраза "Результаты голосования"
```

```
15              $current_results=1;
16              //Всего проголосовало человек
17              $itog=0;
18  ?>
19  <form action="add_golos.php" method="post">
20  <?php
21  //Выводим тему голосования
22  echo $variants[$current_variants]."<BR>";
23  //Увеличиваем текущий элемент массива $variants на 1
24  $current_variants++;
25*
    //Пока есть варианты для голосования, выводим их
26  while (count($variants)!=$current_variants)
27  {
28    //Выводим вариант голосования в виде переключателя
29    echo '<input name="first" type=radio
30  value="'.$current_variants.'">'.$variants[$current_variants];
31    //также выводим результаты для этого варианта,
32    //он будет заключен в скобки
33    echo ' ('.$results[$current_results].')';
34    //Прибавляем к общему числу проголосовавших
35    //значение по текущему варианту
36    $itog=$itog+$results[$current_results];
37    //Перевод строки
38    echo '<BR>';
39    //Увеличиваем текущий элемент массива $variants на 1
40    $current_variants++;
41    //Увеличиваем текущий элемент массива $results на 1
42    $current_results++;
43  }//end - while
44
45  //Выводим общее число проголосовавших
46  echo '<BR> Всего проголосовало: '.$itog.' человек <BR>';
47  ?>
48  <input name='add_golos' type="submit" value="Проголосовать">
49  </form>
50  <?php
```

```
51  //Засекаем конечное время
52  $end_time=our_time();
53  //Рассчитываем время формирования скрипта
54  $result_time=$end_time-$begin_time;
55  //Выводим результат
56  echo "Скрипт выполнен за: ".$result_time. " секунд";
57  ?>
```

Результат работы измененного index.php представлен на рис. 9.72.

Как видите, скрипт формируется достаточно быстро, но не совсем удобно то, что выводится слишком много чисел во времени. Для того чтобы результат сделать более коротким, воспользуемся функцией округления `random()`, ее синтаксис:

```
Random(число, количество_знаков);
```

Данная функция округляет число, переданное в ка-честве первого параметра, до количества знаков, пере-данного в качестве второго параметра. Например, сле-дующий скрипт:

```
<?php
echo round(0.000908136367798,4);
?>
```

вернет в результате выполнения число `0.0009`.

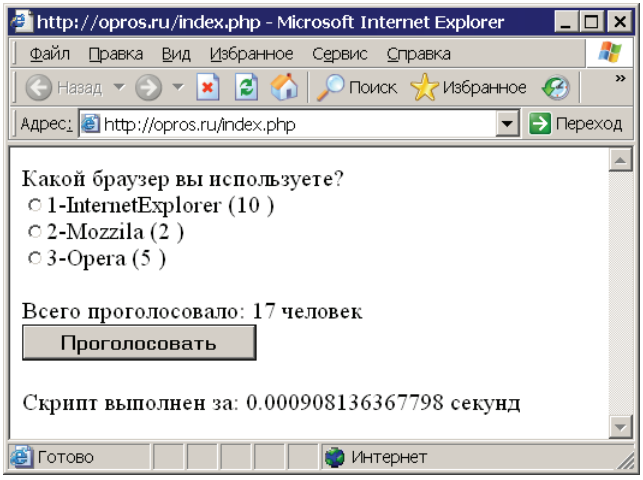


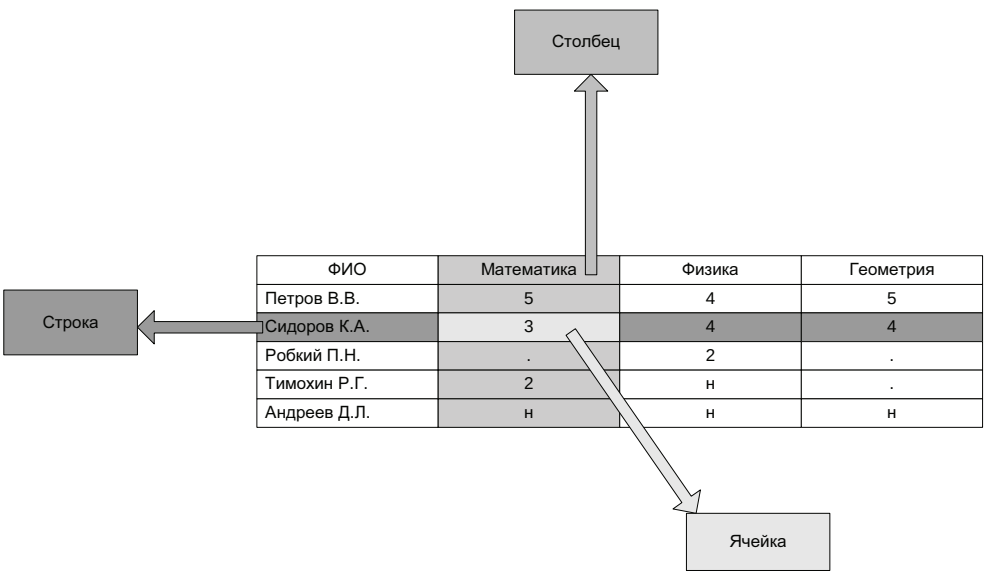
Рис. 9.72. Результат работы измененного index.php со встроенной системой определения быстродействия

Глава 10
Базы данных и работа с MySQL

10.1. Основные понятия

База данных (БД) предоставляет программисту очень удобный способ хранения информации, а также позволяет сократить время разработки ваших программ и повысить качество их функционирования.

База данных может состоять из множества таблиц, каждая из которых в свою очередь состоит из строк и столбцов, на их пересечении образуются ячейки. Рассмотрим пример школьного журнала (рис. 10.1).



ФИО	Математика	Физика	Геометрия
Петров В.В.	5	4	5
Сидоров К.А.	3	4	4
Робкий П.Н.	.	2	.
Тимохин Р.Г.	2	н	.
Андреев Д.Л.	н	н	н

Рис. 10.1. Простейший вариант базы данных, состоящей из одной таблицы с названием Школьный журнал

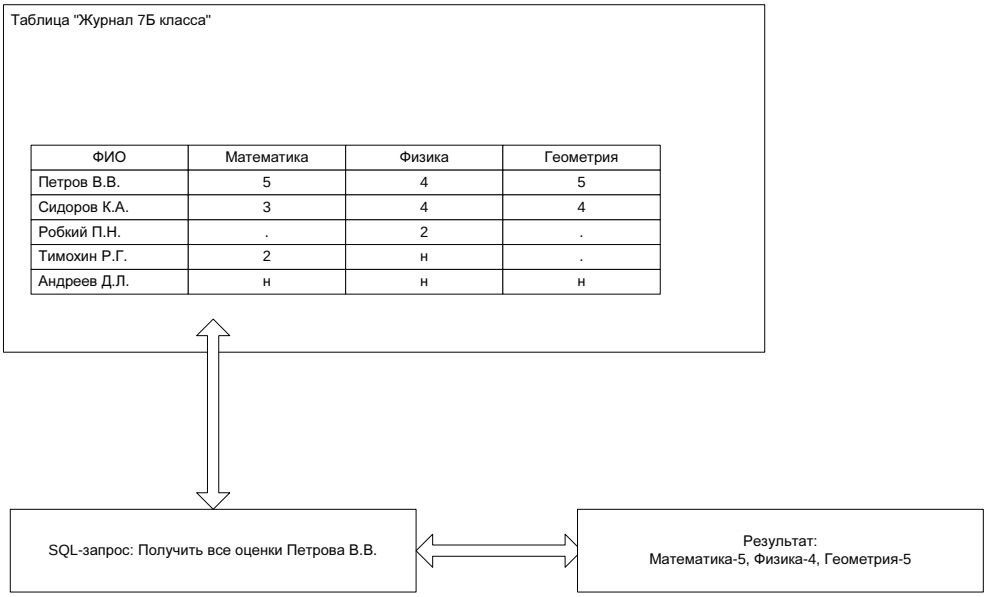


Рис. 10.2. Пример составления SQL-запроса для получения информации с заданным условием

Как видите, *столбцы* предназначены для хранения однотипной информации. Например, столбец `ФИО` хранит информацию о полном имени учеников, столбец `Математика` — оценки учеников по предмету Математика.

Ячейка сама по себе не несет никакого смысла — это будет просто цифра `2` или просто `Петров В.В.`, которых может быть огромное количество во всем мире. А вот если рассматривать ячейку как элемент столбца и строки, то сразу появляется смысл, например, можно сказать, что `Петров В.В.` — учится хорошо, потому что у него `5` и `4`, а `Андреев Д.Л.` — прогульщик, у него одни `н`.

Одно из главных преимуществ использования БД — это легкое манипулирование хранящейся в ней информации, потому что для ее получения или изменения используется специальный язык под названием SQL. Он очень прост. Команды, составленные на нем, называются *запросами*. Их составление очень похоже на составление обычных предложений, что иллюстрирует схема, изображенная на рис. 10.2.

То есть у нас есть действие, в данном случае `"Получить все оценки"`, есть объект воздействия — таблица `Школьный журнал`, а также дополнение — в виде условия `"нам необходимы оценки только Петрова П.П."`.

Еще один плюс в сторону работы с базой данных заключается в том, что она очень удобна для организации одновременного доступа многих пользователей

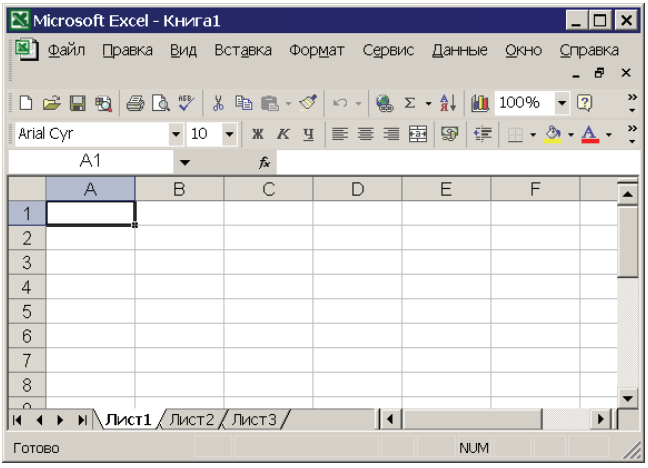


Рис. 10.3. Аналог таблицы — рабочий лист Microsoft Excel

к одной и той же информации — намного удобнее, чем файлы. Все что мы делали с текстовыми файлами — голосование, гостевая книга, все это можно сделать, используя в качестве хранилища информации базу данных. В этом случае процесс разработки программы даже будет быстрее и удобнее.

Для приобретения навыков работы с БД необходимо разбираться в основных понятиях, которые используются при работе с ней. Как вы уже знаете, сама по себе база данных — это некое хранилище, наподобие книжного шкафа, который хранит книги. Таблицы — это как полки у шкафа, каждая предназначена для чего-то своего, например, нижняя для романтических рассказов, средняя — для фантастики, а верхняя — для стихов. Самый простой аналог таблицы — это рабочий лист Microsoft Excel, который также состоит из строк, столбцов и ячеек (рис. 10.3).

Для работы с базой данных используется специальная программа, которая называется СУБД (система управления БД). Хотя существует достаточно много различных СУБД (например, MS SQL Server, Firebird, ORACLE), наибольшую распространенность в среде Web-программирования получила MySQL, т. к. она обладает оптимальным соотношением цены, скорости работы и устойчивости к ошибкам среди аналогов, к тому же MySQL очень проста в изучении. Официальный сайт MySQL **http://www.mysql.com**, русскоязычная документация доступна по адресу **www.mysql.com/doc/ru/index.html**.

Если у вас возникает вопрос, зачем же мы тогда изучали файлы? То ответ будет очень простым — нужно владеть навыками работы с различными инструментами, чтобы уметь решить одну и ту же задачу несколькими способами, потому что в некоторых случаях использование файлов будет более предпочтительным, например, при хранении картинок.

Стоит отметить, что очень часто при работе с БД строки называют записями, а столбцы — полями.

10.2. phpMyAdmin — первое знакомство

Дело в том, что у MySQL нет удобного графического интерфейса, поэтому для работы с этой СУБД чаще всего используют специальный инструмент, который позволяет до максимума упростить этот процесс. Его название phpMyAdmin, официальный сайт **www.phpmyadmin.net**, русскоязычный сайт доступен по адресу **http://www.php-myadmin.ru**.

Наберите в строке браузера `localhost`, а затем выберите пункт **phpMyAdmin**, вы увидите следующее окно — рис. 10.4.

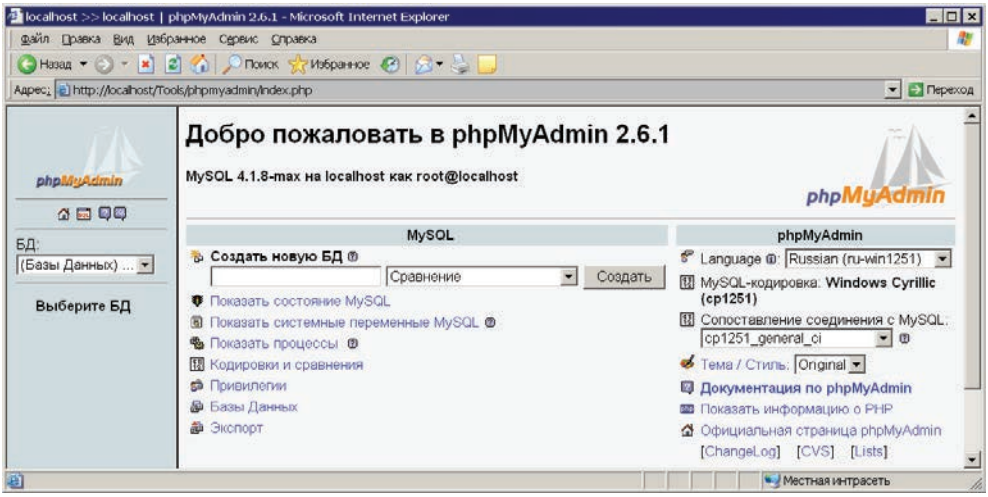


Рис. 10.4. phpMyAdmin

Замечание

phpMyAdmin полностью написан на PHP.

С помощью phpMyAdmin вы можете делать практически все — начиная от создания базы данных и заканчивая созданием пользователей для доступа к ней.

10.3. Разрабатываем структуру будущей базы данных

Мы рассмотрим работу с phpMyAdmin на примере реализации *форума*. Но прежде чем приступать к практическим действиям, составим план будущего проекта, потому что именно от грамотного составленного плана зависит важнейшая со-

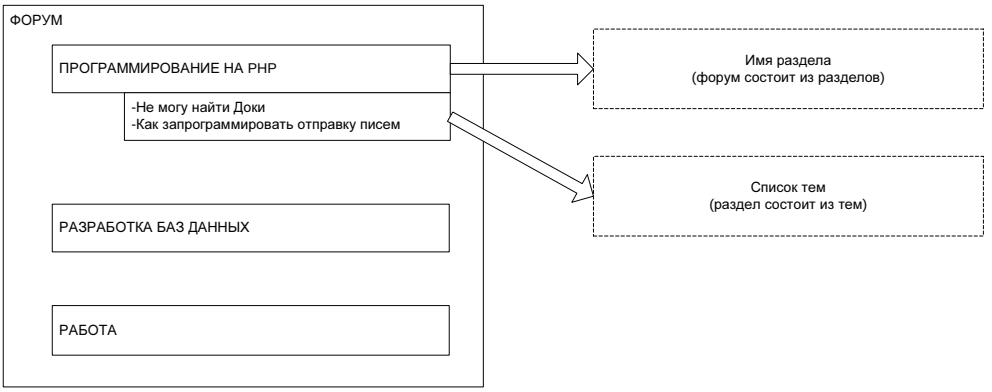


Рис. 10.5. Форум состоит из разделов, которые состоят из тем

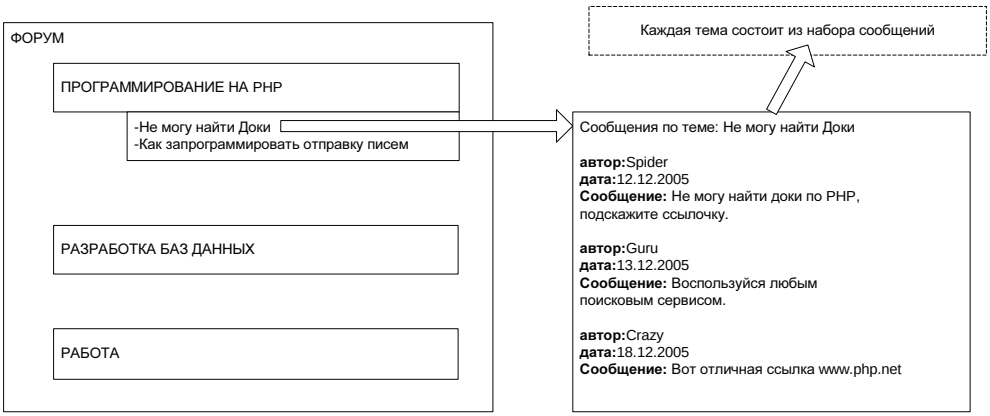


Рис. 10.6. Каждая тема состоит из набора сообщений

ставляющая успеха реализации качественного проекта, с минимальным количеством ошибок и внесений изменений на этапе кодирования.

В заранее предопределенных разделах форума пользователи могут общаться на различные темы (рис. 10.5).

Соответственно каждая тема состоит из набора сообщений пользователей (или посетителей), которые участвуют в ее обсуждении (рис. 10.6).

Если пользователь зарегистрирован, то у него есть имя и пароль, которые он использует при авторизации на форуме. В противном случае его сообщения размещаются под ником **Гость** (рис. 10.7).

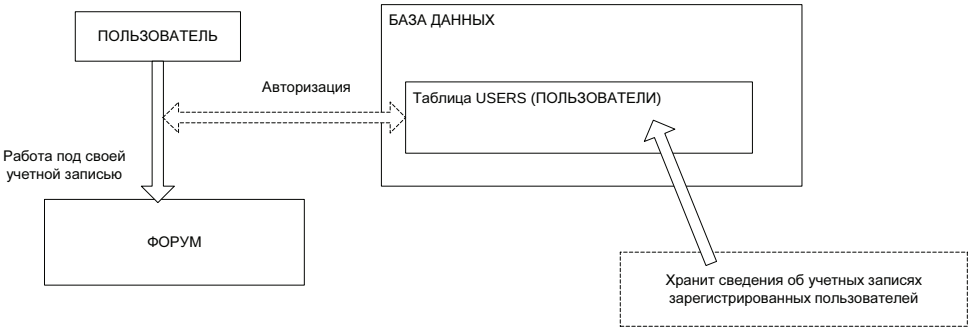


Рис. 10.7. Схема авторизации пользователя на форуме

Информация об учетных записях посетителей форума будет храниться в таблице `USERS`, ее структура представлена в табл. 10.1.

Таблица 10.1. *Структура таблицы* `USERS`

№	Название поля	Описание
1	`id`	Поле-счетчик
2	`name`	Имя пользователя
3	`pass`	Пароль пользователя
4	`role`	Роль пользователя, возможен один из двух вариантов: `admin` — администратор форума или `user` — обычный пользователь

У каждого пользователя будет свой идентификационный номер, хранящийся в поле `id`, причем, нам не нужно беспокоиться о его уникальности, т. к. эту заботу можно смело возложить на плечи MySQL (о том, как это сделать, будет сказано далее).

Также нам понадобится таблица `TOPIC`, которая будет хранить информацию о разделах и темах форума, ее структура представлена в табл. 10.2.

Рассмотрим, как будут выглядеть записи (или строки) в таблице `TOPIC`, описывающие раздел (табл. 10.3).

Комментарии:

- в `id` будет храниться уникальный номер, который характеризует эту запись в пределах всей таблицы;
- `kodofrazdel` будет содержать 0;
- `name` будет содержать название раздела;
- `name_creater` будет пустым;

- name_last_answer будет пустым;
- date_last_answer будет пустым.

Таблица 10.2. *Структура таблицы* TOPIC

№	Название поля	Описание
1	id	Поле-счетчик
2	kodofrazdel	Это поле предназначено для темы, в нем будет храниться id раздела, которому принадлежит тема. Для раздела в этом поле будет храниться 0
3	name	Название раздела или темы
4	name_creator	Поле предназначено для темы, в нем будет храниться имя ее автора. Для раздела поле будет пустым
5	name_last_answer	Поле предназначено для темы, в нем будет храниться имя пользователя, ответ которого для данной темы был последним. Для раздела поле будет пустым
6	date_last_answer	Поле предназначено для темы, в нем будет храниться дата последнего ответа по теме. Для раздела поле будет пустым

Таблица 10.3. *Записи для 2 разделов*

id	kodofrazdel	name	name_creator	name_last_answer	date_last_answer
1	0	Для программистов	пустое	пустое	пустое
2	0	Работа	пустое	пустое	пустое

Рассмотрим, как будут выглядеть записи (или строки) в таблице TOPIC, описывающие темы (табл. 10.4).

Таблица 10.4. *Записи для 2 тем, раздела "Для программистов"*

id	kodofrazdel	name	name_creator	name_last_answer	date_last_answer
3	1	Нужна помощь в поиске мануала по MySQL	Zero	MegaMan	2006-02-01
4	1	Подскажите, как установить PHP	Haos	Lion	2005-12-07

Комментарии:

- `id` будет хранить уникальный номер, который характеризует эту запись в пределах всей таблицы;
- `kodofrazdel` будет содержать `id` раздела, которому принадлежит данная тема, в нашем случае темы относятся к разделу "Для программистов" (видите, как легко можно делить все записи таблицы по группам (раздел, тема) с помощью уникального номера);
- `name` — название темы;
- `name_creater` — имя автора, который создал эту тему;
- `name_last_answer` — имя автора, чей ответ был последним по данной теме;
- `date_last_answer` — дата последнего ответа по данной теме.

Также в базе данных `FORUM` будет присутствовать таблица `MESSAGE`, предназначение которой состоит в хранении сообщений для всех тем, имеющихся на форуме (табл. 10.5).

Таблица 10.5. *Структура таблицы* `MESSAGE`

№	Название поля	Описание
1	`id`	Поле-счетчик
2	`kodoftopic`	Будет хранить `id` темы, которой принадлежит данное сообщение
3	`text_message`	Текст сообщения
4	`name_man`	Имя автора, разместившего сообщение
5	`date_answer`	Дата размещения сообщения

Рассмотрим, как будут выглядеть записи в таблице `MESSAGE`, описывающие сообщения (табл. 10.6).

Таблица 10.6. *Записи для 2 сообщений, темы "Нужна помощь в поиске мануала по MySQL"*

id	kodoftopic	text_message	name_man	date _answer
1	3	Где можно найти мануал по MySQL?	Zero	2006-01-05
2	3	Ты пробовал смотреть на официальном сайте?	MegaMan	2006-02-01

Стоит отметить, что наличие в таблицах поля с уникальным номером позволяет связывать между собой информацию, содержащуюся в этих таблицах. Яркий пример этому связь между таблицами `MESSAGE` и `TOPIC` по полю `kodoftopic` и `id` (рис. 10.8).

На рис. 10.9 представлена схема взаимосвязи таблиц базы данных `FORUM`.

id	kodoftopic	text_message	name_man	date_answ er
1	3	Где можно найти мануал по MySQL	Zero	15/01/06
2	3	Ты пробовал смотреть на официальном сайте?	MegaMan	1/02/06

эти сообщения принадлежат теме с идентификатором, равным 3

id	kodofrazdel	name	name_creater	name_last_answ er	date_last_answ er
3	1	Нужна помощь в поиске мануала по MySQL	Zero	MegaMan	1/02/06
4	1	Подскажите как установить PHP	Haos	Lion	7/12/05

Рис. 10.8. Схема связи таблиц MESSAGE и TOPIC

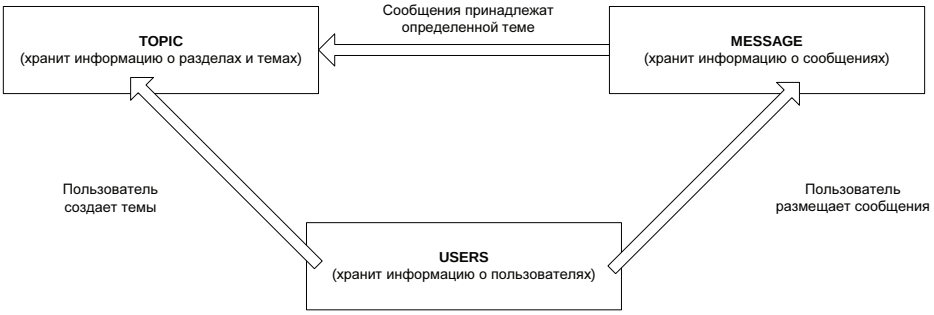


Рис. 10.9. Схема связи таблиц TOPIC, MESSAGE и USERS

10.4. Создаем БД или работаем в phpMyAdmin

Приступаем к созданию базы данных. Введите FORUM (это имя будущей базы данных) в поле **Создать новую БД** (рис. 10.10).

Далее нажмите кнопку **Создать**. Вы увидите следующее — рис. 10.11.

База создана. В левой части phpMyAdmin вы можете видеть имя созданной базы данных. Также можно наблюдать SQL-запрос, который характеризует только что выполненное действие. Стоит отметить, что язык SQL является связующим звеном между разработчиком и СУБД, т. к. последняя понимает именно этот язык,

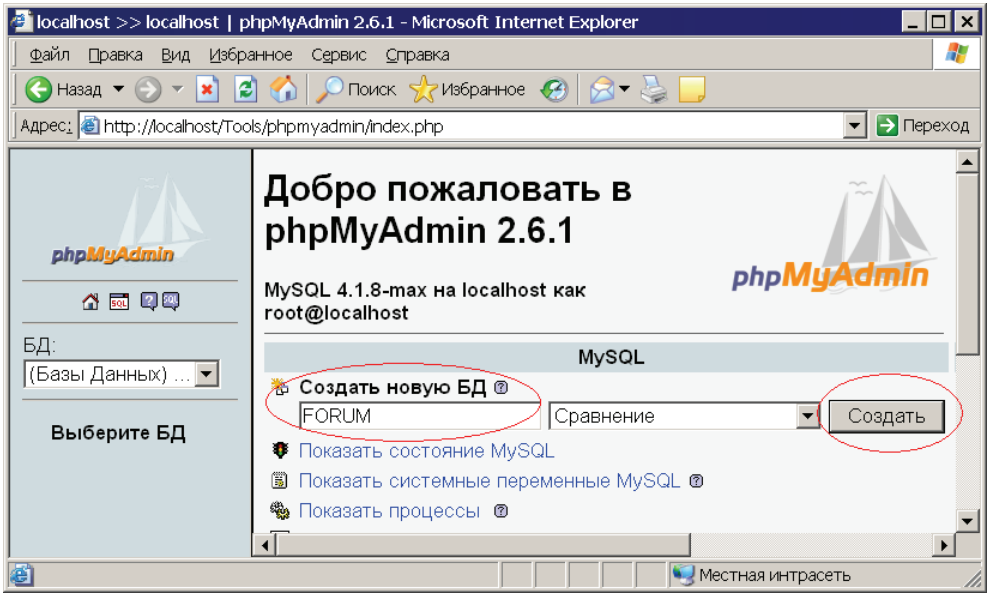


Рис. 10.10. Задание имени для будущей базы данных

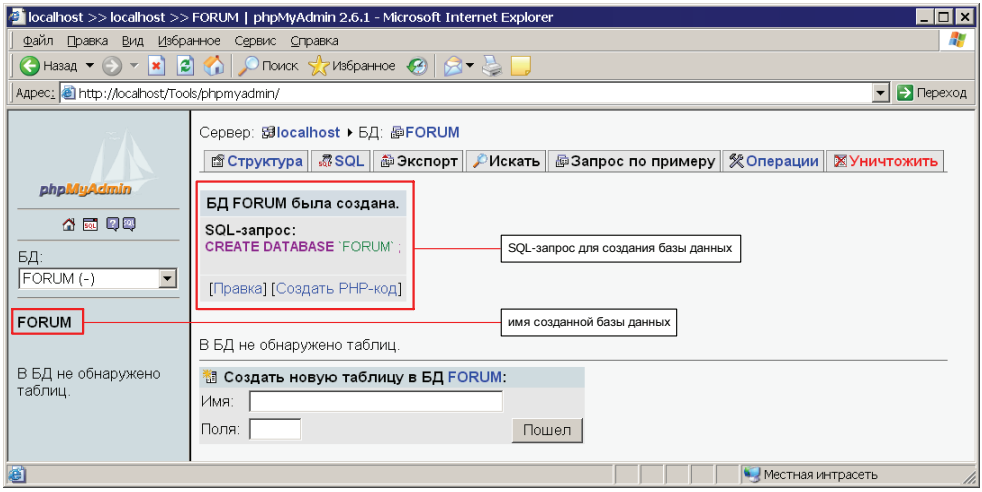


Рис. 10.11. Результат создания базы данных

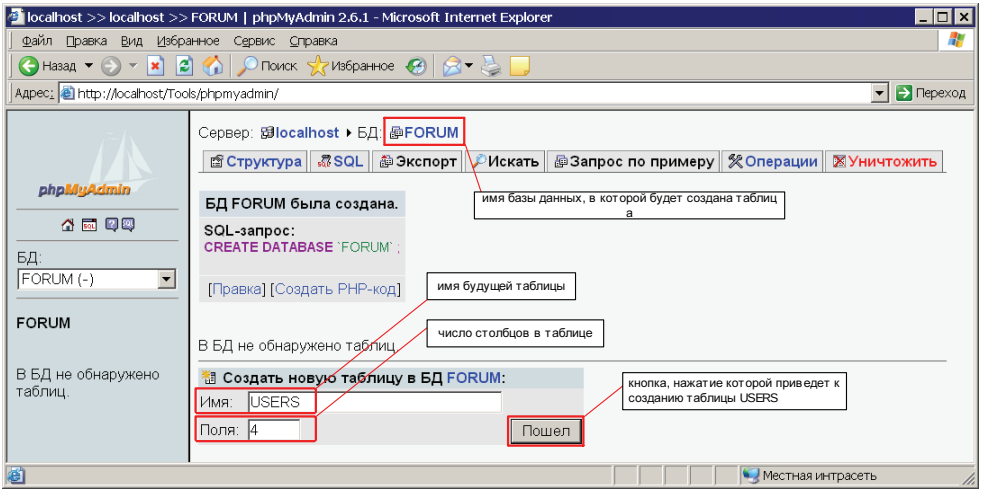


Рис. 10.12. Первый этап создания таблицы: ввод ее имени и числа столбцов, из которых она будет состоять

поэтому, чтобы совершить какое-то действие, необходимо составить SQL-запрос и отправить его СУБД. Система phpMyAdmin берет эту заботу на себя, предоставляя разработчику визуальный интерфейс для работы с MySQL, при этом phpMyAdmin показывает разработчику каждый отправленный ею SQL-запрос.

Теперь создадим таблицу `USERS` (ее структура рассмотрена в табл. 10.1), для этого в поле **Имя**, расположенном под надписью **Создать новую таблицу в БД FORUM**, введите `USERS`, а в поле **Поля** число `4` (рис. 10.12).

Нажмите кнопку **Пошел**, вы увидите набор из различных секций, каждая из которых состоит из однотипных элементов, предназначенных для ввода информации о будущих столбцах таблицы (рис. 10.13).

В секцию **Поле**, предназначенную для имени столбца, введите `id`, в секции **Тип** выберите **INT** — это тип данных для столбца (рис. 10.14).

Давайте рассмотрим более подробно наиболее часто используемые типы данных:

- `VARCHAR` — текстовый тип, предназначенный для хранения до 255 символов;
- `TEXT` — текстовый тип данных, предназначенный для хранения до 65 535 символов;

Команда `CREATE DATABASE` в переводе с англ. означает "создать БД" — лишнее доказательство простоты SQL — его команды строятся из обычных слов.

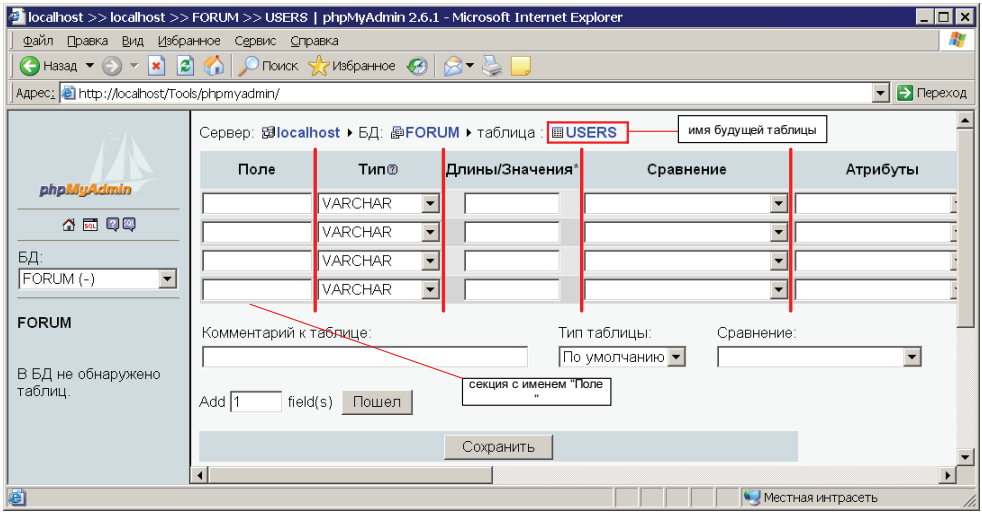


Рис. 10.13. Форма для ввода информации о столбцах для создаваемой таблицы

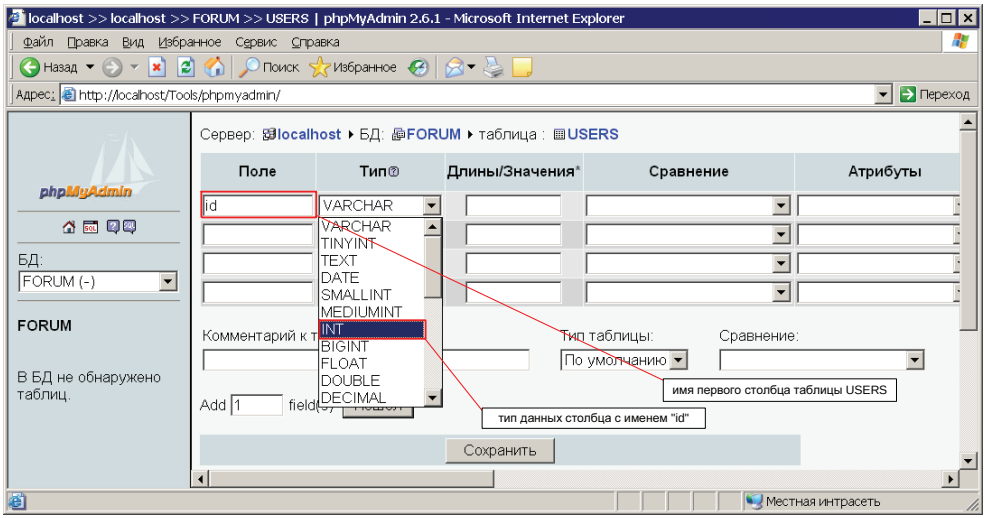


Рис. 10.14. Установка свойств первого столбца таблицы USERS

- INT — числовой тип, предназначенный для хранения целых чисел в диапазоне от –2 147 483 648 до 2 147 483 647;
- DATE — тип данных, предназначенный для хранения даты, как правило, она преобразуется к формату гггг-мм-дд.

Добавьте столбец name типа VARCHAR с максимально возможным количеством символов, равным 20. А также столбец pass типа VARCHAR с максимальным количеством символов, равным 32 (это нужно, т. к. в нем будет храниться хеш пользовательского пароля, возвращенный функцией md5(), а она возвращает 32-значное число). И последним добавьте столбец role типа VARCHAR с максимальным количеством символов, равным 10 (рис. 10.15).

Теперь займемся настройкой таблицы USERS. Сначала сделаем так, чтобы столбцу id значения присваивались автоматически, для этого прокрутите окно браузера с помощью горизонтального скроллинга, пока не будет виден выпадающий список **Дополнительно**. В нем необходимо выбрать единственный доступный пункт **auto_increment**. Но это еще не все, также необходимо установить переключатель

Замечание

Реальные хостинги чаще всего работают под управлением UNIX или LINUX, в которых несоблюдение регистра букв при обращении к тем или иным элементам приведет к тому, что они не будут найдены. Поэтому, чтобы не запутаться как пишется то или иное название элемента, предлагаю все таблицы называть прописными буквами, а поля строчными.

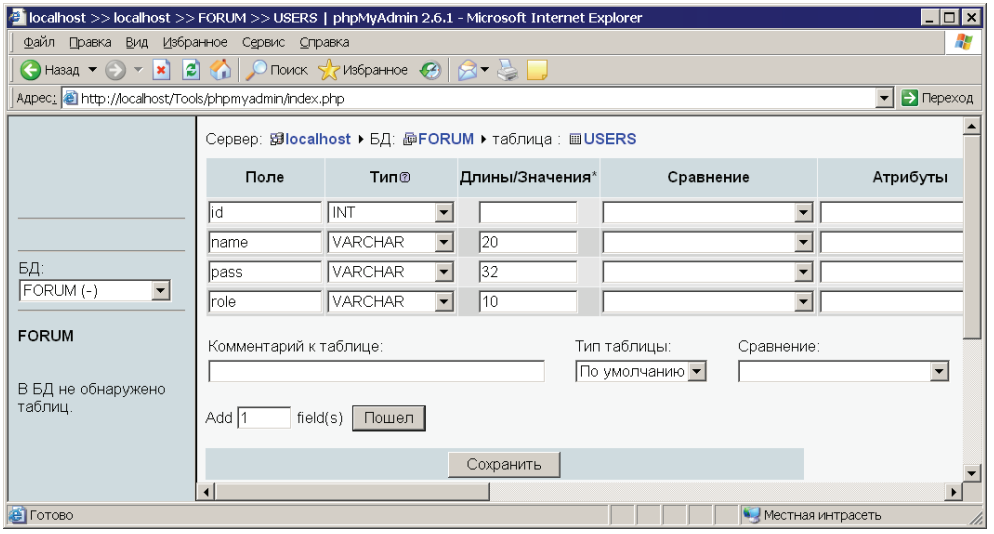


Рис. 10.15. Добавление столбцов в будущую таблицу с именем USERS

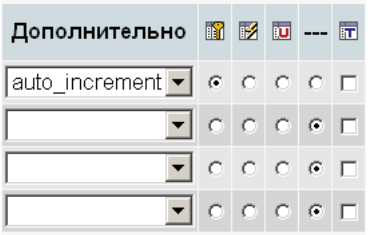


Рис. 10.16. Включение режима `auto_increment` и создание первичного ключа

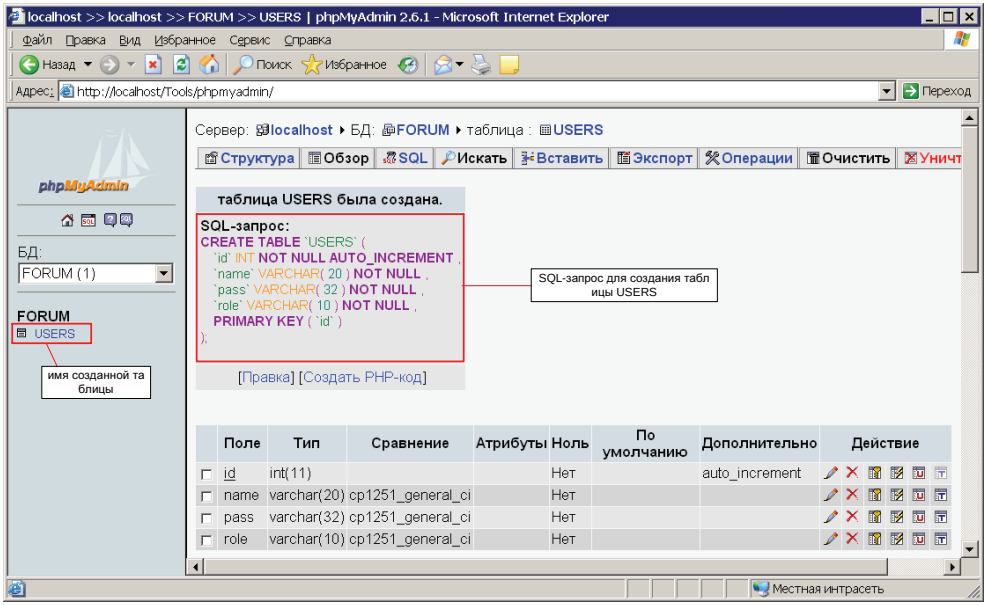


Рис. 10.17. Результат создания таблицы `USERS`

с изображением ключа **primary key** (первичный ключ), назначение которого — указать СУБД, что значения столбца будут уникальными. Это обязательное условие для работы режима `auto_increment`, смысл данного действия сводится к следующему: столбцу невозможно будет присвоить значение, которое уже в нем имеется, а если будет совершена попытка сделать это, то СУБД выдаст ошибку.

Нажмите кнопку **Сохранить**, вы увидите следующее — рис. 10.17.

Слева вы можете увидеть имя созданной таблицы. Давайте сразу добавим 2 записи, одна из которых будет характеризовать администратора, а другая — пользователя. Для этого щелкните левой кнопкой мыши на вкладке **Вставить**, расположенной в верхней части окна **phpMyAdmin** (рис. 10.18).

Это приведет к появлению формы, состоящей из двух абсолютно идентичных частей, для вставки новой записи в таблицу, которая является в данный момент активной, т. е. таблицы `USERS` (рис. 10.19).

Рис. 10.18. Набор вкладок, посредством которых можно выполнять различные действия с таблицей

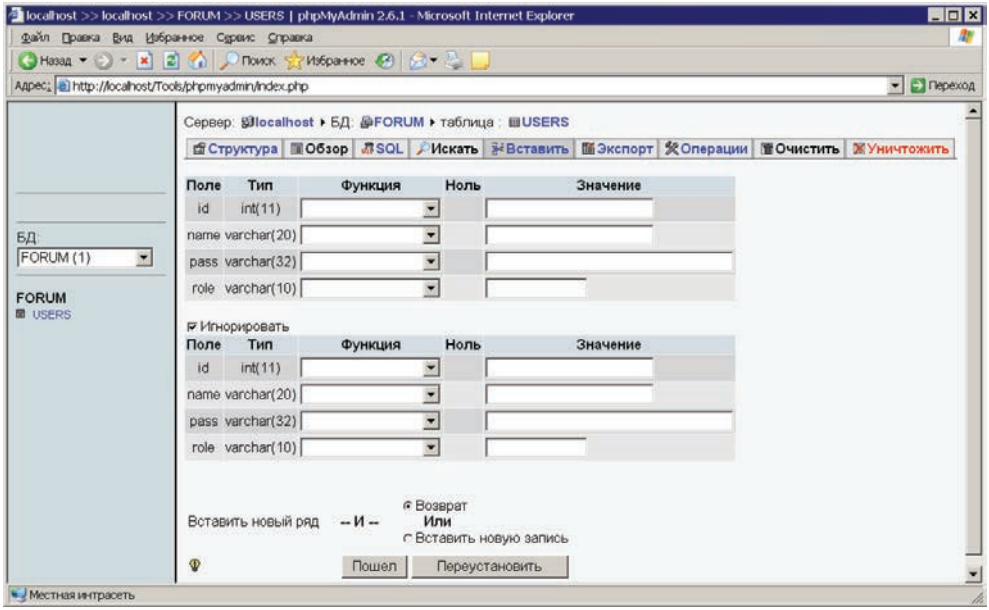


Рис. 10.19. Форма для вставки новых записей

Введите в верхнюю часть, в секцию **Значение**, следующую информацию:

- **id** — вводить не надо, т. к. MySQL его поставит автоматически;
- **name** — `administrator`;
- **pass** — `12345`;
- **role** — `admin`.

Как было указано ранее, пароль не хранится в открытом виде, вместо этого используется его хеш, полученный как результат функции `md5()`. И тут phpMyAdmin предоставляет нам удобный способ сделать это — в разделе **Функция** выберите в выпадающем списке, расположенном напротив слова **pass**, пункт **MD5** (рис. 10.20).

В результате в таблицу `USERS` в поле `role` будет записано значение, полученное от выполнения функции `md5()` с параметром `12345`, т. е. хеш от введенного вами значения, в чем вы убедитесь чуть позже.

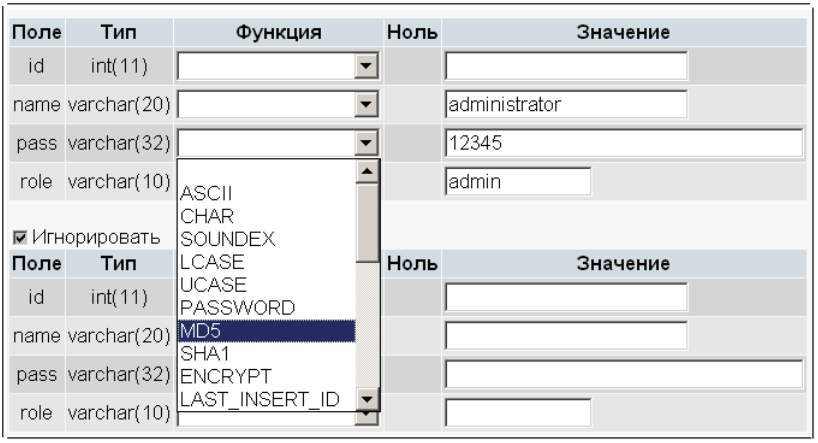


Рис. 10.20.
Определение хеша пароля средствами phpMyAdmin

Таким образом, с помощью phpMyAdmin можно не только добавлять данные, но и сразу обрабатывать их стандартными функциями.

Команда `INSERTO INTO... VALUES...` *в переводе с англ. означает "добавить в ... значения ... ".*

Теперь в нижнюю часть формы для добавления записи введите в секцию **Значение** следующую информацию о пользователе:

- **id** — вводить не надо, т. к. MySQL его поставит автоматически;
- **name** — `BigMan`;
- **pass** — `qwas` (не забудьте в выпадающем списке, расположенном слева от поля, куда вы ввели значение, выбрать пункт **MD5**);
- **role** — `user`.

Результат заполнения формы для добавления двух записей изображен на рис. 10.21.

Нажмите кнопку **Пошел**, после чего вы увидите следующее — рис. 10.22.

Теперь нажмите левой кнопкой мыши на вкладку **Обзор**, чтобы убедиться в том, что записи были добавлены (рис. 10.23).

Проведем небольшой эксперимент — попробуем добавить нового пользователя, идентификационный номер которого будет равен уже имеющемуся в базе. Нажмите левой кнопкой мыши на вкладку **Вставить** и в верхнюю форму введите следующую информацию (рис. 10.24).

- **id** — `1`;
- **name** — `sss`;
- **pass** — `aaa`;
- **role** — `user`.

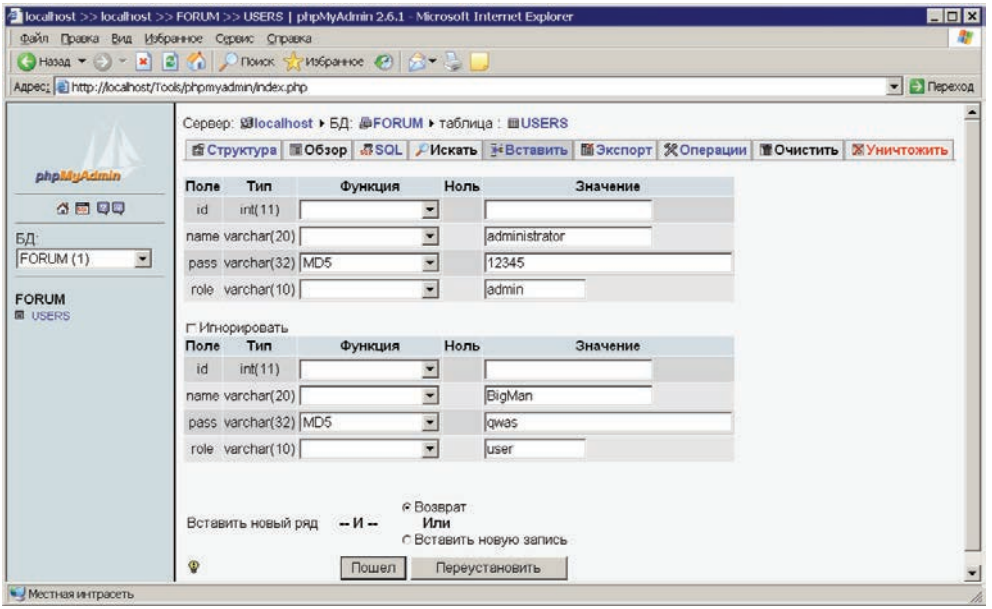


Рис. 10.21. Ввод данных для добавления двух записей в таблицу USERS

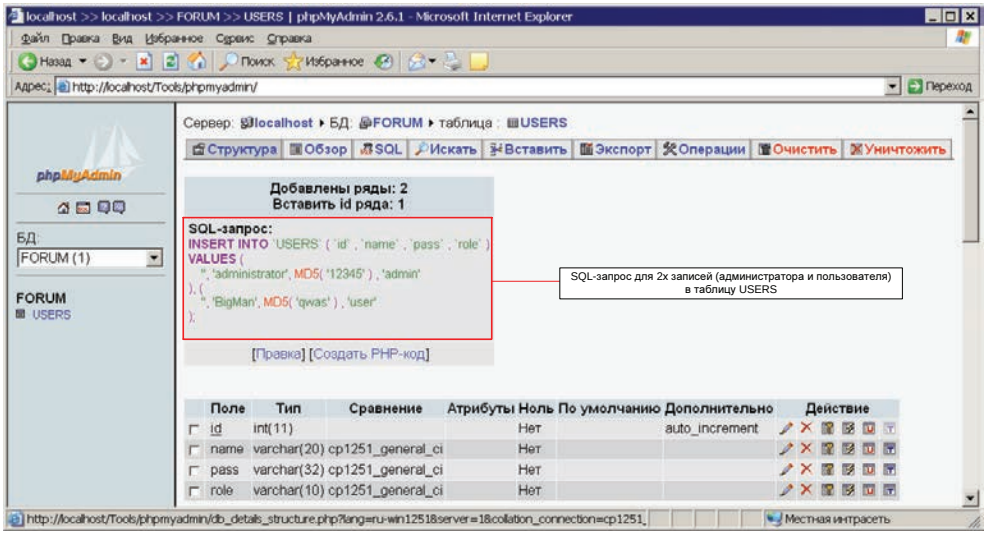


Рис. 10.22. Результат добавления 2 записей в таблицу USERS

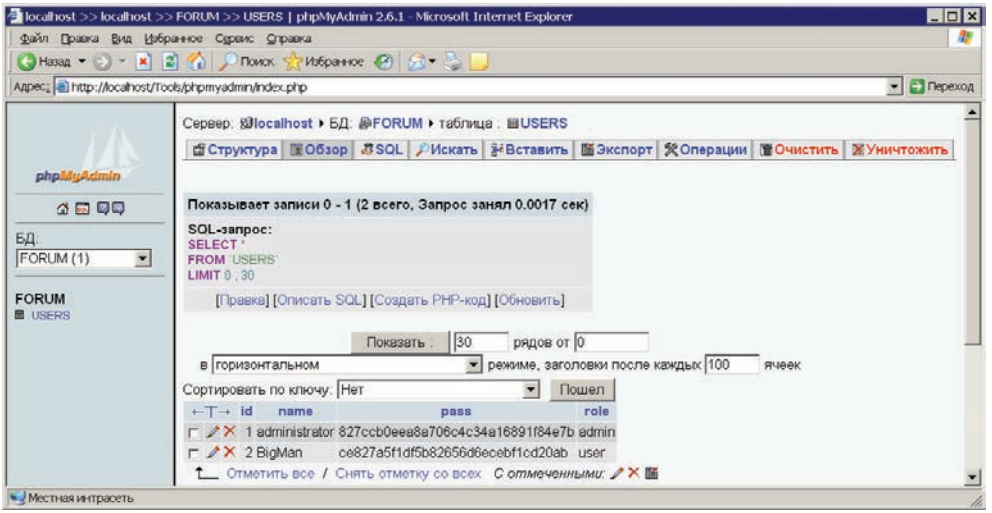


Рис. 10.23. Просмотр содержимого таблицы USERS

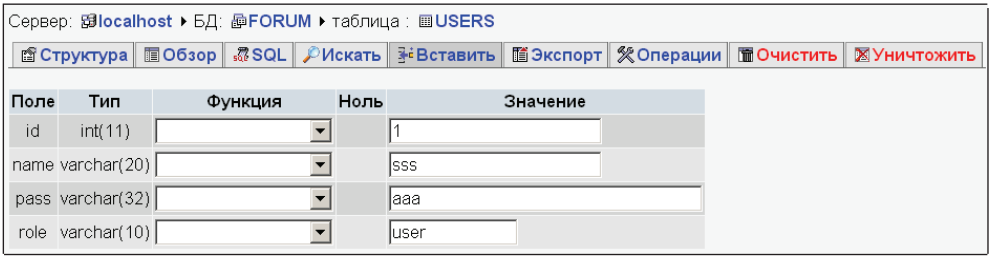


Рис. 10.24. Вставка заведомо ошибочной записи

Нажмите **Пошел**, в результате чего вы увидите сообщение об ошибке (рис. 10.25).

Переходим к созданию таблицы TOPIC, нажмите левой кнопкой мыши на ссылку **FORUM**, расположенную вверху (рис. 10.26).

Далее в поле **Имя**, расположенном под надписью **Создать новую таблицу в БД FORUM**, введите TOPIC, а в поле **Поля** — число 6 и нажмите кнопку **Пошел**. После этого заполните информацию о будущих столбцах таблицы TOPIC, представленную далее (рис. 10.27):

- имя — id, тип данных INT, значения присваиваются автоматически, первичный ключ;
- имя — kodofrazdel, тип данных INT;

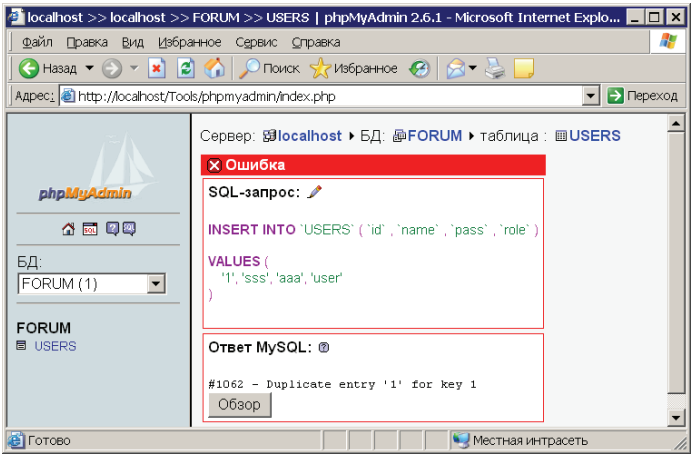


Рис. 10.25. Сообщение об ошибке, возникшее в результате попытки вставить запись с уже существующим значением столбца `id`

Рис. 10.26. Начало создания новой таблицы

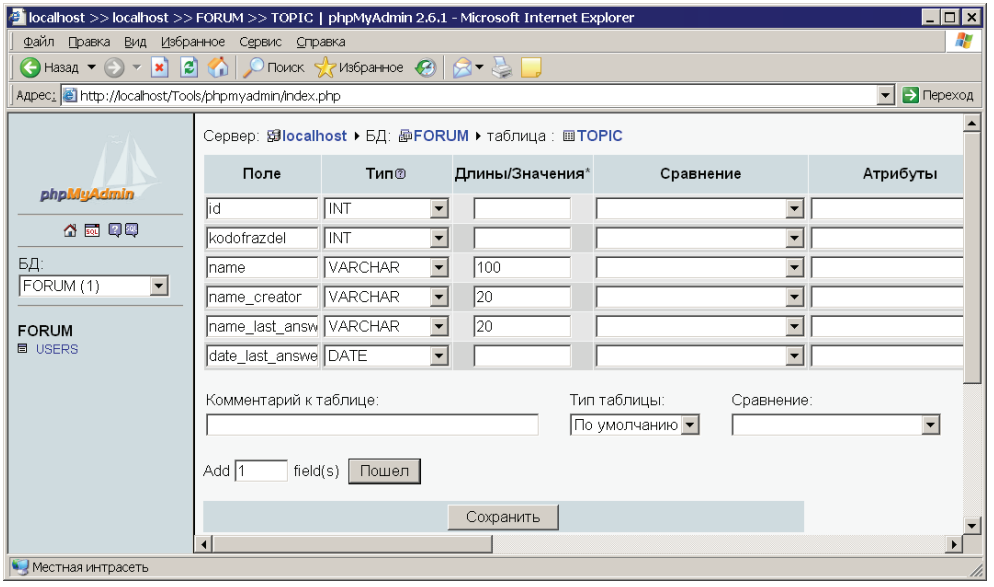


Рис. 10.27. Создание таблицы `TOPIC`

- имя — name, тип данных VARCHAR, максимально возможное количество символов — 100;
- имя — name_creator, тип данных VARCHAR, максимально возможное количество символов — 20;
- имя — name_last_answer, тип данных VARCHAR, максимальное возможное количество символов — 20;
- имя — date_last_answer, тип данных DATE.

Нажмите кнопку **Сохранить**, после чего добавьте в только что созданную таблицу две записи, которые будут характеризовать разделы (см. табл. 10.3), и две записи, которые будут характеризовать темы (см. табл. 10.4). Результат представлен на рис. 10.28.

Создайте таблицу MESSAGE, информация о ее столбцах:

- имя — id, тип данных INT, значения присваиваются автоматически, первичный ключ;
- имя — kodoftopic, тип данных INT;
- имя — text_message, тип данных TEXT;
- имя — name_man, тип данных VARCHAR, максимально возможное количество символов — 20;
- имя — date_answer, тип данных DATE.

После этого добавьте в нее две записи, согласно табл. 10.6, результат представлен на рис. 10.29.

id	kodofrazdel	name	name_creator	name_last_answer	date_last_answer
1	0	Для программистов			0000-00-00
2	0	Работа			0000-00-00
3	1	Нужна помощь в поиске мануала по MySQL	Zero	MegaMan	2006-02-01
4	1	Подскажите как установить PHP	Haos	Lion	2005-12-07

Рис. 10.28. Записи, добавленные в таблицу TOPIC

id	kodoftopic	text_message	name_man	date_answer
1	3	Где можно найти мануал по MySQL?	Zero	2006-01-05
2	3	Ты пробовал смотреть на официальном сайте?	MegaMan	2006-02-01

Рис. 10.29. Записи, добавленные в таблицу MESSAGE

10.5. Разрабатываем план кодирования

Весь форум будет состоять из модулей. Первый — это `Авторизация` — описывает механизм входа пользователя на форум под своей учетной записью. Если пользователь будет не авторизован, то все сообщения, размещенные им, будут сохраняться в базе данных под именем `gost`.

Второй — один из основных, это модуль, отвечающий за вывод информации на форуме (формирование списка разделов, вывод списка тем для выбранного раздела, а также вывод сообщений для заданной темы).

Третий модуль — действие, в нем будет реализована возможность создания новой темы, ответа в тему и т. д.

На рис. 10.30 представлена схема функционирования трех описанных ранее модулей.

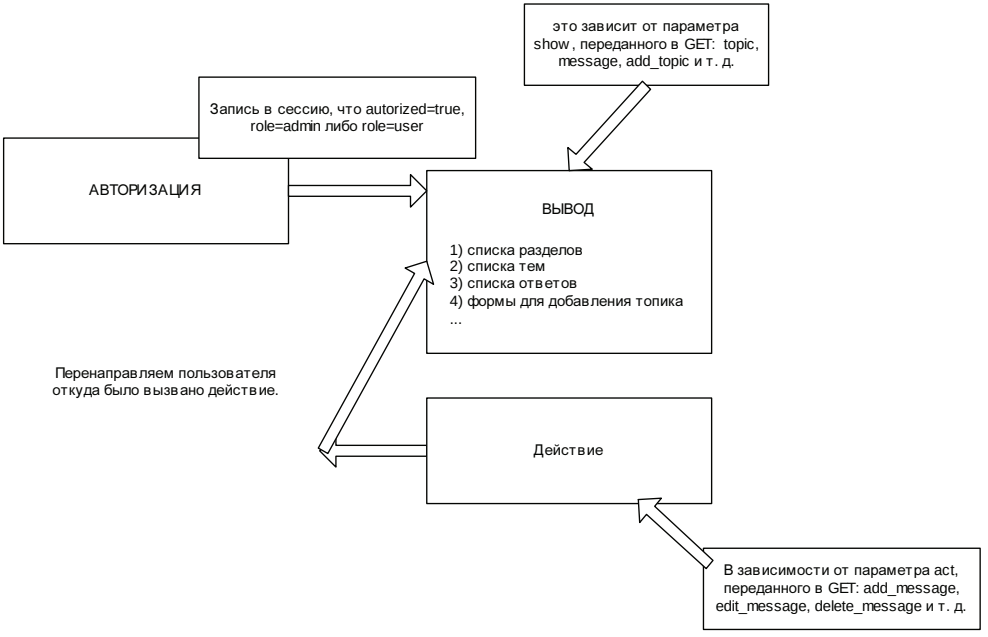


Рис. 10.30. Схема функционирования форума

10.6. SQL-запросы

Последним, что осталось рассмотреть, прежде чем приступать непосредственно к процессу кодирования, является тема составления SQL-запросов.

В основе построения запроса лежит задание ключевого слова (команды), которое указывает на то, что мы хотим сделать, а каждая команда состоит из блоков, в которых необходимо указать дополнительные параметры.

Самая часто используемая команда SQL — `SELECT` — предназначена для выборки данных. Общая схема ее построения представлена ниже:

```
SELECT имя_столбца1, имя_столбца2, ..., имя_столбцаN
       FROM таблица1, таблица 2, ..., таблицаN
       WHERE столбец1=условие1, столбец2=условие2, ...,
столбецN=условиеN
       ORDER BY столбец1, столбец2, ..., столбецN
```

Команда `SELECT` состоит из следующих блоков:

- `SELECT` указывает серверу, что клиент хочет получить данные;
- слово `FROM` указывает источник получения данных, им может быть таблица, представление и т. д.;
- слово `WHERE` позволяет указать условия выбора данных;
- слово `ORDER BY` указывает атрибуты, по которым будет произведена сортировка.

В phpMyAdmin выберите таблицу `MESSAGE` и нажмите левой кнопкой мыши на вкладку **SQL**, в поле для ввода многострочного текста **Выполнить SQL запрос на БД** введите следующий запрос:

```
SELECT id, kodoftopic, text_message, name_man, date_answer  FROM MESSAGE
```

который выберет все записи из таблицы `MESSAGE` (рис. 10.31).

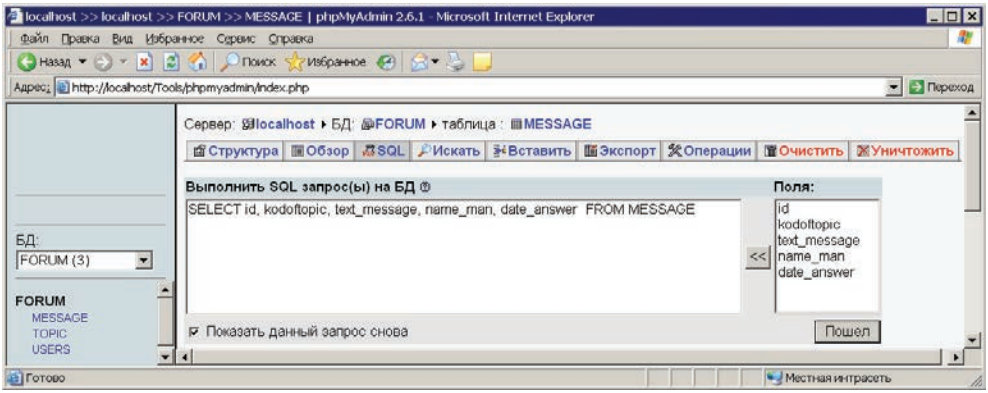


Рис. 10.31. Ввод SQL-запроса в phpMyAdmin

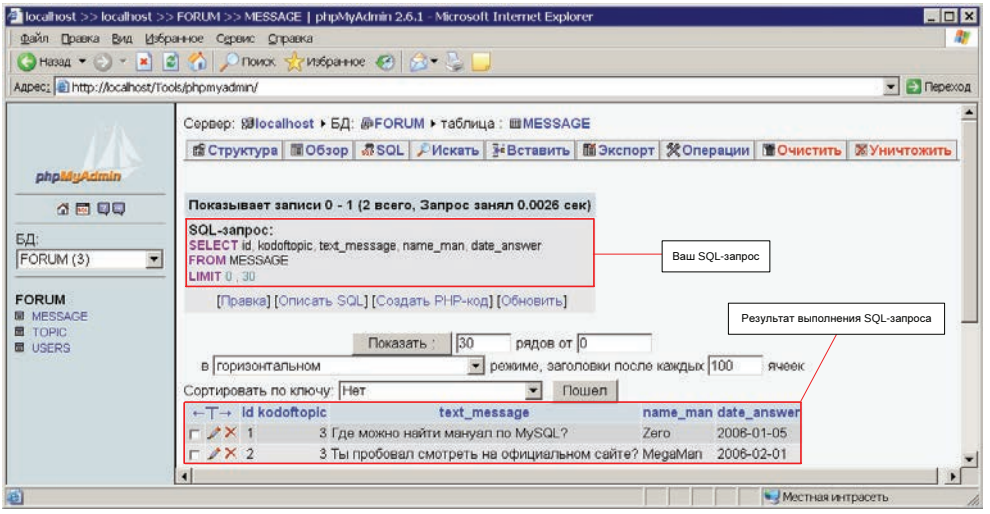


Рис. 10.32. Результат выполнения запроса

Нажмите кнопку **Пошел**, вы увидите следующий результат — рис. 10.32.

Теперь опять выберем все записи из таблицы MESSAGE, но еще и отсортируем их по столбцу name_man, для этого необходимо выполнить следующий запрос (рис. 10.33.):

```
SELECT id, kodoftopic, text_message,
name_man,
date_answer  FROM MESSAGE ORDER BY name_man
```

Обратите внимание, теперь на первом месте стоит запись с идентификационным номером, равным 2.

Для того чтобы выбрать записи, соответствующие определенному условию, используется специальное слово WHERE, после которого идет само условие или условия, если их несколько. Например, запрос:

```
SELECT id, kodoftopic, text_message, name_man, date_answer FROM MESSAGE
WHERE name_man='ZERO'
```

выведет следующий результат — рис. 10.34.

Когда необходимо добавить информацию в таблицу, используется команда INSERT, общая схема построения которой следующая:

```
INSERT INTO имя_таблицы
       SET
            имя_столбца1=значение1,
```

Замечание

*phpMyAdmin в некоторых случаях (например, при использовании вкладки **Обзор**) заключает имена столбцов и таблиц в апострофы. Это является альтернативным вариантом составления SQL-запроса для MySQL.*

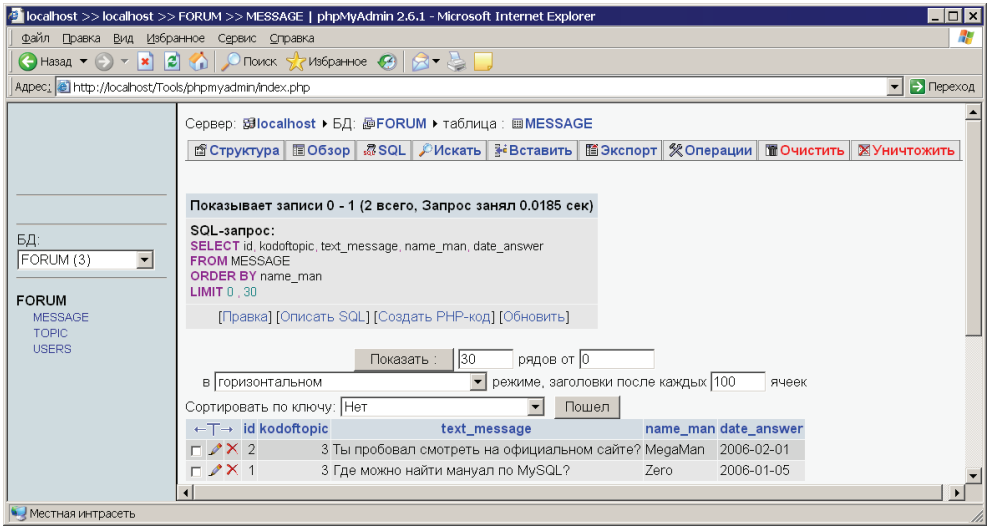


Рис. 10.33. Результат выполнения запроса с сортировкой по столбцу `name_man`

	id	kodoftopic	text_message	name_man	date_answer
	1	3	Где можно найти мануал по MySQL?	Zero	2006-01-05

Рис. 10.34. Результат выполнения запроса с условием

```
имя_столбца2=значение2,
...
имя_столбцаN=значениеN
```

Давайте добавим в таблицу `TOPIC` информацию о новом разделе с именем `"Все о работе с MySQL"`, для этого достаточно выполнить следующий запрос:

```
INSERT INTO TOPIC SET name=
'Все о работе с  MySQL'
```

А теперь добавим пустую тему для этого раздела, для чего выполните следующий запрос, только не забудьте посмотреть идентификационный номер только что добавленного раздела, у меня он равняется `5`:

```
INSERT INTO TOPIC
```

Обратите внимание на то, что текст в условии взят в кавычки, это является обязательным действием.

```
SET
    kodofrazdel=5,
    name='Как добавить новую запись в таблицу?',
    name_creator='Dragon'
```

Когда необходимо изменить уже существующую информацию в таблице, используется команда UPDATE, общая схема построения которой следующая:

```
UPDATE имя_таблицы
    SET
        имя_столбца1=новое_значение1,
        имя_столбца2=новое_значение2,
        ...
        имя_столбцаN=новое_значениеN
WHERE  Условие, при котором меняется значение записей таблицы
```

Давайте изменим имя пользователя BigMan на ManOfHaos и его роль с user на admin, для этого выполните следующий запрос:

```
UPDATE USERS SET name='ManOfHaos', role='admin' WHERE id=2
```

Стоит отметить, что пароль при этом будет не затронут и останется старым. А теперь вернем пользователю его старую роль user, для этого выполните следующий запрос:

```
UPDATE USERS SET role='user' WHERE id=2
```

Когда необходимо удалить информацию из таблицы, используется команда DELETE, общая схема построения которой представлена ниже:

```
DELETE FROM имя_таблицы
WHERE Условие, по которому будет осуществлено удаление
```

Давайте удалим тему с именем "Как добавить новую запись в таблицу?", для этого выполните следующий запрос, только не забудьте посмотреть идентификационный номер данной темы, у меня он равняется 6:

```
DELETE FROM TOPIC WHERE id=6
```

Конечно, приведенной в данном разделе информации не достаточно, чтобы почувствовать себя экспертом в языке запросов SQL, если у вас возникло желание углубиться в SQL, введите в любом поисковике фразу "Учебник по SQL", поверьте, вам будет из чего выбрать. Можно также посетить сайт **www.sql-ex.ru**, где вы не только познакомитесь с тонкостями языка SQL, но и проверите свои знания с помощью специальных упражнений.

10.7. Кодирование

Создайте виртуальный хост **forum.ru**, и в нем папку www, далее перезапустите Денвер — только после этого новый хост станет доступен. Все файлы, разработанные в данном разделе, необходимо сохранять в forum.ru\www.

10.7.1. Подключаемся к базе

Подключение к MySQL — одно из первых действий, которое нужно совершить в программе, работающей с данной СУБД. Для этого используется следующая функция:

```
mysql_connect(hostname, username, password)
```

где `hostname` — имя хоста, чаще всего указывается localhost, т. к. обычно MySQL и PHP-скрипт находятся на одном и том же сервере.

`username` — имя пользователя, под которым будет осуществлено подключение к серверу.

`password` — пароль пользователя.

Если произвести подключение не удалось, то функция вернет `FALSE`, поэтому очень часто `mysql_connect()` используется следующим образом:

```
mysql_connect(hostname, username, password)
or die("Не удалось подключиться к базе")
```

После того как подключение к MySQL осуществлено, необходимо выбрать базу данных, с которой будет производиться работа, для этого используется следующая функция:

```
mysql_db(database_name)
```

где `database_name` — имя базы данных.

Если `mysql_db()` была выполнена успешно, то эта функция вернет `TRUE`, в противном случае — `FALSE`. Для `mysql_db()` также часто используют защиту в виде `die`.

На рис. 10.35 вы можете увидеть схему работы этих двух функций.

Приступаем к кодингу, создайте файл connect.php (листинг 10.1).

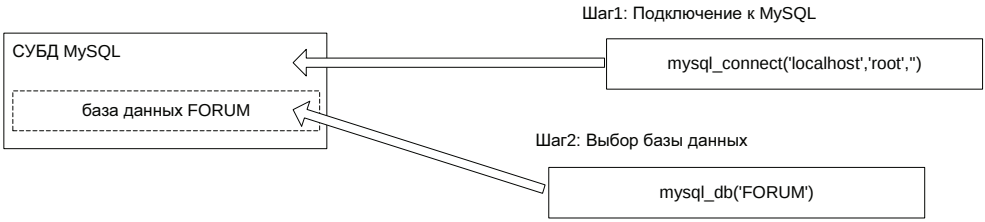


Рис. 10.35. Схема работы функций `mysql_connect()` и `mysql_db()`

connect.php будет модулем подключения к базе данных FORUM, именно он будет использоваться во всех скриптах, входящих в данный проект. Главное преимущество в его применении заключается в следующем: чтобы изменить параметры работы с MySQL, например, после закачки на реальный хост, достаточно будет изменить пару строчек в connect.php, и форум будет успешно функционировать.

Листинг 10.1. Файл connect.php

```
1   <?php
2   //хостинг
3   $sqlhost="localhost";
4   //имя пользователя
5   $sqluser="root";
6   //пароль
7   $sqlpass="";
8   //имя базы данных
9   $db="FORUM";
10
11  //Подключаемся к MySQL
12* mysql_connect($sqlhost, $sqluser,$sqlpass)or die("MySQL не
    доступен! ".mysql_error());
13  //Подключаемсся к базе данных
14* mysql_select_db($db)or die("Нет соединения с базой данных!
    ".mysql_error());
15  ?>
```

Все параметры подключения (имя хоста, имя пользователя, его пароль и имя базы данных) сохранены в специальных переменных: `$sqlhost`, `$sqluser`, `$sqlpass`, `$db`. Для Денвера имя хоста будет `localhost`, имя пользователя — `root`, а пароль будет пустым.

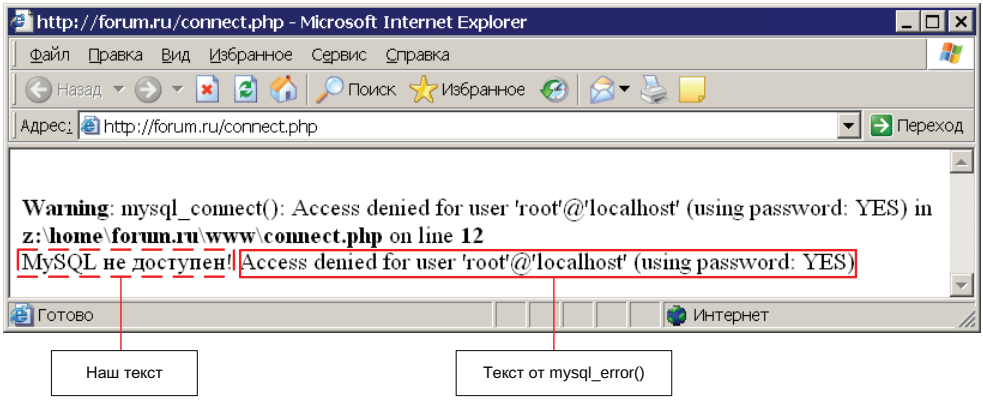


Рис. 10.36. Функция `mysql_error()` в действии

В строке **12** осуществляется подключение к MySQL, в строке **14** выбирается база данных, с которой будет осуществлена работа. В обоих случаях используется защитная конструкция `die`, которой в качестве параметра передается как текст, так и результат выполнения функции `mysql_error()`. Синтаксис `mysql_error()`:

```
mysql_error()
```

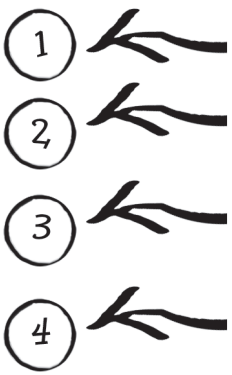
Данная функция возвращает информацию об ошибке для последней совершенной операции с MySQL. Например, если строка **12**:

```
mysql_select_db($db)or die("MySQL не доступен! ".mysql_error());
```

выполнится с отрицательным результатом (`FALSE`), т. е. подключиться к базе данных не удастся, то вы увидите следующее — рис. 10.36.

10.7.2. Модуль авторизации

Данный модуль очень похож на те, которые были разработаны для авторизации ранее. Логика его очень проста:



1. Пользователю выводится форма авторизации.
2. Далее введенные им данные обрабатываются.
3. Если авторизация успешна, то в сессию пользователя записывается информация об этом факте, а также о логине и роле пользователя.
4. В противном случае авторизация считается непройденной.

Создайте файл login.php (листинг 10.2).

Листинг 10.2. Файл login.php

```
1   <?php
2   //Данный модуль возвращает в $_SESSION['autorized'] значение TRUE,
3   //если авторизация пройдена
4
5   //Начинаем сессию
6   session_start();
7*  //Проверяем, как запущен скрипт - обработчик? или как форма для
    авторизации?
8   if (!isset($_POST['enter']))
```

```
{
 //Выводим форму авторизации
 ?>
 <form method='post' action=''>
 Авторизация на форуме<BR>
 имя:<input type='text' name='name' value=''><BR>
 пароль:<input type='password' name='pass'><BR>
 <input name='enter' type='submit' value='Войти'>
 <?php
}
//Если как обработчик, то пытаемся авторизовать пользователя
else
{
 //Проверяем, ввел ли пользователь имя и пароль
 if ($_POST['name']!='' and $_POST['pass']!='')
 {
    //Защита от взлома
    $safe_name=mysql_escape_string($_POST['name']);
    $safe_pass=mysql_escape_string($_POST['pass']);
    //Преобразуем пароль в хеш
    $safe_pass=md5($safe_pass);
    //Подключаемся к MySQL и базе данных
    require_once('connect.php');
    //Формируем запрос
    $sql="SELECT name,pass,role FROM USERS WHERE
name='".$safe_name."' and pass='".$safe_pass."'";
    //Получаем результат запроса в переменную $result
    $result=mysql_query($sql);
    //Проверяем, есть ли такой пользователь
    if (!mysql_num_rows($result))
        //Если такого пользователя нет, то отказываем ему в доступе
        die("Неверный логин или пароль <a href='index.php'>
Назад!</a>");
    //Иначе записываем факт авторизации в сессию
    else
    {
          //Получаем результат запроса
          $line=mysql_fetch_row($result);
          //Записываем факт авторизации в сессию
```

```
46          $_SESSION['autorized']=true;
47          //Сохраняем имя пользователя
48          $_SESSION['name']=$_POST['name'];
49          //Сохраняем роль пользователя
50          $_SESSION['role']=$line[2];
51          //Выводим пользователю информацию, что он был авторизован
52*         echo "Авторизация прошла успешно! <a
href=index.php>Вернуться в форум</a>";
53      }
54  }
55  //Если пользователь не ввел данные
56  else
57  {
58          //Отказываем ему в доступе
59*         die("Неправильный логин или пароль <a href='index.php'>
Назад!</a>");
60  }
61  }
62  ?>
```

В строке **6** происходит запуск сессии, т. к. информация об авторизации будет записываться в нее. В строке **8** проверяется условие `(!isset($_POST['enter']))`, т. к. сам скрипт работает в двух режимах:

Он просто выводит форму авторизации, когда условие `(!isset($_POST['enter']))` верно.

Осуществляет авторизацию пользователя, если он ввел данные в форму и нажал кнопку **Войти**.

В строке **23** происходит проверка, ввел ли пользователь логин и пароль с использованием логического оператора `and`, позволяющего строить логическое выражение, обе части которого должны быть равны `TRUE` — именно в этом случае условие считается верным.

В строках **26** и **27** данные, введенные пользователем, обрабатываются с помощью защитной функции `mysql_escape_string()`, которая позволяет обезопасить программу от взлома, ее синтаксис представлен ниже:

```
mysql_escape_string(строка);
```

Эта функция экранирует специальные символы (добавляет перед ними слэши), используемые в SQL-запросе, т. е. заменяет их на альтернативные — безопасные варианты (табл. 10.7).

Таблица 10.7. Список опасных символов для SQL-запроса

Специальный символ	После обработки функцией mysql_escape_string
\n	\\n
\r	\\r
\	\\
'	\'
"	\"

Такая замена необходима для предотвращения атак типа SQL injection (дословно "SQL-инъекция"), которые заключаются в том, что злоумышленник пытается изменить SQL-запрос, используемый в PHP-скрипте, таким образом, чтобы получить доступ к базе данных и возможность совершить над ней нежелательные действия.

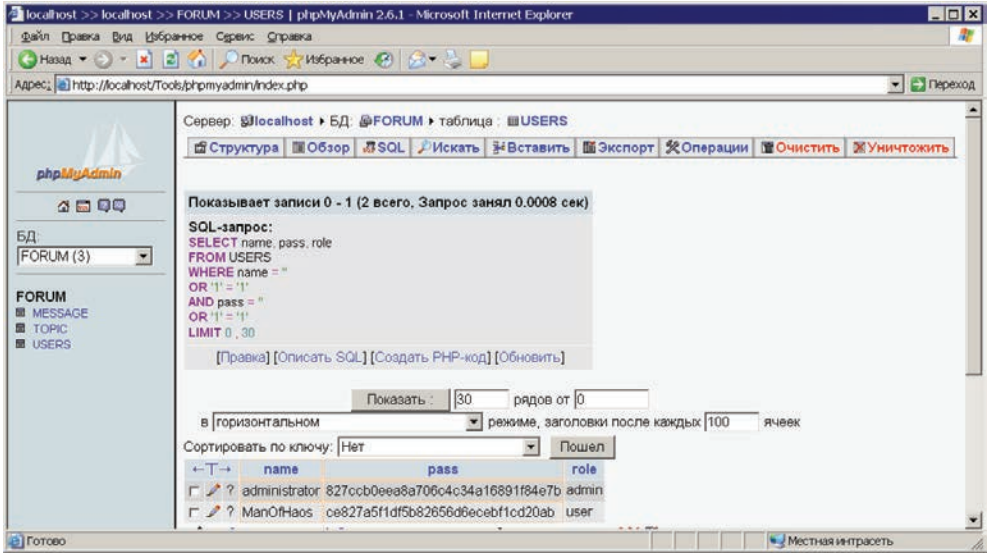
Рассмотрим небольшой пример. Запустите phpMyAdmin и выполните следующий запрос:

```
SELECT name,pass,role FROM USERS
WHERE name='' or '1'='1' and pass='' or '1'='1'
```

Результат вы можете видеть на рис. 10.37.

Таким образом, не зная ни логина, ни пароля, нам стало доступно все содержимое таблицы `USERS`, поэтому старайтесь всегда использовать функцию

Рис. 10.37. Пример SQL injection



mysql_escape_string(), иначе злоумышленнику не составит труда подвергнуть вас атаки типа SQL injection.

Замечание

Чтобы больше узнать о SQL-инъекциях, введите в любом поисковом сервисе фразу "описание SQL injection".

Возвращаемся к login.php, в строке **29** пароль, введенный пользователем, преобразуется в хеш. Затем, в строке **31**, подключается модуль соединения с базой данных connect.php. В строке **33** формируется SQL-запрос:

```
SELECT name,pass FROM USERS WHERE
name='".$safe_name."' and
pass='".$safe_pass."'";
```

который выберет имя пользователя и его роль из таблицы USERS для записи, у которой совпадает пароль и логин, введенные пользователем. В запросе используются два условия, объединенных логическим оператором and, это означает, что они оба должны быть верными. Схема данного запроса представлена на рис. 10.38.

В строке **35** происходит выполнение запроса с помощью функции mysql_query(), синтаксис которой представлен ниже:

```
mysql_query(SQL-запрос)
```

Данная функция отправляет MySQL-запрос, переданный в качестве параметра. Если это запрос типа SELECT и его выполнение успешно, то функция возвращает указатель на результат, в противном случае — FALSE. Когда работа осуществляется с запросами типа INSERT, UPDATE или DELETE, то функция в случае успеха возвращает TRUE, в случае неудачи — FALSE.

Важно отметить, что если запрос составлен неправильно, то выполнение mysql_query() не вызовет ошибку, поэтому данная функция часто используется вместе с защитной конструкцией die. В противном случае, когда вы не получите желаемого результата, понять причину этого будет сложно, т. к. никаких поясняющих сообщений в браузере вы не увидите. Рассмотрим пример, в котором исполь-

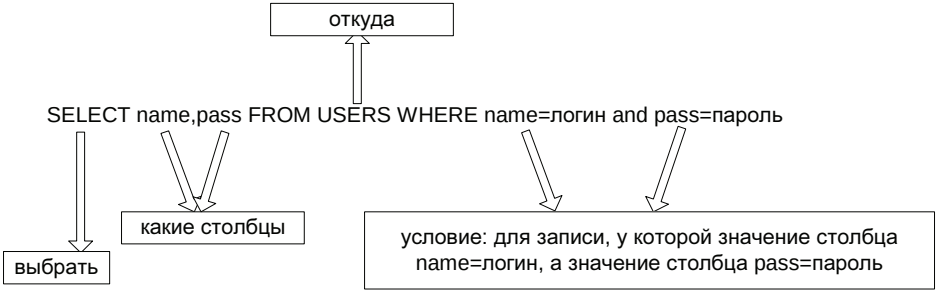


Рис. 10.38. Схема запроса, используемого для авторизации пользователя

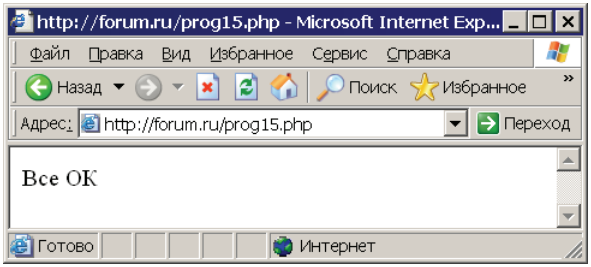


Рис. 10.39. Выполнение неправильно составленного SQL-запроса без использования защитной конструкции `die`

зуется некорректный SQL-запрос, т. к. в нем указано имя несуществующего столбца (prog15.php):

```
<?php
require_once("connect.php");
$sql="SELECT dragon FROM MESSAGE";
mysql_query($sql);
echo "Все OK";
?>
```

Результат работы скрипта изображен на рис. 10.39.

Как видите, сообщение об ошибке не возникло. А теперь выполните следующий скрипт (prog16.php):

```
<?php
require_once("connect.php");
$sql="SELECT dragon FROM MESSAGE";
mysql_query($sql) or die("Возникла ошибка: ".mysql_error());
echo "Все OK";
?>
```

Его результат работы представлен на рис. 10.40.

В данном случае мы можем видеть сообщение о некорректном запросе, что позволит более быстро исправить возникшую в программе ошибку.

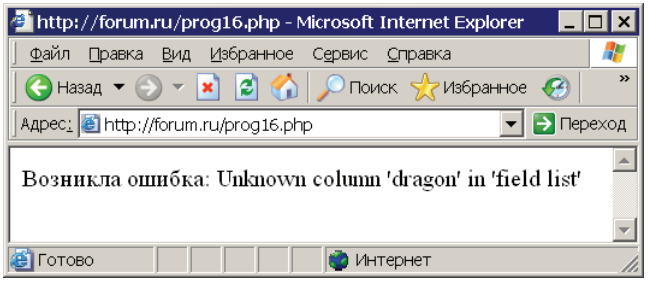


Рис. 10.40. Выполнение неправильно составленного SQL-запроса с использованием защитной конструкции `die`

Чтобы в дальнейшем использовать результат выполнения SQL-запроса, необходимо сохранить указатель на результат, возвращенный функцией `mysql_query()`, как правило, это делается следующим образом:

```
Имя переменной=mysql_query(SQL-запрос);
```

После этого уже переменная будет ссылаться на возвращенный результат. Для того чтобы получить его, надо воспользоваться одной из специальных функций, наиболее часто используемой из которых является `mysql_fetch_row()`, ее синтаксис:

```
mysql_fetch_row(ссылка на результат SQL-запроса);
```

Функция возвращает одну запись из результата SQL-запроса в виде массива, ключи которого пронумерованы по порядку, начиная с нуля. Например, в результате выполнения приведенной далее программы (prog17.php):

```
<?php
require_once("connect.php");
$sql="SELECT id,name FROM TOPIC";
$data=mysql_query($sql);
$line=mysql_fetch_row($data);
echo "Идентификационный номер: ".$line[0];
echo "<BR>Название: ".$line[1];
?>
```

вы увидите следующее — рис. 10.41.

А в результате выполнения следующей программы (prog18.php):

```
<?php
require_once("connect.php");
$sql="SELECT id,name FROM TOPIC";
$data=mysql_query($sql);
$line=mysql_fetch_row($data);
echo "Идентификационный номер: ".$line[0];
echo "<BR>Название: ".$line[1];
$line=mysql_fetch_row($data);
```

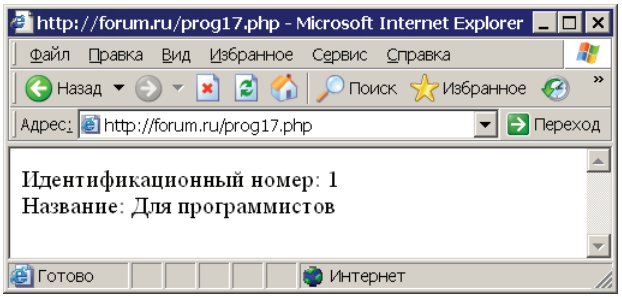


Рис. 10.41. Хотя запрос возвращает 4 записи, единократное обращение к функции `mysql_fetch_row()` позволяет отобразить только 1 запись

```
echo "<BR><BR>Идентификационный номер: ".$line[0];
echo "<BR>Название: ".$line[1];
$line=mysql_fetch_row($data);
echo "<BR><BR>Идентификационный номер: ".$line[0];
echo "<BR>Название: ".$line[1];
$line=mysql_fetch_row($data);
echo "<BR><BR>Идентификационный номер: ".$line[0];
echo "<BR>Название: ".$line[1];
?>
```

вы увидите следующее — рис. 10.42.

На рис. 10.43 показана схема работы функции `mysql_fetch_row()`.

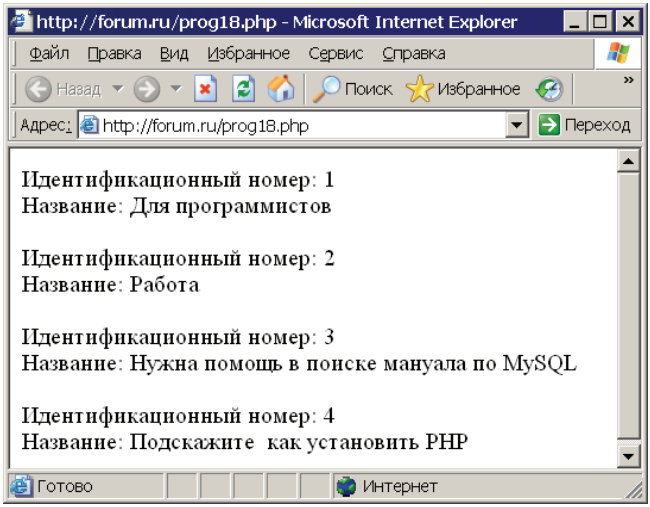


Рис. 10.42. Пример использования функции `mysql_fetch_row()`

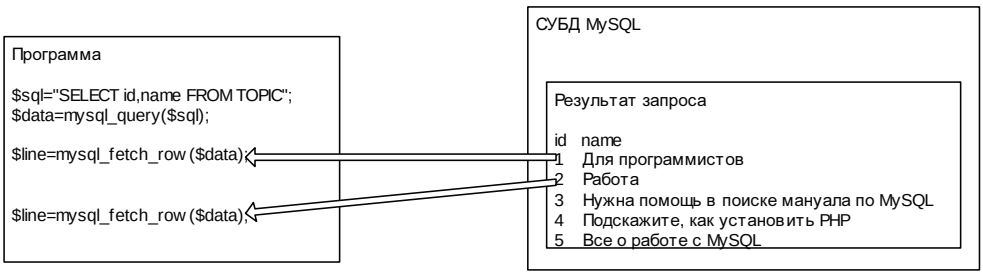


Рис. 10.43. Схема работы функции `mysql_fetch_row()`

Как правило, для получения результатов SQL-запроса, когда их количество больше 1, используют цикл `while` по следующей схеме:

```
while ($line=mysql_fetch_row(ссылка на результат SQL-запроса))
{
  обработка записей, полученных
  в результате выполнения запроса
}
```

Смысл условия цикла заключается в следующем: пока функция `mysql_fetch_row()` возвращает записи, условие верно, как только все записи возвращены, функция вернет `FALSE`, соответственно цикл прекратит работу. Теперь измените программу prog18.php, введя в нее рассмотренный цикл (см. код далее), и сохраните ее с именем prog19.php:

```
<?php
require_once("connect.php");
$sql="SELECT id,name FROM TOPIC";
$data=mysql_query($sql);
while ($line=mysql_fetch_row($data))
{
  echo "<BR>Идентификационный номер: ".$line[0];
  echo "<BR>Название: ".$line[1]."<BR>";
}
?>
```

Рассмотрим еще одну полезную функцию `mysql_num_rows()`, синтаксис:

```
mysql_num_rows(ссылка на результат SQL-запроса)
```

Эта функция возвращает количество записей, которое получилось в результате выполнения запроса. Например, следующий скрипт (prog20.php):

```
<?php
require_once("connect.php");
$sql="SELECT id,name FROM TOPIC";
$data=mysql_query($sql);
echo "В результате выполнения запроса<BR>";
echo "получилось ".mysql_num_rows($data). " записей";
?>
```

выведет на экран текст **В результате выполнения запроса получилось 5 записей** (рис. 10.44).

В строке **37** листинга 10.2 с помощью функции `mysql_num_rows()` осуществляется проверка, вернул ли что-то SQL-запрос, выполненный в строке **35**, если нет, то значит пользователь не зарегистрирован в системе и ему выводится сообщение

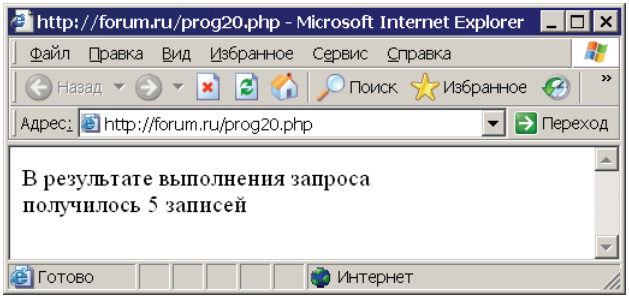


Рис. 10.44. Результат работы программы prog20.php

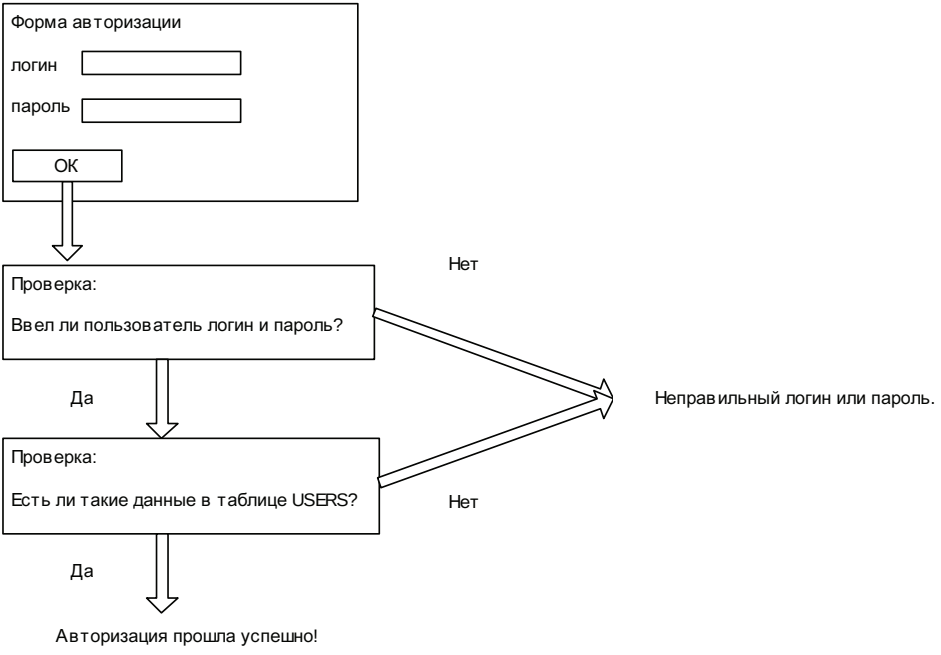


Рис. 10.45. Функционирование модуля login.php

"Неправильный логин и пароль". Если запрос вернул результат, то значит это зарегистрированный пользователь — его имя и роль записываются в сессию. Также в сессию записывается элемент `authorized` со значением `TRUE`.

Схема функционирования модуля login.php представлена на рис. 10.45.

10.7.3. Модуль logout.php

После того как пользователь авторизовался и пообщался на форуме, он может выйти. За этот процесс будет отвечать модуль logout.php (листинг 10.3).

Листинг 10.3. Файл logout.php

```
<?php
//Уничтожаем сессию
session_start();
session_unset();
session_destroy();
//Перенаправляем пользователя на index.php
header('location:index.php');
?>
```

Как это не парадоксально, но перед тем как начать удалять сессию, ее сначала нужно начать, иначе удалять будет нечего, потому что сведения о сессии будут потеряны.

10.7.4. Основной файл форума index.php

Центральным файлом форума является index.php (листинг 10.4). В нем будет объединена единой логикой работа всех модулей.

Листинг 10.4. Файл index.php

```
<?php
//Запускаем сессию
session_start();
//Подключаемся к MySQL и базе данных FORUM
require_once('connect.php');

//Если пользователь не авторизован,
//то выводим ссылку на login.php
if (!isset($_SESSION['autorized']))
{
  ?>
  <p align='right'>
  <a href='login.php'>Авторизация</a><BR>
```

```
14    </p>
15    <?php
16    $_SESSION['name']='guest';
17    $_SESSION['role']='user';
18  }
19  else
20  {
21    //Если пользователь авторизован, то сообщаем ему об этом
22    //и выводим ссылку на logout.php
23*   echo "<p align='right'>Вы авторизованы под ником:
    ".$_SESSION['name']."<BR>";
24    echo "<a href='logout.php'>Выход</a></p>";
25  }
26
27  //Если выполняется действие,
28  if (isset($_GET['act']))
29  {
30      //то подключаем модуль, отвечающий за это,
31      //и совершаем действие
32      require_once('action.php');
33  }
34  //Иначе осуществляем простой вывод информации
35  else
36      require_once('show.php');
37  ?>
```

Работа скрипта начинается с проверки, авторизован ли пользователь. Если нет, то выводится ненавязчивая ссылка **Авторизация**, нажатие на которую приведет к запуску login.php, пользователь в этом случае будет иметь имя `guest` и роль `user` (см. строки **16** и **17**). Если же пользователь авторизован, то выводится текст **Вы авторизованы под ником** и ник пользователя, а также ссылка **Выход**, нажатие на которую приведет к запуску logout.php.

Далее в строке **28** осуществляется проверка, какой из модулей нужно подключать. Если пользователь совершает какое-то действие, например, создает тему, то в index.php методом `GET` будет передан параметр `act`, и осуществится подключение action.php, в противном случае будет подключен модуль show.php. Схема работы index.php представлена на рис. 10.46.

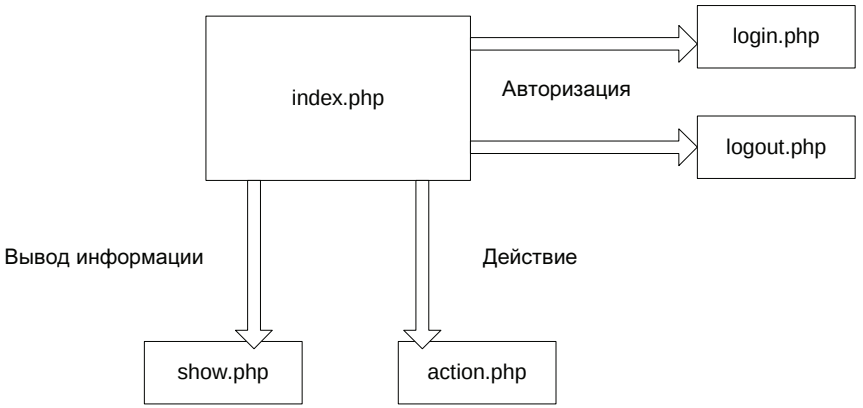


Рис. 10.46. Функционирование index.php

10.7.5. Модуль вывода информации show.php

Создайте файл show.php, его содержимое представлено в листинге 10.5.

Листинг 10.5. Файл show.php

```
<?php
//Запуск данного скрипта без параметра show, переданного
//в GET-строке, приведет к выводу разделов
if (!isset($_GET['show']))
{
    //Задаем SQL-запрос
    $sql="SELECT id, name FROM TOPIC WHERE kodofrazdel=0";
    //Выполняем его
    $data=mysql_query($sql);
    //Надпись "Список разделов"
    echo "<BIG><B>Список разделов</B></BIG><BR><BR>";
    //Выводим список всех разделов
    while($line=mysql_fetch_row($data))
    {
       ?>
       <table BORDER=1 cellpadding=20 width=100%>
```

```
17          <tr>
18             <td>
19             <?php
20*            //Ссылка на index.php только с параметром show=topic
21*            echo '<a href="?show=topic&numrazdel=
'.$line[0].'">'.$line[1]."</a>";
22             ?>
23             </td>
24          </tr>
25          </table>
26          <?php
27       }
28       //Больше ничего выполнять не стоит
29       exit;
30    }// end - if (!isset($_GET['show']))
31
32
33    //Если задан параметр show, то в зависимости от него
34    //выводим соответствующую информацию
35    switch ($_GET['show'])
36    {
37    //Если нужно вывести темы для выбранного раздела
38    case 'topic':
39         require_once('SHOW_MODULE/show_topic.php');
40         break;
41
42    //Если нужно вывести сообщения для выбранной темы
43    case 'message':
44         require_once('SHOW_MODULE/show_message.php');
45         break;
46
47    //Если нужно вывести форму создания темы
48    case 'add_topic':
49         require_once('SHOW_MODULE/show_addtopic.php');
50         break;
51
52    //Если нужно вывести форму редактирования темы
```

```
53  case 'edit_topic':
54       require_once('SHOW_MODULE/show_edittopic.php');
55       break;
56
57  //Если нужно удалить тему
58  case 'del_topic':
59       require_once('SHOW_MODULE/show_deltopic.php');
60       break;
61
62  //Если нужно вывести форму редактирования сообщения
63  case 'edit_message':
64       require_once('SHOW_MODULE/show_editmessage.php');
65       break;
66
67  //Если нужно удалить сообщение
68  case 'del_message':
69       require_once('SHOW_MODULE/show_delmessage.php');
70       break;
71  }//end - case
72  ?>
```

Как видите, разбив всю задачу на небольшие модули, мы добились того, что код каждого из них стал очень компактным, простым и понятным.

Вся работа данного модуля строится в зависимости от параметра `show`, переданного методом `GET`. Если его нет, то значит будет выводиться просто список разделов, для чего используется следующий SQL-запрос:

```
SELECT id, name FROM TOPIC WHERE
kodofrazdel=0
```

который можно описать как: выбрать все идентификационные номера и имена для записей, у которых столбец `kodofrazdel` имеет значение 0, из таблицы `TOPIC`. То есть, другими словами, выбрать все разделы.

Затем каждый раздел, возвращенный SQL-запросом, выводится как ссылка (см. строку **21** кода) следующего содержания: `?show=topic&numrazdel=идент. номер раздела`. Таким образом, нажатие на эту ссылку приведет к вызову скрипта show.php с параметром `show`, равным `topic`, и параметром `numrazdel`, равным идентификационному номеру выбранного раздела, в этом случае будет выполнен участок кода, заключенный между строками **38**—**40**, который принадлежит конструкции `switch()` (или, другими словами, будет вызван еще не описанный нами минимодуль show_topic.php).

Создайте папку SHOW_MODULE и в ней файл show_topic.php (листинг 10.6).

Листинг 10.6. Файл SHOW_MODULE/show_topic.php

```
<?php
//Задаем SQL-запрос
$sql="SELECT id, kodofrazdel, name, name_creator,
name_last_answer, date_last_answer FROM TOPIC WHERE
kodofrazdel=".$_GET['numrazdel']." ORDER BY date_last_answer";
//Выполняем его
$data=mysql_query($sql);
//Задаем SQL-запрос, который вернет имя выбранного
//пользователем раздела
$sql2="SELECT name FROM TOPIC WHERE id=".$_GET['numrazdel'];
//Выполняем его
$data2=mysql_query($sql2);
//Получаем результат - одна запись
$line2=mysql_fetch_row($data2);

//Выводим надпись
echo "<BIG><B>Список тем для ";
echo "раздела: ".$line2[0]."</B></BIG><BR><BR>";

//Кнопка для создания новой темы
?>
<p align='right'>
<form action="?show=add_topic&numrazdel=<?php echo
$_GET['numrazdel'];?>" method="post">
<input type="submit" value="Создать новую тему">
</form>
</p>

<?php
//Выводим заголовок для таблицы
?>
<table BORDER=1 cellpadding=3 width=100%>
<tr>
 <td width=60%>
```

```
 Название темы
 </td>
 <td width=10%>
 <font size=2>Автор</font>
 </td>
 <td width=30%>
 <font size=2>Последнее сообщение (Кто|Дата)</font>
 </td>
</tr>
</table>
<?php
//Выводим список всех тем для выбранного раздела
while($line=mysql_fetch_row($data))
{
?>
   <table BORDER=1 cellpadding=20 width=100%>
   <tr>
    <td width=60%>
    <?php
    //Это в виде ссылки, она на index.php
    //только с параметром message
    echo '<a
href="?show=message&numtopic='.$line[0].'">'.$line[2].'</a>' ;
    //Если это админ, то он может редактировать название темы
    //и удалять ее
    if ($_SESSION['role']=='admin')
    {
      ?>
      <form action="?show=edit_topic&numtopic=<? echo $line[0]?>"
method="post">
      <input type="submit" value="Изменить название">
      </form>
      <form action="?show=del_topic&numtopic=<? echo $line[0]?>"
method="post">
      <input type="submit" value="Удалить тему">
      </form>
      <?php
    }//end - if
```

```
67      ?>
68      </td>
69      <td width=10%>
70      <?php
71      //Имя создавшего тему
72      echo $line[3];
73      ?>
74      </td>
75      <td width=10%>
76      <?php
77      //Имя последнего ответившего
78      echo $line[4];
79      ?>
80      </td>
81      <td width=20%>
82      <?php
83      //Дата последнего ответа
84      echo $line[5];
85      ?>
86      </td>
87     </tr>
88     </table>
89     <?php
90  }//end - while
91  ?>
```

В строке **3** задается SQL-запрос на выбор всех тем для выбранного раздела, его идентификатор доступен через `$_GET['numrazdel']`. Сам запрос представлен далее:

```
SELECT id, kodofrazdel, name, name_creator,
       name_last_answer, date_last_answer
FROM TOPIC
WHERE kodofrazdel=идент. номеру выбранного
пользователем раздела
ORDER BY date_last_answer
```

Листинг show_topic.php получился довольно большим, но в основном это из-за HTML-кода.

В строке **5** происходит выполнение этого запроса и ссылка на результат сохраняется в переменной `$data`. В строке **8** формируется еще один SQL-запрос, предназначение которого — получить имя выбранного пользователем раздела, т. к. оно тоже понадобится. Сам запрос представлен далее:

```
SELECT name
FROM TOPIC
WHERE id=идентификационный номер выбранного пользователем раздела
```

В строке **10** происходит выполнение этого запроса, ссылка на результат сохраняется в переменной `$data2`, затем в строке **12** она сразу обрабатывается функцией

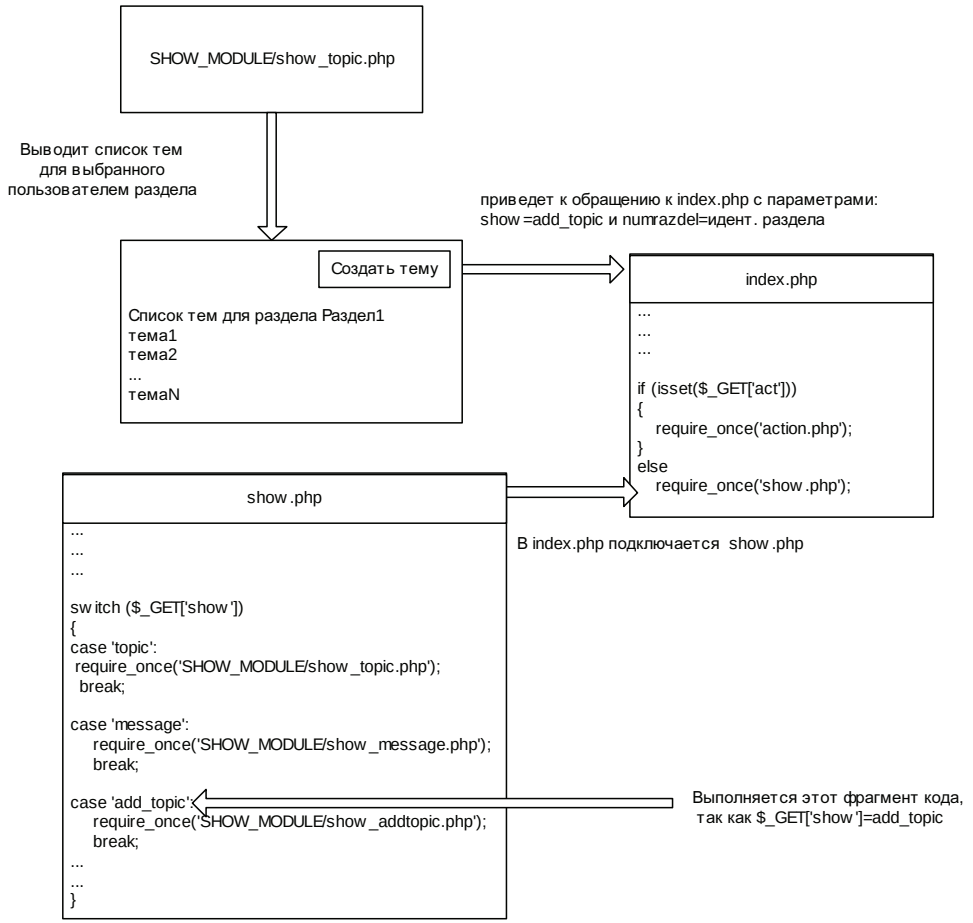


Рис. 10.47. Схема событий, которые произойдут после нажатия кнопки **Создать новую тему**

`mysql_fetch_row()` и результат этого действия — имя выбранного пользователем раздела — сохраняется в переменной `$line2`.

В строках **15** и **16** выводится текст `"Список тем для раздела"` и значение переменной `$line2`. А в строке **22** создается кнопка **Создать новую тему**, нажатие которой приведет к вызову файла index.php со следующей строкой параметров:

```
?show=add_topic&numrazdel=<?php echo $_GET['numrazdel'];
```

то есть это означает, что необходимо будет вывести форму создания новой темы для раздела с идентификационным номером, хранящимся в `$_GET['numrazdel']` (рис. 10.47).

В строке **56** осуществляется проверка роли пользователя, и если он имеет права администратора, то ему будет доступен дополнительный инструмент для работы с форумом, реализованный в виде двух кнопок: **Изменить название** и **Удалить тему**.

Создайте файл SHOW_MODULE/show_addtopic.php (листинг 10.7).

Листинг 10.7. Файл SHOW_MODULE/show_addtopic.php

```
1   <?php
2   echo "<H2>Создание темы</H2>";
3   ?>
4*  <form action="?act=add_topic&numrazdel=<?php echo
    $_GET['numrazdel']?>" method="post">
5   Название темы:<BR>
6   <input name="name_topic" type="text" value="">
7   <BR>
8   Текст сообщения:<BR>
9   <textarea name="message" cols=40 rows=5></textarea>
10  <BR>
11  <input type="submit" value="Создать тему">
12  </form>
```

Как видите, это обычная форма с полем ввода, именуемым `name_topic` и предназначенным для ввода названия темы, и полем для ввода многострочного текста с именем `message`, предназначенным для ввода сообщения для создаваемой темы. Также имеется кнопка **Ответить**, нажатие которой приведет к вызову файла index.php со строкой параметров:

```
?act=add_topic&numrazdel= номер выбранного пользователем раздела
```

Запустите форум (введите в браузере `forum.ru`), выберите любой раздел и нажмите кнопку **Создать новую тему**, вы увидите следующее — рис. 10.48.

Создайте файл SHOW_MODULE/show_edittopic.php (листинг 10.8).

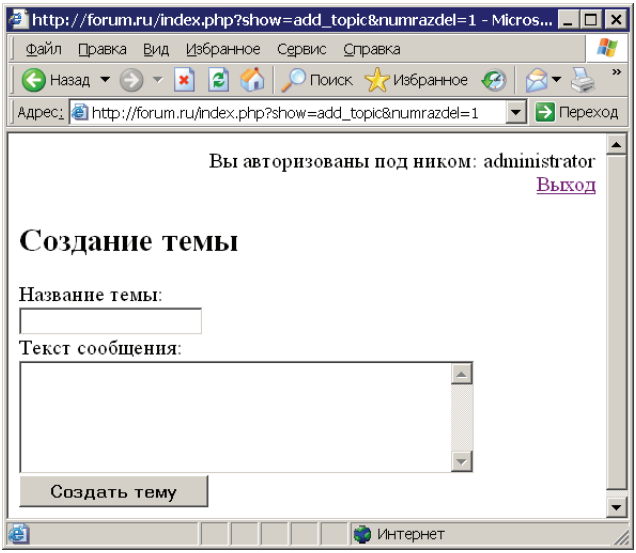


Рис. 10.48. Форма добавления новой темы

Листинг 10.8. Файл SHOW_MODULE/show_edittopic.php

```
1  <?php
2  echo "<H2>Редактирование темы</H2>";
3  //Задаем SQL-запрос, который выберет данные по теме
4  $sql="SELECT id, name FROM TOPIC WHERE `id`=".$_GET['numtopic'];
5  //Выполняем его
6  $data=mysql_query($sql) or die(mysql_error());
7  //Получаем результат запроса - одна запись
8  $line=mysql_fetch_row($data);
9  ?>
10 <!--Создаем форму для редактирования темы-->
11* <form action="?act=edit_topic&numtopic=<?php echo $line[0];?>"
   method="post">
12 Название темы:<BR>
13 <input name="name_topic" type="text" value="<?=$line[1]?>" size=40>
14 <BR>
15 <input type="submit" value="Изменить">
16 </form>
```

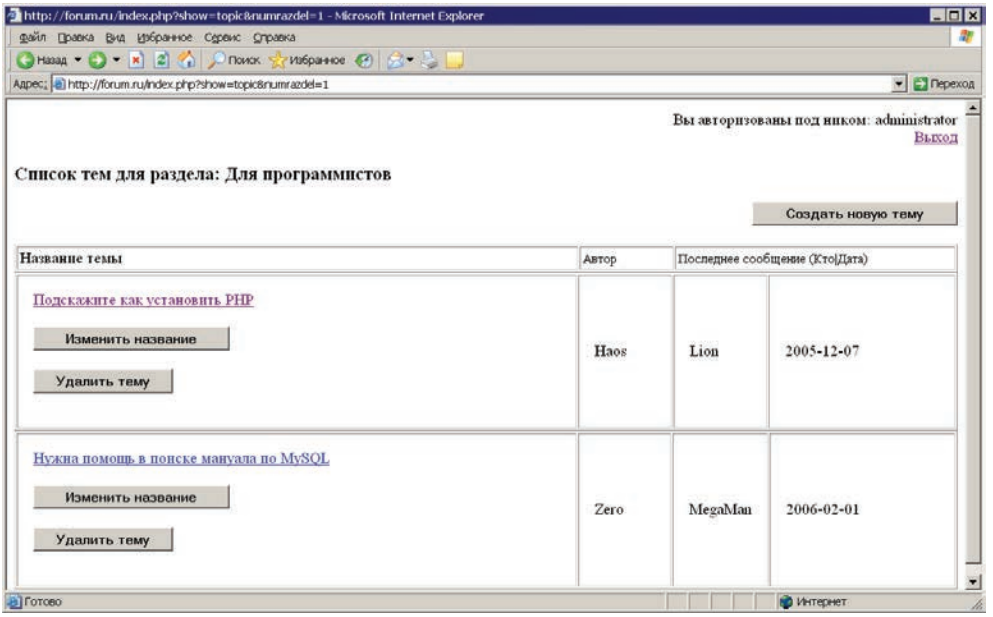


Рис. 10.49. Просмотр списка тем в режиме администратора

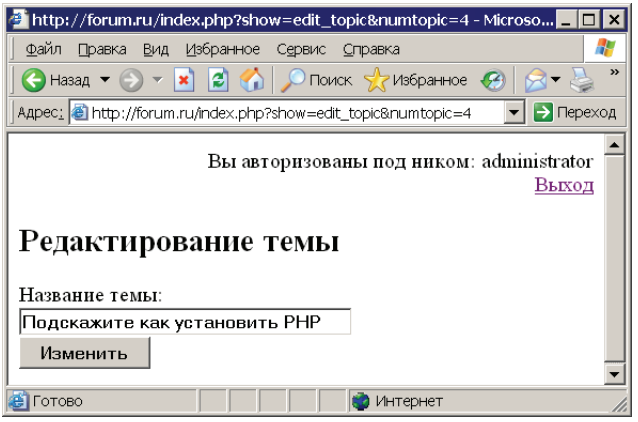


Рис. 10.50. Форма для редактирования названия темы

Этот мини-модуль выводит обычную форму с единственным элементом — полем ввода, в котором уже будет находиться название темы, выбранное пользователем для редактирования. После нажатия кнопки **Изменить** произойдет обращение к файлу index.php с передачей ему следующей строки параметров:

```
?act=edit_topic&numtopic=идент. номер темы, которая подверглась редактированию
```

На рис. 10.49 показан форум, в котором отображается список тем в режиме администратора, а на рис. 10.50 изображена форма редактирования названия темы.

Создайте файл SHOW_MODULE/show_deltopic.php (листинг 10.9).

Листинг 10.9. Файл SHOW_MODULE/show_deltopic.php

```
1   <?php
2   echo "<H2>Удаление темы</H2>";
3   //Задаем SQL-запрос, который выберет данные по удаляемой теме
4   $sql="SELECT id, name FROM TOPIC WHERE id=".$_GET['numtopic'];
5   //Выполняем его
6   $data=mysql_query($sql) or die(mysql_error());
7   //Получаем результат - одна запись
8   $line=mysql_fetch_row($data);
9   //Выводим надпись
10* echo "Вы действительно хотите удалить тему: <B>".$line[1]."</B>,
    и все ее сообщения?";
11  ?>
12  <!--Создаем форму для удаления темы-->
13* <form action="?act=del_topic&numtopic=<?php echo $line[0]?>"
    method="post">
14  <input type="submit" value="Да        ">
15  </form>
16  <form action="index.php" method="post">
17  <input type="submit" value="Отмена">
18  </form>
```

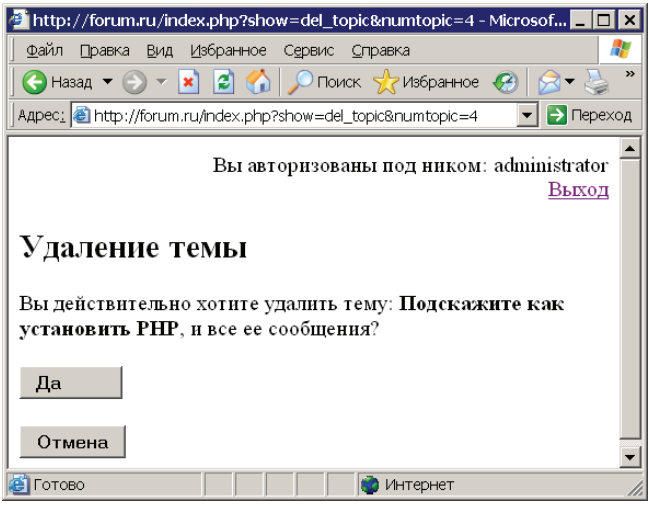


Рис. 10.51. Форма подтверждения удаления выбранной темы

Этот мини-модуль выводит запрос на подтверждение удаления темы, а также две кнопки **Да** и **Отмена** каждая из них расположена в отдельной форме (рис. 10.51). Нажатие на кнопку **Да** приведет к вызову файла index.php с передачей ему следующей строки параметров:

```
?act=del_topic&numtopic= идент. номер темы, которая подверглась редактированию
```

Нажатие же кнопки **Отмена** осуществит отмену действия.

Создайте файл SHOW_MODULE/show_message.php (листинг 10.10). Он будет отвечать за вывод сообщений для выбранной пользователем темы.

Листинг 10.10. Файл SHOW_MODULE/show_message.php

```
<?php
//Задаем SQL-запрос, который выберет все сообщения для заданной темы
$sql="SELECT id, text_message, name_man, date_answer".
     " FROM MESSAGE WHERE kodoftopic=".$_GET['numtopic'].
     " ORDER BY date_answer";
//Выполняем его
$data=mysql_query($sql);
//Задаем SQL-запрос, который вернет имя выбранной пользователем темы
$sql2="SELECT name FROM TOPIC WHERE id=".$_GET['numtopic'];
//Выполняем его
$data2=mysql_query($sql2);
//Получаем результат - одна запись
$line2=mysql_fetch_row($data2);
//Выводим надпись
echo "<BIG><B>Список сообщений для ";
echo "темы: ".$line2[0]."</B></BIG><BR><BR>";
//Выводим заголовок для таблицы
?>
<table BORDER=1 cellpadding=3 width=100%>
<tr>
 <td width=70%>
 Сообщение
 </td>
 <td width=10%>
 <font size=2>Автор</font>
 </td>
 <td width=20%>
```

```
28   <font size=2>Дата</font>
29   </td>
30  </tr>
31  </table>
32  <?php
33  //Выводим список всех сообщений для выбранной темы
34  while($line=mysql_fetch_row($data))
35  {
36  ?>
37  <table BORDER=1 cellpadding=20 width=100%>
38  <tr>
39   <td width=70%>
40   <?php
41   echo $line[1];
42   //Если это админ, то он может редактировать сообщение и удалять его
43   if ($_SESSION['role']=='admin')
44   {
45     ?>
46*    <form action="?show=edit_message&nummessage=<?=$line[0]?>"
     method="post">
47     <input type="submit" value="Редактировать сообщение">
48     </form>
49*    <form action="?show=del_message&nummessage=<?=$line[0]?>"
     method="post">
50     <input type="submit" value="Удалить сообщение">
51     </form>
52     <?php
53   }//end - if
54   ?>
55   </td>
56   <td width=10%>
57   <?php
58   //Имя пользователя, создавшего сообщение
59   echo $line[2];
60   ?>
61   </td>
62   <td width=20%>
63   <?php
```

```
64    //Дата размещения сообщения
65    echo $line[3];
66    ?>
67    </td>
68  </tr>
69  </table>
70  <?php
71  }//end - while
72  ?>
73* <form action="?act=add_message&numtopic=<?php echo
    $_GET['numtopic']?>" method="post">
74  Текст сообщения:<BR>
75  <textarea name="message" cols=40 rows=5></textarea>
76  <BR>
77  <input type="submit" value="Ответить">
78  </form>
```

Этот мини-модуль выбирает список всех сообщений для темы, идентификатор которой передан в `$_GET['numtopic']`, сообщения сортируются по дате размещения, т. е. последние будут снизу. Под последним сообщением будет также размещена форма для добавления нового сообщения с кнопкой **Ответить**, нажатие которой приведет к вызову файла index.php с передачей ему следующей строки параметров:

```
?act=add_message&numtopic= идент. номер темы, для которой добавляется
сообщение
```

В строке **43** осуществляется проверка роли пользователя, и если он имеет права администратора, то ему будет доступен дополнительный инструмент для работы с сообщениями, реализованный в виде двух кнопок: **Редактировать сообщение** и **Удалить сообщение** (рис. 10.52).

Создайте файл SHOW_MODULE/show_editmessage.php (листинг 10.11).

Листинг 10.11. Файл SHOW_MODULE/show_editmessage.php

```
1   <?php
2   echo "<H2>Редактирование сообщения</H2>";
3   //SQL-запрос, который выберет данные по теме
4*  $sql="SELECT id, text_message FROM MESSAGE WHERE
    id=".$_GET['nummessage'];
5   //Выполняем запрос
6   $data=mysql_query($sql) or die(mysql_error());
7   //Получаем результат - одна запись
```

```
8   $line=mysql_fetch_row($data);
9   ?>
10* <form action="?act=edit_message&nummessage=<?php echo $line[0]?>"
    method="post">
11  Текст сообщения:<BR>
12  <textarea name="message" cols=40 rows=5><?=$line[1]?></textarea>
13  <BR>
14  <input type="submit" value="Изменить">
15  </form>
```

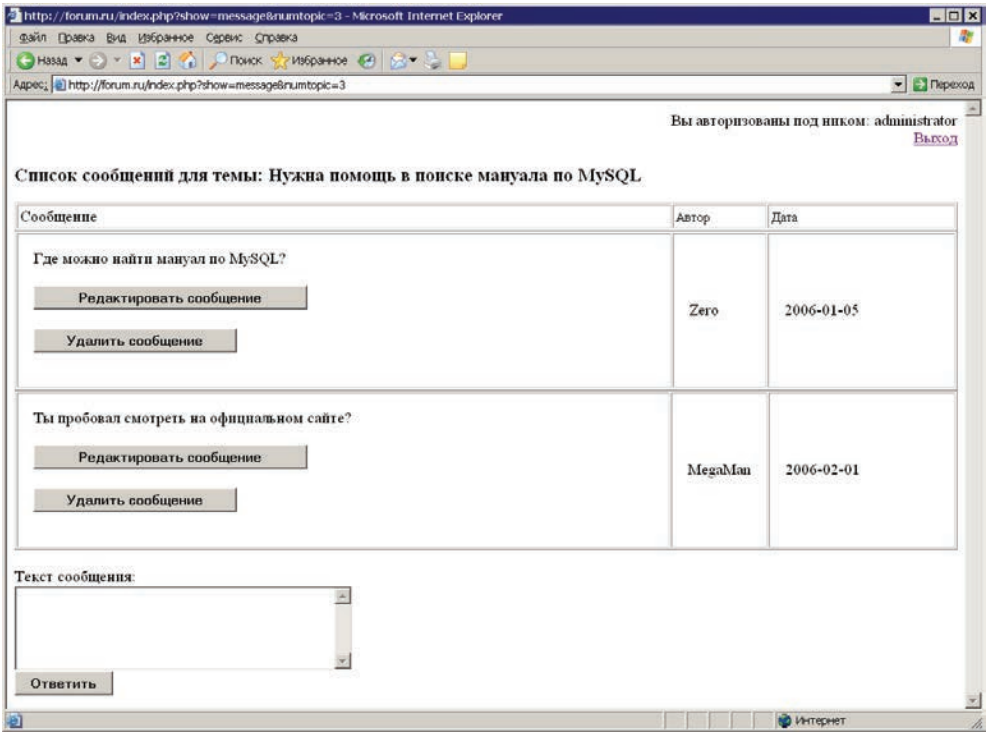


Рис. 10.52. Работа с сообщениями

Этот мини-модуль представляет собой полную аналогию с SHOW_MODULE/show_edittopic, только сообщение редактируется в поле для ввода многострочного текста и обработчиком формы является:

```
?act=edit_message&nummessage= идент. номер редактируемого сообщения
```

На рис. 10.53 вы можете увидеть только что созданный модуль в работе.

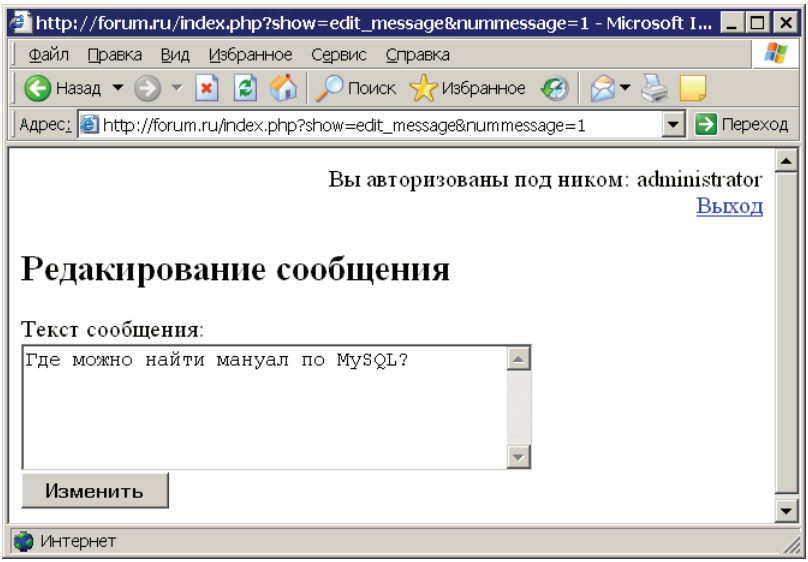


Рис. 10.53.
Редактирование сообщения

И последний модуль из серии SHOW_MODULE — это show_delmessage.php (листинг 10.12). Он предназначен для удаления сообщения.

Листинг 10.12. Файл SHOW_MODULE/show_delmessage.php

```
1   <?php
2   echo "<H2>Удаление сообщения</H2>";
3   //SQL-запрос, который выберет идент. номер удаляемого сообщения
4   $sql="SELECT id FROM MESSAGE WHERE id=".$_GET['nummessage'];
5   //Выполняем запрос
6   $data=mysql_query($sql) or die(mysql_error());
7   //Получаем результат - одна запись
8   $line=mysql_fetch_row($data);
9   //Выводим надпись
10  echo "Вы действительно хотите удалить выбранное сообщение?";
11  ?>
12* <form action="?act=del_message&nummessage=<?php echo $line[0]?>"
    method="post">
13  <input type="submit" value="Да      ">
```

```
</form>
<form action="index.php" method="post">
<input type="submit" value="Отмена">
</form>
```

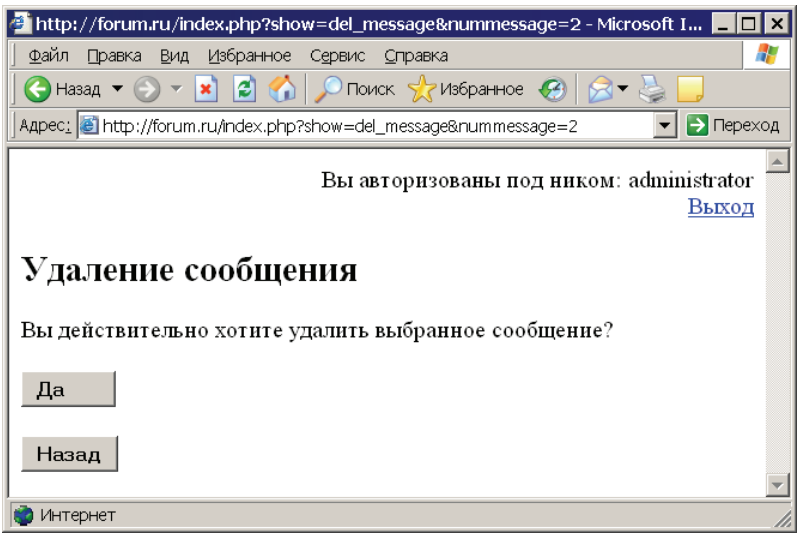


Рис. 10.54. Удаление сообщения

Этот мини-модуль представляет собой полную аналогию с SHOW_MODULE/show_deltopic, только обработчиком формы является:

```
?act=del_message&nummessage= идент. номер редактируемого сообщения
```

На рис. 10.54 вы можете увидеть только что созданный модуль в работе.

Листинг 10.13. Файл action.php

```
<?php
//Добавление темы
if ($_GET['act']=='add_topic')
{
```

```
5       //Обрабатываем название темы в целях безопасности
6       $safe_topic=mysql_escape_string($_POST['name_topic']);
7       //SQL-запрос для добавления темы
8*      $sSQL="INSERT INTO TOPIC SET
kodofrazdel=".$_GET['numrazdel']."", name='".$safe_topic."',
name_creator='".$_SESSION['name']."',
date_last_answer='".date('Y-m-d')."'";
9       //Выполняем запрос
10      mysql_query($sSQL)or die(mysql_error());
11      //Обрабатываем текст сообщения в целях безопасности
12      $safe_message=mysql_escape_string($_POST['message']);
13      //Определяем номер созданной темы
14      $id=mysql_insert_id();
15      //SQL-запрос, добавляющий сообщение для вновь созданной темы
16*     $sSQL="INSERT INTO MESSAGE  SET kodoftopic=".$id.",
text_message='".$safe_message."', name_man='".$_SESSION['name']."',
date_answer='".date('Y-m-d')."'";
17      //Выполняем запрос
18      mysql_query($sSQL)or die(mysql_error());
19      //Выводим надпись и ссылку на список тем для текущего раздела
20      echo "Тема создана<BR>";
21*     echo "<a href='index.php?show=topic&numrazdel=
".$_GET['numrazdel']."'>";
22      echo "Назад к списку тем</a>";
23  }
24
25  //Изменение названия темы
26  if ($_GET['act']=='edit_topic')
27  {
28      //Обрабатываем название темы в целях безопасности
29      $safe_topic=mysql_escape_string($_POST['name_topic']);
30      //SQL-запрос, который изменит название темы
31*     $sSQL="UPDATE TOPIC SET name='".$safe_topic."'
    WHERE id=".$_GET['numtopic'];
32      //Выполняем запрос
33      mysql_query($sSQL)or die(mysql_error());
34      //Выбираем код раздела, чтобы можно было перенаправить
35      //пользователя на список тем для этого раздела
36*     $sSQL="SELECT kodofrazdel FROM TOPIC
    WHERE id=".$_GET['numtopic'];
```

```
37      //Выполняем запрос
38      $data=mysql_query($sSQL);
39      //Получаем результат - одна запись
40      $line=mysql_fetch_row($data);
41      //Выводим надпись и ссылку на список тем для текущего раздела
42      echo "Название темы изменено<BR>";
43      echo "<a href='index.php?show=topic&numrazdel=$line[0]'>";
44      echo "Назад к списку тем</a>";
45  }
46
47  //Удаление темы и всех ее сообщений
48  if ($_GET['act']=='del_topic')
49  {
50      //Выбираем код раздела, чтобы можно было вернуться в него
51*     $sSQL="SELECT kodofrazdel FROM TOPIC
    WHERE id=".$_GET['numtopic'];
52      $data=mysql_query($sSQL);
53      //Получаем результат - одна запись
54      $line=mysql_fetch_row($data);
55      //Удаляем все сообщения для выбранной темы
56      $sSQL="DELETE FROM MESSAGE WHERE kodoftopic=".$_GET['numtopic'];
57      mysql_query($sSQL)or die(mysql_error());
58      //Удаляем саму тему
59      $sSQL="DELETE FROM TOPIC WHERE id=".$_GET['numtopic'];
60      mysql_query($sSQL)or die(mysql_error());
61     //Выводим надпись и ссылку на список тем для текущего раздела
62      echo "Тема удалена<BR>";
63*     echo "<a href='index.php?show=topic&numrazdel=$line[0]'>Назад
    к списку тем</a>";
64  }
65
66  //Добавление нового сообщения
67  if ($_GET['act']=='add_message')
68  {
69     //Обрабатываем текст в целях безопасности
70     $safe_message=mysql_escape_string($_POST['message']);
71     //Запрос для добавления сообщения
```

```
72*     $sSQL="INSERT INTO MESSAGE  SET kodoftopic=".$_GET['numtopic'].",
text_message='".$safe_message."', name_man='".$_SESSION['name']."',
date_answer='".date('Y-m-d')."'";
73      //Выполняем запрос
74      mysql_query($sSQL)or die(mysql_error());
75
76      //Теперь добавляем информацию об имени посетителя и дате
77      //размещаемого сообщения для темы, которой принадлежит сообщение
78*     $sSQL="UPDATE TOPIC SET name_last_answer='".$_SESSION['name']."',
date_last_answer='".date('Y-m-d')."' WHERE id=".$_GET['numtopic'];
79      mysql_query($sSQL)or die(mysql_error());
80      //Выводим надпись и ссылку на список сообщений для текущей темы
81      echo "Ответ принят<BR>";
82*     echo "<a href='index.php?show=message&numtopic=
".$_GET['numtopic']."'>";
83      echo "Назад к обсуждению темы</a>";
84  }
85
86  //Изменение сообщения
87  if ($_GET['act']=='edit_message')
88  {
89      //Обрабатываем название в целях безопасности
90      $safe_message=mysql_escape_string($_POST['message']);
91      //Меняем текст сообщения
92      $sSQL="UPDATE MESSAGE SET text_message='".$safe_message."'
WHERE id=".$_GET['nummessage'];
93*     mysql_query($sSQL)or die(mysql_error());
94      //Выбираем код темы, чтобы можно было перенаправить
95      //пользователя на список сообщений для этой темы
96      $sSQL="SELECT kodoftopic FROM MESSAGE
WHERE id=".$_GET['nummessage'];
97*     $data=mysql_query($sSQL);
98      //Получаем результат - одна запись
99      $line=mysql_fetch_row($data);
100     //Выводим надпись и ссылку на список сообщений для текущей темы
101     echo "Название сообщения изменено<BR>";
102     echo "<a href='index.php?show=message&numtopic=".$line[0]."'>";
103     echo "Назад к обсуждению темы </a>";
104 }
105
```

```
106     //Удаление сообщения
107 if ($_GET['act']=='del_message')
108 {
109     //Выбираем код темы, чтобы можно было вернуться
110     //в список сообщений для нее
111     $sSQL="SELECT kodoftopic FROM MESSAGE".
112           " WHERE id=".$_GET['nummessage'];
113     $data=mysql_query($sSQL);
114     //Получаем результат - одна запись
115     $line=mysql_fetch_row($data);
116     //Удаляем выбранное сообщение
117     $sSQL="DELETE FROM MESSAGE WHERE id=".$_GET['nummessage'];
118     //Выполняем запрос
119     mysql_query($sSQL)or die(mysql_error());
120    //Выводим надпись и ссылку на список сообщений для текущей темы
121     echo "Тема удалена<BR>";
122     echo "<a href='index.php?show=message&numtopic=".$line[0]."'>";
123     echo "Назад к обсуждению темы </a>";
124 }
125 ?>
```

Весь модуль поделен на небольшие условные фрагменты, в которых проверяется значение `$_GET['act']`. Давайте рассмотрим схему его работы (рис. 10.55).

Если происходит добавление темы, то `$_GET['act']='add_topic'`, само действие осуществляется с помощью двух SQL-запросов.

- Первый запрос добавляет в таблицу `TOPIC` информацию о новой теме.

```
INSERT INTO TOPIC
SET     kodofrazdel=номер раздела,
        name=название создаваемой темы,
        name_creator=имя автора,
        date_last_answer=дата создания темы;
```

В качестве названия темы берется значение из `$safe_topic`, в качестве имени пользователя, создавшего тему, берется значение из `$_SESSION['name']`, дата создания темы формируется с помощью стандартной функции `date()`. Создаваемая тема привязывается к разделу, идентификационный номер которого доступен в `$_GET['numrazdel']`.

- Второй запрос добавляет в таблицу `MESSAGE` первое сообщение для только что созданной темы.

```
INSERT INTO MESSAGE
SET         kodoftopic=номер созданной темы,
```

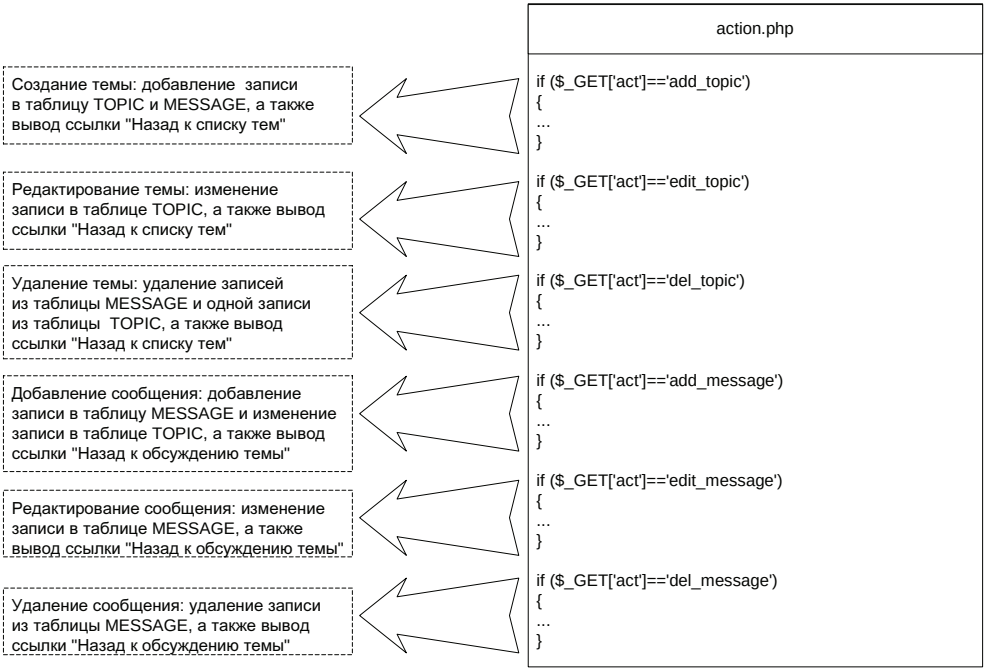


Рис. 10.55.
Упрощенная схема работы скрипта action.php

10.7.6. Модуль действий action.php

Создайте в forum.ru/www файл action.php, его содержимое представлено в листинге 10.13.

```
                text_message='текст
сообщения',
                name_man='имя автора',
               date_answer='дата создания
сообщения';
```

Для привязки сообщения к теме используется его идентификационный номер, который запрашивается у MySQL с помощью специальной функции

mysql_insert_id() т. к. он заранее не известен, потому что тема была недавно создана. Синтаксис функции:

```
mysql_insert_id()
```

Данная функция возвращает номер, сгенерированный MySQL при последнем INSERT-запросе, для столбца, у которого установлена автоматическая нумерация. Когда происходит изменение названия темы, то $_GET['act']='edit_topic', само действие осуществляется с помощью одного SQL-запроса:

```
UPDATE TOPIC
SET name='название темы'
WHERE id=номер изменяемой темы;
```

Когда происходит удаление темы, то $_GET['act']='del_topic', само действие осуществляется с помощью двух SQL-запросов.

- Первый запрос удаляет все сообщения из таблицы MESSAGE, которые принадлежат удаляемой теме.

```
DELETE FROM MESSAGE
WHERE kodoftopic=код темы, все сообщения которой будут удалены;
```

- Второй запрос удаляет выбранную пользователем тему.

```
DELETE FROM TOPIC
WHERE id=номер удаляемой темы;
```

Если происходит добавление сообщения, то $_GET['act']='add_message', само действие осуществляется с помощью двух SQL-запросов.

- Первый запрос добавляет в таблицу MESSAGE новое сообщение.

```
INSERT INTO MESSAGE
SET        kodoftopic= код темы, сообщение для которого будет
                                                    добавлено,
           text_message='текст добавляемого сообщения',
            name_man='имя автора',
            date_answer='дата добавления сообщения'
```

В качестве текста сообщения берется значение из $safe_message, в качестве имени пользователя, создавшего сообщение, берется значение из $_SESSION['name'], дата создания сообщения формируется с помощью стандартной функции date(). Создаваемое сообщение привязывается к теме, идентификационный номер которой доступен в $_GET['numtopic'].

- Второй запрос изменяет в таблице TOPIC имя последнего ответившего автора и дату последнего ответа для темы, сообщение в которую было добавлено первым запросом.

```
UPDATE TOPIC
SET    name_last_answer='имя автора',
       date_last_answer='дата добавления сообщения'
WHERE id=номер темы, для которой было добавлено сообщение
```

Когда происходит изменение сообщения, то `$_GET['act']='edit_message'`, само действие осуществляется с помощью одного SQL-запроса:

```
UPDATE MESSAGE
SET text_message='текст сообщения'
WHERE id=номер изменяемого сообщения
```

Когда происходит удаление сообщения, то `$_GET['act']='del_message'`, само действие осуществляется с помощью одного SQL-запроса:

```
DELETE FROM MESSAGE
WHERE id=номер удаляемого сообщения
```

10.8. В заключение главы

Денвер хранит все базы, созданные в MySQL, по следующему пути: usr\local\mysql4\data. Например, если вы сделаете копию папки FORUM, содержащейся там, и переименуете ее в TEST_BASA, то в MySQL у вас будет доступна новая база данных с именем `TEST_BASA`, содержимое которой полностью аналогично базе данных `FORUM`.

Глава 11
FCKEditor

FCKEditor — это текстовый редактор, специально предназначенный для браузера. Позволяет создать документ и затем получить его в HTML-формате с корректно расставленными HTML-тегами. FCKEditor абсолютно бесплатен и распространяется по лицензии Open Source, к тому же он обладает многоязыковой поддержкой и совместим с различными видами браузеров.

Создайте виртуальный хост **editor.ru**, в нем папку www, далее перезапустите Денвер — только после этого новый хост станет доступен. Все файлы, разработанные в данном разделе, необходимо сохранять в editor.ru\www.

11.1. Установка

Для начала необходимо скачать дистрибутив. Это можно сделать либо с официального сайта:

www.fckeditor.net

либо по следующей ссылке:

http://sourceforge.net/project/showfiles.php?group_id=75348

Далее распакуйте дистрибутив под именем fckeditor в editor.ru/www. В распакованном состоянии FCKEditor занимает приблизительно 2,3 Мбайт. Вы можете удалить часть файлов, которые никак не влияют на функциональность редактора, например, смело можно удалять папки: _samples, _testcases, editor_source (только пока не торопитесь это делать). После чего удастся сэкономить около 370 Кбайт. Более подробно об этом можно прочитать в официальной документации, которая доступна по адресу: **http://fckeditor.wikiwikiweb.de/**.

11.2. Первое знакомство

Для начала обратимся к примерам, которые находятся в папке fckeditor_samples\php\ (рис. 11.1):

- sample01.php;
- sample02.php;
- sample03.php;
- sample04.php.

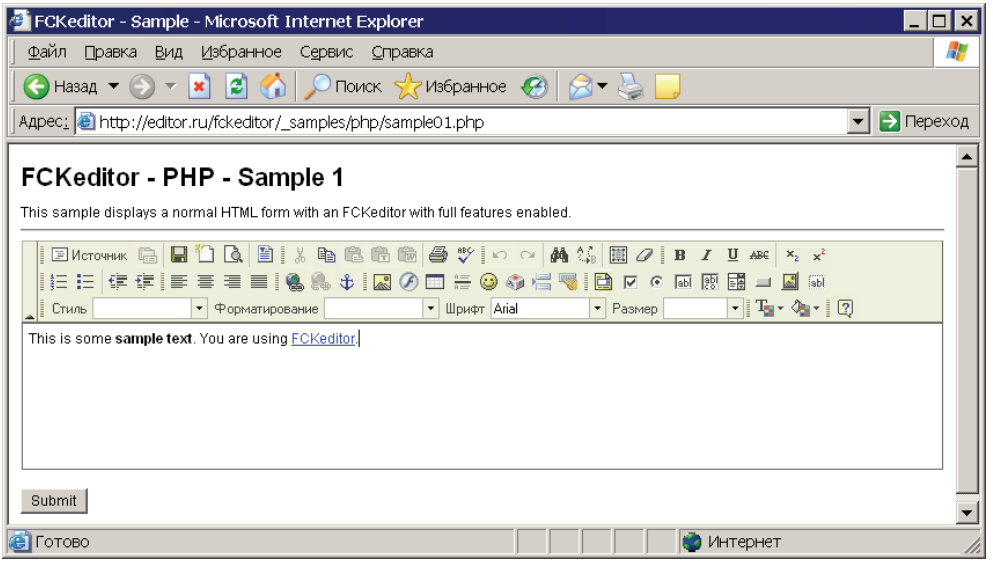


Рис. 11.1. Файл sample01.php, запущенный в браузере Internet Explorer

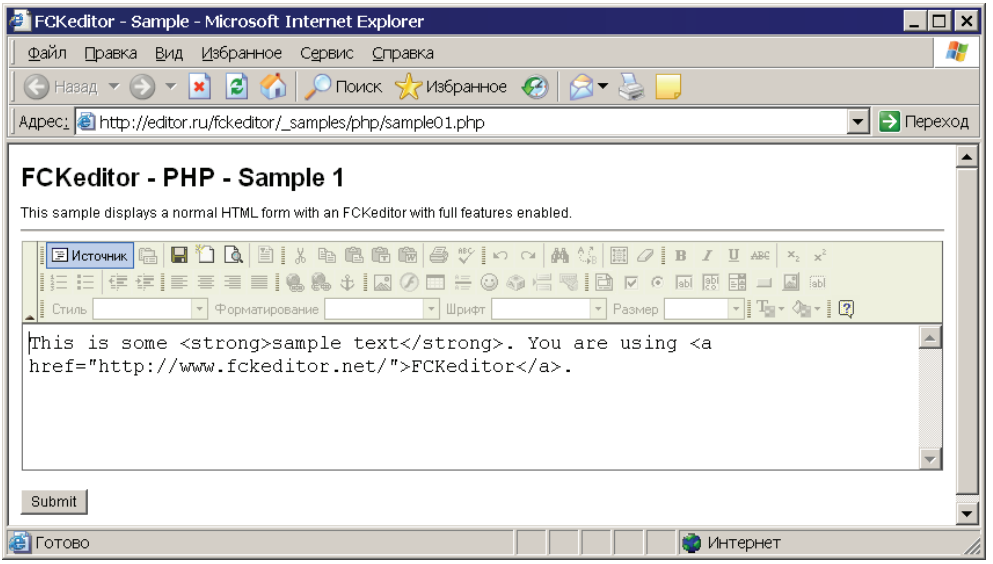


Рис. 11.2. Превращение обычного текста в HTML-документ

При нажатии кнопки **Источник**, расположенной в верхнем левом углу, вы увидите тот же самый текст, который располагался в редакторе, только теперь он будет оформлен HTML-тегами (рис. 11.2) .

Поэкспериментируйте самостоятельно с остальными примерами, поставляемыми с FCKEditor, чтобы получить общее впечатление о возможностях этого редактора.

11.3. Простой пример

Первое, что необходимо сделать, — это подключить модуль fckeditor.php, который находится в папке fckeditor:

```
require_once("fckeditor/fckeditor.php");
```

Далее с помощью следующего кода создается окно редактора FCKEditor:

```
$oFCKeditor = new fckeditor('fckeditor1');
$oFCKeditor->BasePath = '/fckeditor/';
$oFCKeditor->Value = 'Текст';
$oFCKeditor->Create() ;
```

Теперь, если вы создадите скрипт с именем mysample01.php следующего содержания:

```
<?php
require_once("fckeditor/fckeditor.php");

$oFCKeditor = new fckeditor('fckeditor1') ;
$oFCKeditor->BasePath = '/fckeditor/';
$oFCKeditor->Value = 'Это текст, который вы увидите в FCKEditor';
$oFCKeditor->Create() ;
?>
```

то после его запуска увидите следующее — рис. 11.3.

Для того чтобы задать размеры редактора, необходимо использовать следующий код (размеры можно задавать либо в процентах, либо в обычных единицах):

```
$oFCKeditor->Width  = '80%';
$oFCKeditor->Height = '300';
```

Следующий пример — mysample02.php создаст редактор FCKEditor с заданными размерами (рис. 11.4):

```
<?php
require_once("fckeditor/fckeditor.php");
```

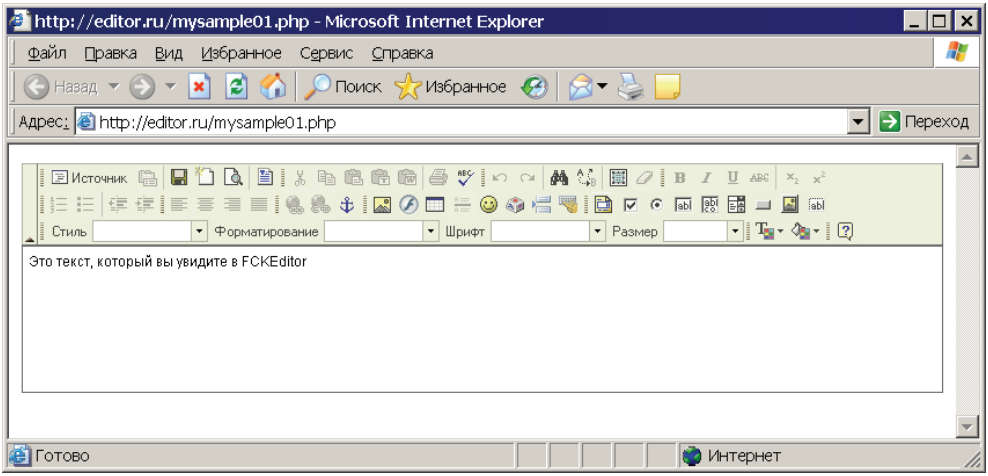


Рис. 11.3. Результат работы программы mysample01.php

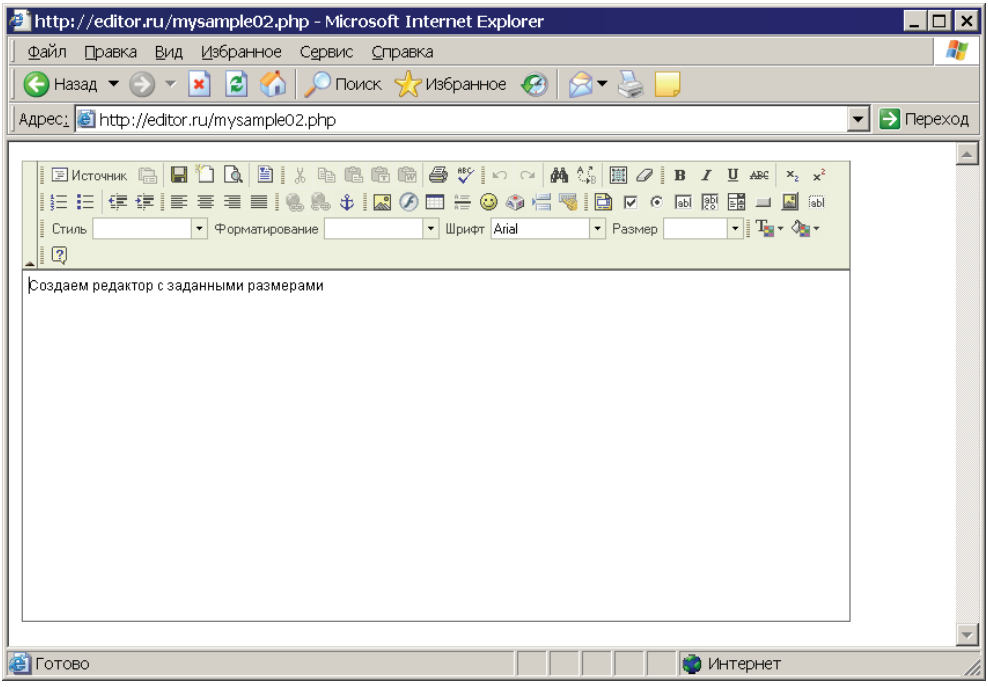


Рис. 11.4. Создание редактора FCKEditor с заданными размерами

```
$oFCKeditor = new fckeditor('fckeditor1') ;
$oFCKeditor->BasePath = '/fckeditor/';
$oFCKeditor->Value = 'Создаем редактор с заданными размерами';
//Длину задаем в процентах
$oFCKeditor->Width  = '90%' ;
//Ширину в обычных единицах
$oFCKeditor->Height = '400' ;
$oFCKeditor->Create() ;
?>
```

11.4. Настраиваем панели инструментов

По умолчанию в FCKEditor доступны два набора панелей инструментов:

- **Default** — включает все панели инструментов, и отображается всегда, когда при вызове редактора не указано ничего другого;
- **Basic** — включает самый минимальный набор.

Для того чтобы задать имя набора, с которым должен быть отображен редактор FCKEditor, необходимо использовать следующий код:

```
$oFCKeditor->ToolbarSet='Название панели';
```

Скрипт mysample03.php создаст редактор с набором панелей **Basic** (рис. 10.5):

```
<?php
require_once("fckeditor/fckeditor.php");

$oFCKeditor = new fckeditor('fckeditor1') ;
$oFCKeditor->BasePath = '/fckeditor/';
$oFCKeditor->Value = 'Создаем редактор с заданным набором панелей';
//Задаем имя набора панелей инструментов
$oFCKeditor->ToolbarSet='Basic';

$oFCKeditor->Create() ;
?>
```

Конечно же, в реальной ситуации понадобится гораздо больший простор по настройке как панелей инструментов, так и отдельных ее элементов. Поэтому откройте файл fckconfig.js. Далее можно пойти двумя путями: либо перенастроить набор под названием **Default**, либо создать собственный набор панелей.

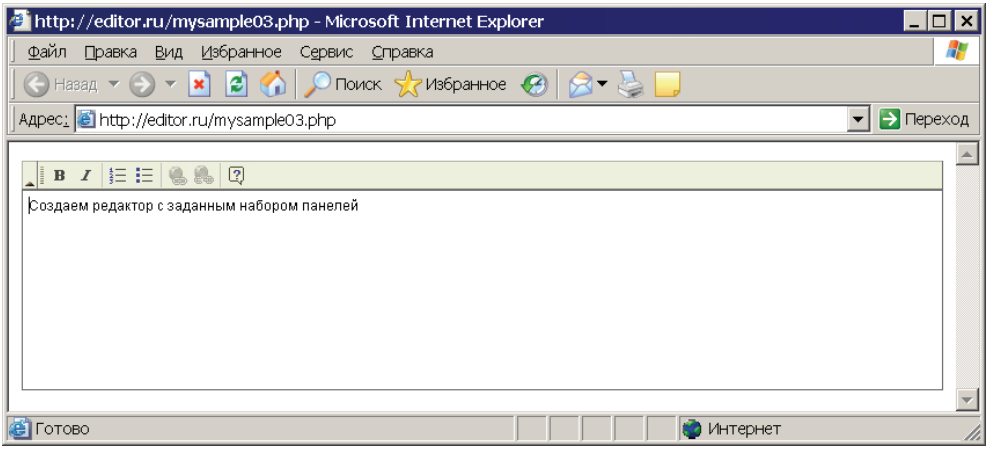


Рис. 11.5. Создание редактора FCKEditor с заданным набором панелей инструментов

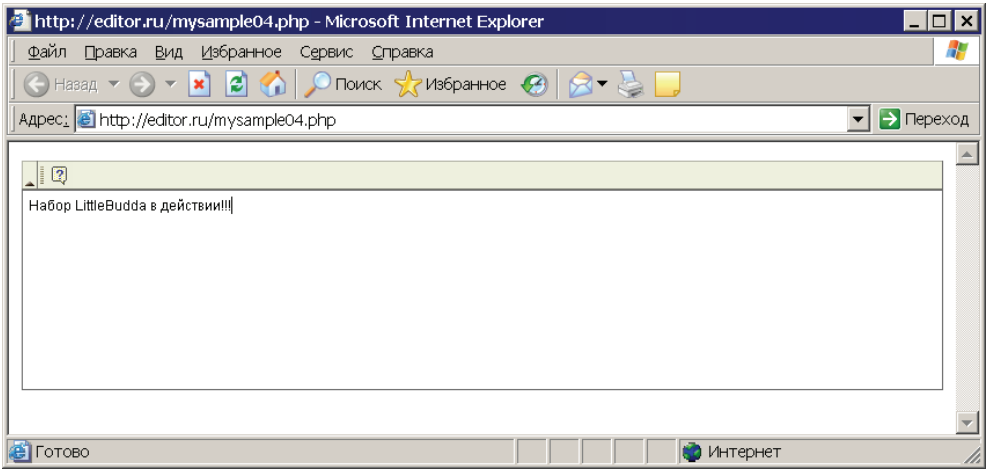


Рис. 11.6. Использование собственного набора панелей инструментов в редакторе FCKEditor

Для того чтобы перенастроить набор **Default**, достаточно найти в файле fckconfig.js строчку `FCKConfig.ToolbarSets["Default"] =` , после нее будет идти указание тех панелей, которые включает набор **Default** (а он включает абсолютно все панели). Вам останется только удалить лишние элементы и сохранить файл настроек.

Для того чтобы создать свой набор, достаточно после строчек создания наборов панелей **Default** и **Basic** описать собственный набор по аналогии с уже существующими. Например, создание набора под названием **LittleBudda**, который включает всего лишь одну кнопку **About**, будет выглядеть следующим образом (рис. 10.6):

```
FCKConfig.ToolbarSets["LittleBudda"] = [
        ['About']

] ;
```

11.5. Получаем данные из редактора

Для того чтобы получить данные из редактора, его необходимо поместить в форму, а затем использовать в скрипте-обработчике формы следующий код:

```
$sValue = stripslashes($_POST['FCKeditor1'] );
```

Например, если вы хотите вывести на экран набранный в FCKEditor текст, то можно воспользоваться следующим примером (mysample05.php):

```
<?php
include("fckeditor/fckeditor.php") ;
?>
<form action="showdata.php" method="post">

<?php
$oFCKeditor = new FCKeditor('FCKeditor1') ;
$oFCKeditor->BasePath = '/fckeditor/';
$oFCKeditor->Value = 'Нажми кнопку и получишь результат';
$oFCKeditor->Create() ;
?>

<input type="submit" value="Показать результат">
</form>
```

обработчиком которого будет скрипт showdata.php со следующим кодом:

```
<?php
echo "<H1>Данные, полученные из редактора FCKEditor</H1>";
```

```
$sValue = stripslashes( $_POST['FCKeditor1'] ) ;
echo $sValue;
?>
```

Результаты работы описанных скриптов показаны на рис. 11.7 и 11.8.

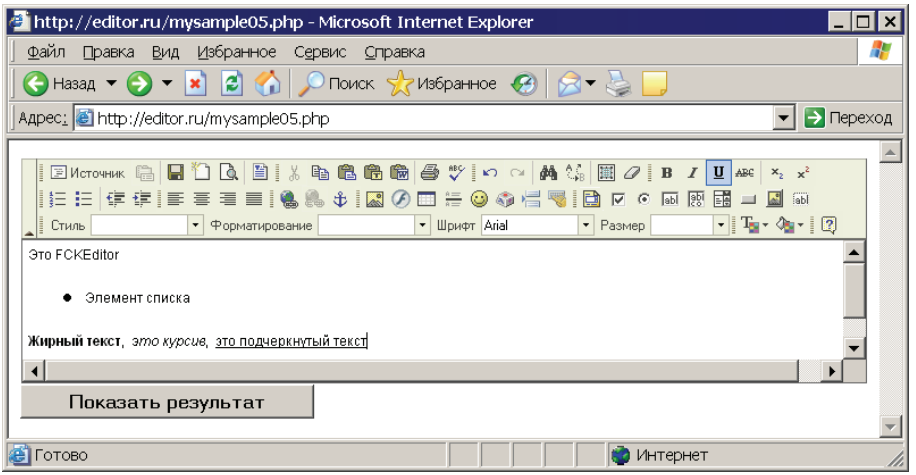


Рис. 11.7. Результат выполнения mysample05.php

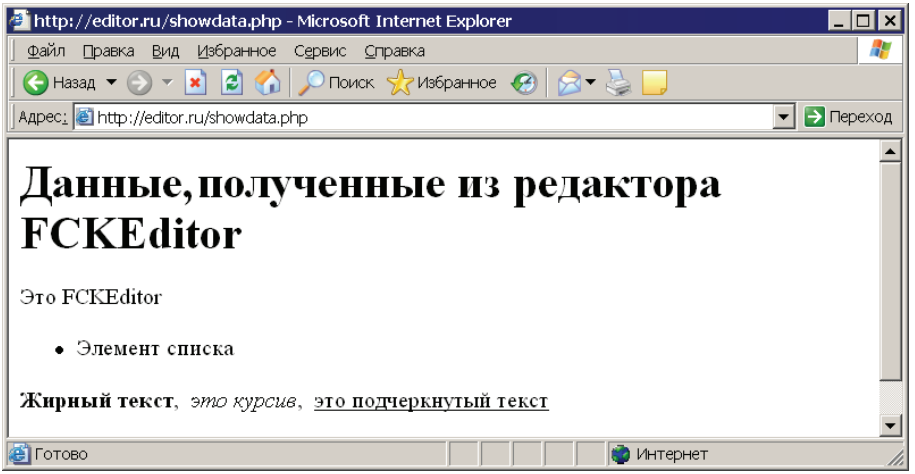


Рис. 11.8. Результат выполнения showdata.php

11.6. Настраиваем инструмент по закачке файлов

Настройка загрузки файлов начинается с создания папки, в которой эти файлы будут храниться. Создайте в корне виртуального хоста (**editor.ru/www**) папку upload_picture (рис. 11.9).

Теперь загляните в "сердце" FCKEditor — файл fckconfig.jc, найдите следующие строки (в версии 2.2 — последней на момент написания книги, это строки **132** и **133**):

```
var _FileBrowserLanguage        = 'asp' ;
var _QuickUploadLanguage        = 'asp' ;
```

и замените их на:

```
var _FileBrowserLanguage        = 'php' ;
var _QuickUploadLanguage        = 'php' ;
```

Этим действием вы обозначили, что загрузка файлов будет осуществляться с помощью PHP-скриптов, поставляемых вместе с FCKEditor, возможны также другие варианты, о которых говорится в комментариях, расположенных рядом с этими строчками (рис. 11.10).

За загрузку файлов отвечают два скрипта, поэтому теперь необходимо выполнить их небольшую настройку, а именно — указать папку, в которую они будут производить загрузку, как вы помните, для этих целей была специально создана upload_picture.

Один файл находится по адресу fckeditor\editor\filemanager\upload\php и именуется config.php, в нем необходимо выполнить следующие действия:

- включить загрузчик, т. е. поменять строку (для версии 2.2 — последней на момент написания книги, это строка **24**):

```
$Config['Enabled'] = false  на  $Config['Enabled'] = true
```

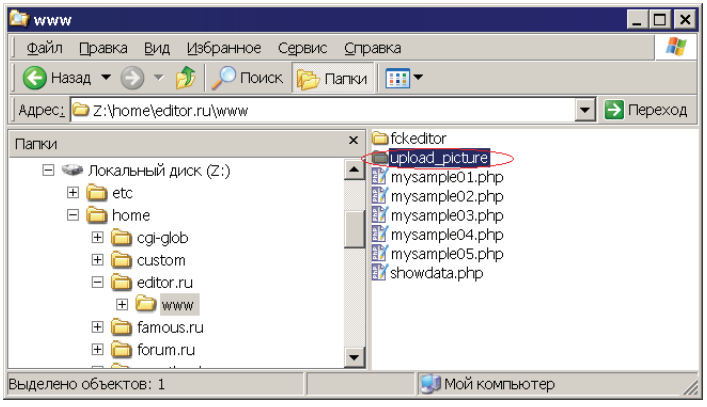


Рис. 11.9. Создание папки upload_picture

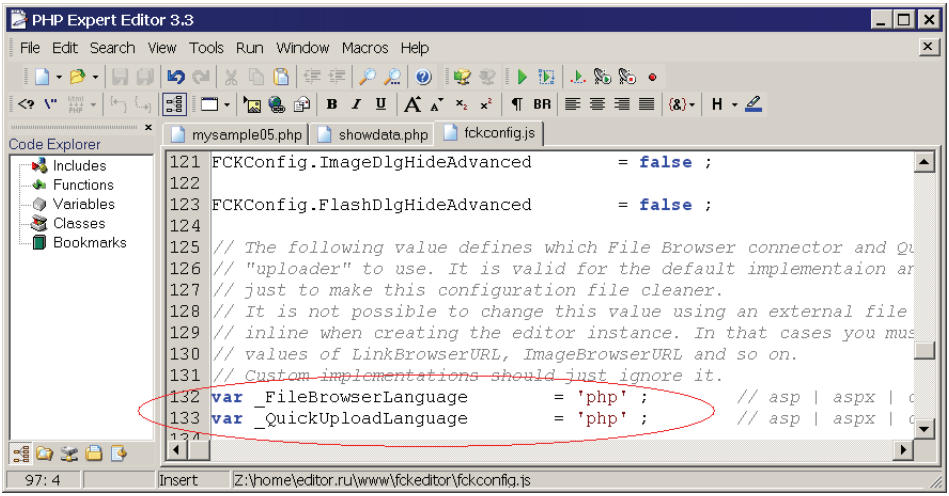


Рис. 11.10. Настройка fckconfig.js

- привести элемент массива `$Config['UserFilesPath']`, в котором по умолчанию установлено `'/UserFiles/'`, к следующему виду (для версии 2.2 — последней на момент написания книги, это строка **27**):

```
$Config['UserFilesPath'] = '/upload_picture/'
```

Не забудьте сохранить файл после внесенных изменений.

Второй файл находится по адресу fckeditor\editor\filemanager\browser\default\connectors\php и имеет также имя config.php, в нем необходимо выполнить следующие действия:

- включить загрузчик, т. е. поменять строку (для версии 2.2 — последней на момент написания книги, это строка **24**):

```
$Config['Enabled'] = false
```

на

```
$Config['Enabled'] = true
```

- привести элемент массива `$Config['UserFilesPath']`, в котором по умолчанию установлено `'/UserFiles/'`, к следующему виду (для версии 2.2 — последней на момент написания книги, это строка **27**):

```
$Config['UserFilesPath'] =
'/upload_picture/'
```

Стоит отметить, что в обоих случаях в config.php также существует массив разрешенных для закачки типов файлов, по умолчанию он выглядит следующим образом:

```
$Config['AllowedExtensions']['Image'] =
array('jpg','gif','jpeg','png') ;
```

Не забудьте сохранить файл после внесенных изменений.

Теперь все готово к тому, чтобы протестировать загрузку картинок на сервер. Запустите FCKEditor, например, для этого подойдет последний разработанный пример с именем mysample05.php, теперь нажмите на панели инструментов кнопку с изображением картинки (рис. 11.11).

После чего вы увидите всплывающее окно **Свойства изображения** (рис. 11.12).

В нем нажмите кнопку **Просмотреть на сервере**, после чего появится еще одно всплывающее окно, в котором с помощью кнопки **Обзор** вам необходимо выбрать файл для закачки, а с помощью кнопки **Upload** вы сможете закачать его на сервер.

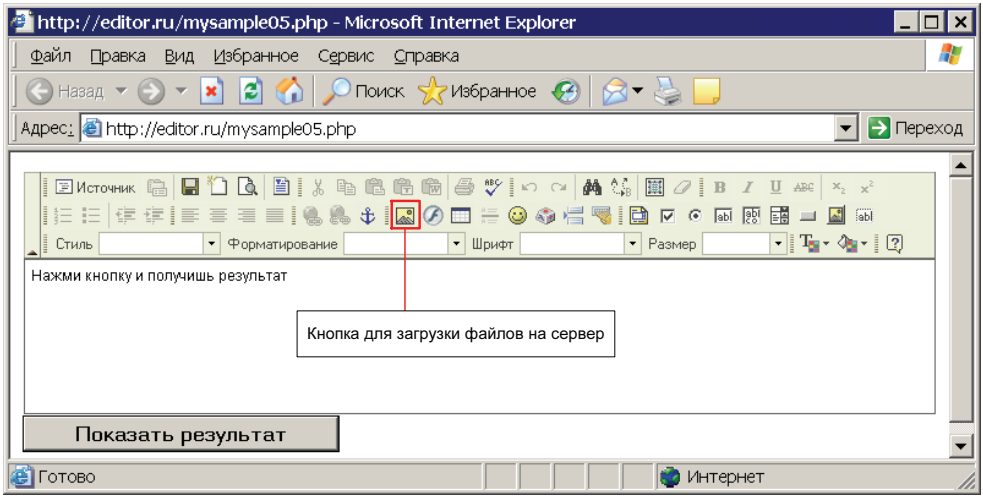


Рис. 11.11. Вызов окна со свойствами изображения из FCKEditor

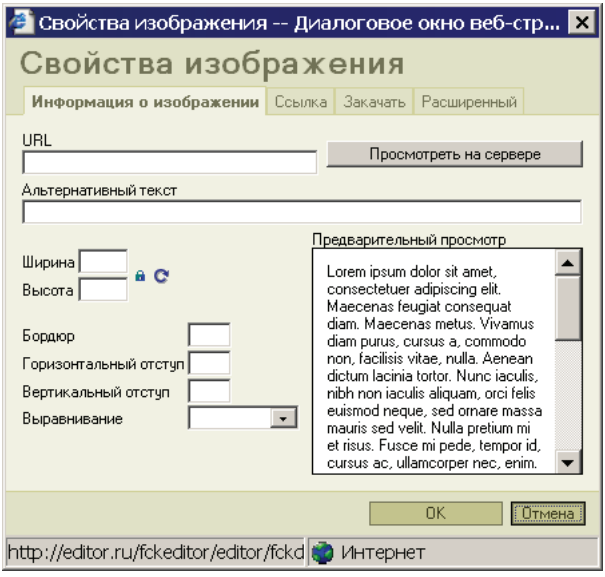


Рис. 11.12. Всплывающее окно **Свойства изображения**

Глава 12
phpBB

Разработать многофункциональный форум с интуитивно понятным интерфейсом — достаточно не простая задача для начинающего PHP-программиста, все дело в том, что для этого нужен определенный опыт. Часто в этом случае используются готовые решения, которые просты в установке и администрировании. Одним из них является очень мощный и в то же время бесплатный форум с открытым кодом под названием phpBB.

Хотя в Интернете можно встретить отрицательные высказывания в его адрес, стоит отметить, что это всего лишь обратная сторона медали популярности какого-либо программного продукта, но не в коем случае не решающий фактор при выборе. Наверняка каждый из вас слышал о найденных уязвимостях в ОС Windows, но все равно после этого мало кто отказался от ее использования.

О популярности phpBB говорит тот факт, что на западе выпущена книга, полностью посвященная ему, информацию о ней можно найти по адресу ***http://www.packtpub.com/phpBB/book****.*

12.1. Установка

Официальным сайтом phpBB является **www.phpbb.com**, там же можно скачать последнюю версию программы и дополнительные элементы к ней.

Российская поддержка форума phpBB осуществляется на сайте **http://myphpbb.com.ru**, где можно как скачать дистрибутив, так и почерпнуть для себя дополнительную информацию — на данном Web-ресурсе есть русскоязычные статьи и форум.

Скачайте последнюю ZIP-версию дистрибутива phpBB (на момент написания книги ею была 2.0.19 размером около 670 Кбайт). Теперь создайте новый виртуальный хост с именем **phpbb.ru** и не забудьте про каталог www. Скопируйте туда содержимое скачанного архива таким образом, чтобы в результате у вас получилось следующее — рис. 12.1.

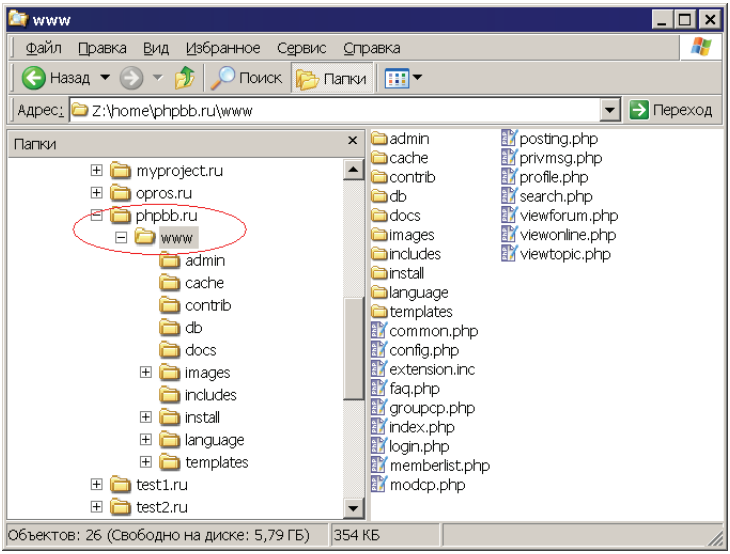


Рис. 12.1. Дистрибутив phpBB в распакованном состоянии

Замечание

В папке docs можно найти документацию на английском языке.

Откройте браузер и введите в нем **phpbb.ru**, вы попадете на мастер установки форума (рис. 12.2).

В выпадающем списке **Database Type** выбирается тип базы данных, в нашем случае это **MySQL 4.x** или **5.x**, хотя можно указать даже обычный **Microsoft Access**.

Далее в поле **Your Database Name** необходимо ввести имя базы данных, но она должна существовать, поэтому запустите phpMyAdmin и создайте базу `phpbb_forum`, имя которой затем и укажите в вышеописанном поле.

В поле **Database Username** введите пользователя базы данных, конечно же, это будет `root`, при условии, что вы используете Денвер.

В поле **Database Password** вводится пароль пользователя, оставьте его пустым, опять же при условии, что вы используете Денвер.

В поле **Admin Email Address** вводится электронный адрес администратора.

В поле **Administrator Username** введите логин администратора форума (например, `admin`), в поле **Administrator Password** — пароль администратора (например, 1), а в поле **Administrator Password [Confirm]** подтвердите введенный пароль администратора.

На рис. 12.3 вы можете видеть мастер установки phpBB с введенными данными.

Нажмите кнопку **Start Install**, после этого будет произведена установка, и по ее окончании вы увидите следующее — рис. 12.4.

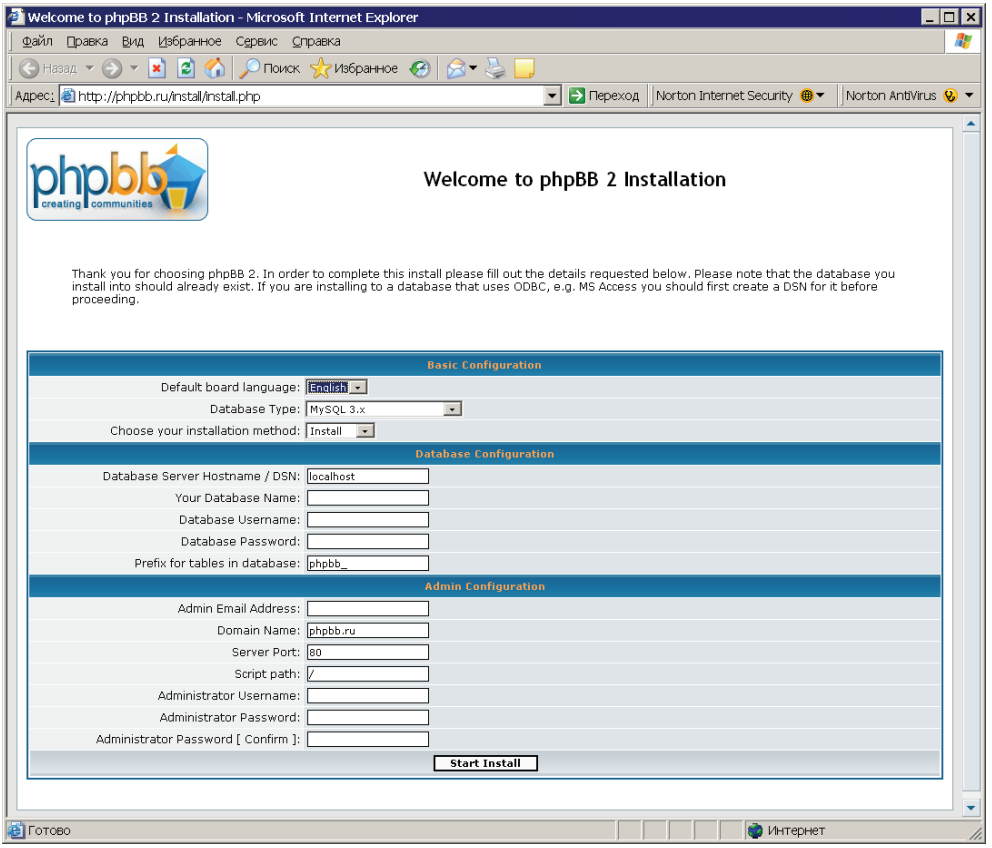


Рис. 12.2. Мастер установки phpBB

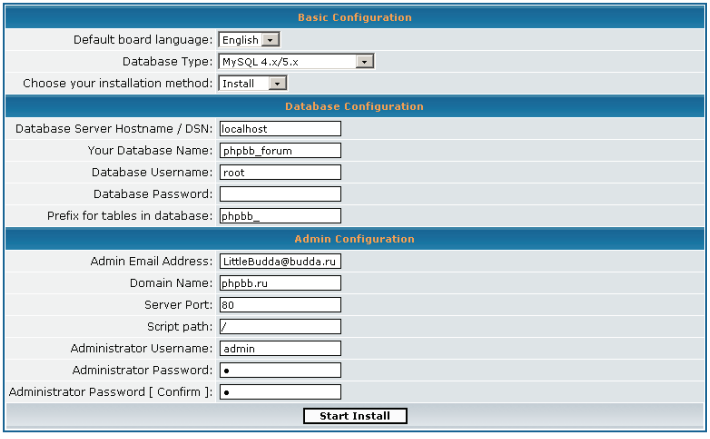


Рис. 12.3. Мастер установки с введенными данными

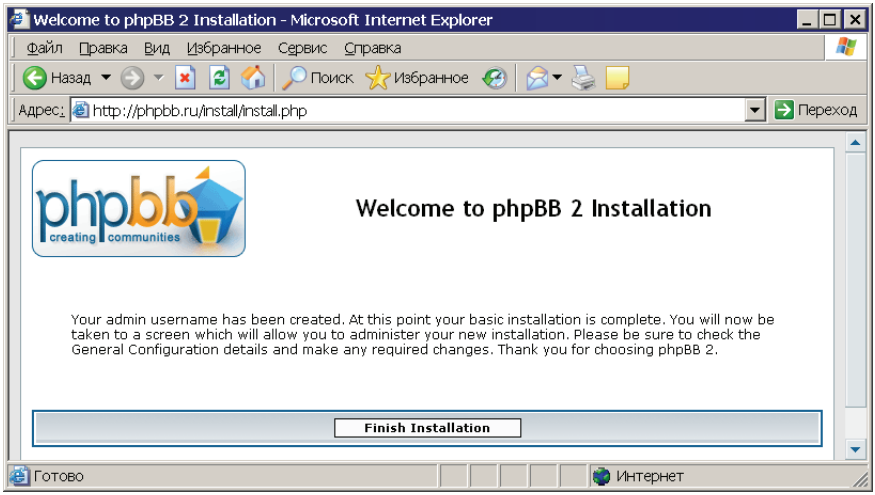


Рис. 12.4. Окончание установки форума phpBB

Нажмите кнопку **Finish Installation** и удалите папки install и contrib, т. к. они после установки больше не понадобятся и будут представлять собой угрозу безопасности для вашего форума.

12.2. Работаем с форумом

По адресу **phpbb.ru** вам доступен только что установленный форум, правда информации на нем пока будет достаточно мало (рис. 12.5).

Первое, чем мы займемся, — это установкой поддержки русского языка. Для этого необходимы два файла: lang_russian_tu.zip и subsilver_russian_tu.zip, в последнем хранятся русскоязычные пиктограммы для кнопок форума. Зайдите на **phpbb.com**, выберите сверху раздел **Downloads**, затем слева пункт **Language & Image Packs**, теперь найдите упомянутые ранее файлы для русского языка (рис. 12.6) и скачайте их.

Далее распакуйте lang_russian_tu.zip в каталог languages вашего форума, а из содержимого архива subsilver_russian_tu.zip скопируйте подкаталог lang_russian_tu по следующему пути: phpbb.ru/www/templates/subSilver/images.

Теперь зайдите в админку phpBB (она доступна по адресу **phpbb.ru/admin**), предварительно необходимо будет пройти авторизацию, во время которой введите имя и пароль администратора, указанные вами при установке. Выберите в разделе **General Admin** пункт **Gonfiguration** и в выпадающем списке **Default Language** установите пункт **Russian [Tu]** (рис. 12.7).

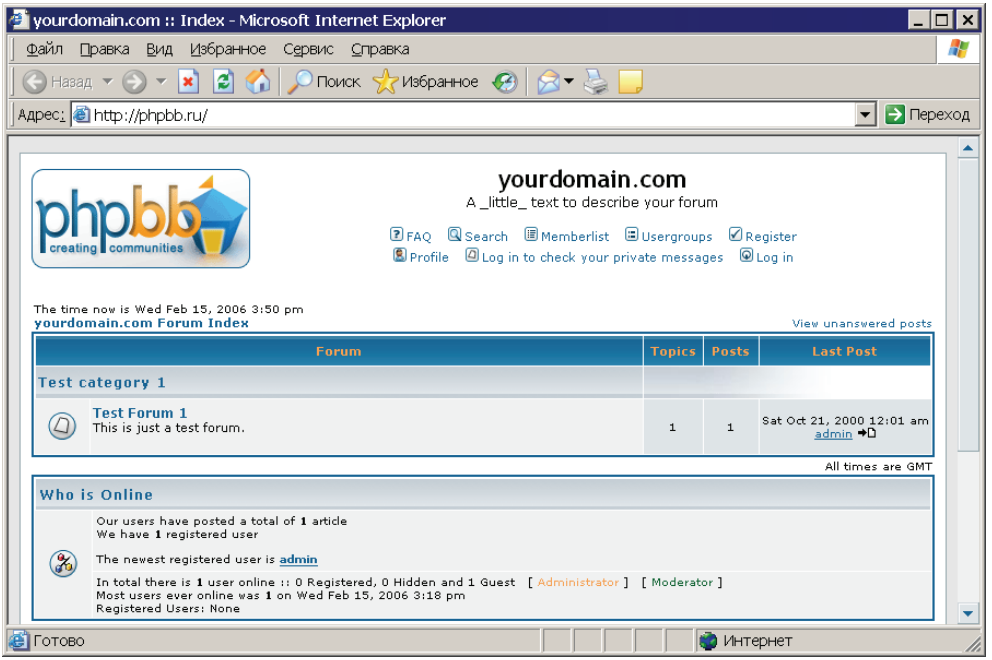


Рис. 12.5. Вид форума сразу после установки

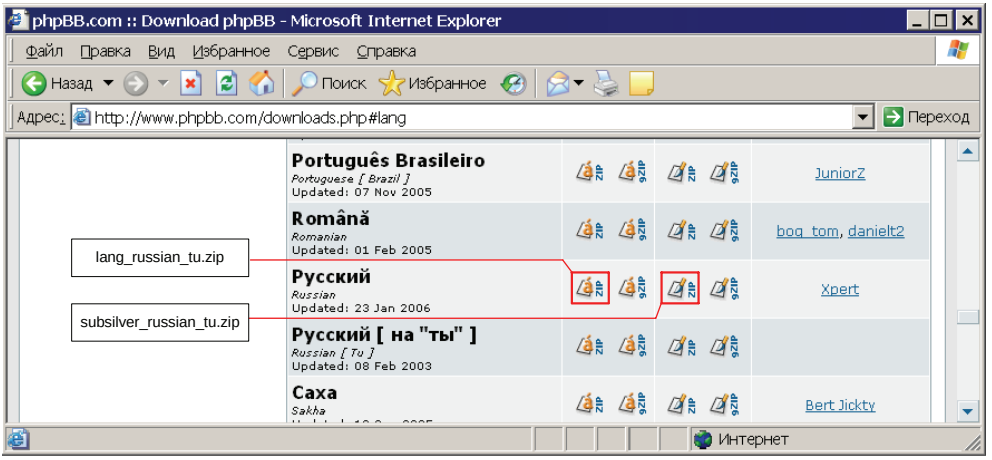


Рис. 12.6. Файлы, необходимые для поддержки форумом phpBB русского языка

Рис. 12.7. Выбор языка, с которым будет работать форум phpBB

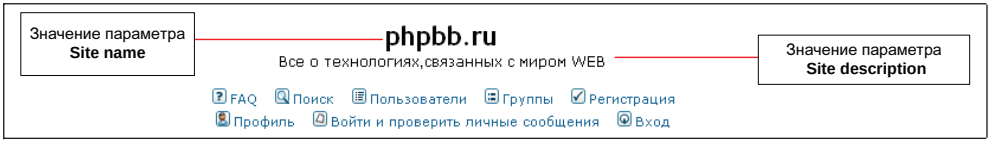


Рис. 12.8. Результат настройки параметров: **Site name** и **Site description**

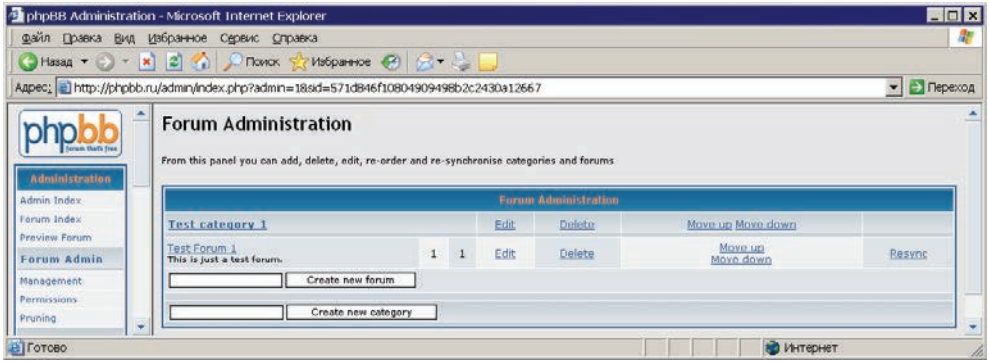


Рис. 12.9. Менеджер форума

Теперь прокрутите вертикальным скроллингом до конца страницы и нажмите кнопку **Submit**. После чего запустите форум и убедитесь, что он стал русскоязычным.

Выберите еще раз пункт **Gonfiguration** раздела **General Admin**. В поле **Site name** введите имя вашего сайта, а в поле **Site description** — его описание, сохраните изменения, после чего вы сможете увидеть только что введенную информацию на своем форуме (рис. 12.8).

Выберите пункт **Management** раздела **Forum Admin**, вы увидите следующее — рис. 12.9.

Здесь вы можете добавлять новые категории и разделы форума с помощью кнопок **Create new category** и **Create new forum**, а также редактировать уже имеющиеся с помощью соответствующих ссылок.

С помощью пункта **Permissions** раздела **Forum Admin** вы можете настраивать права доступа, а с помощью пункта **Word Censors** раздела **General Admin** — создать список слов, подлежащих цензуре.

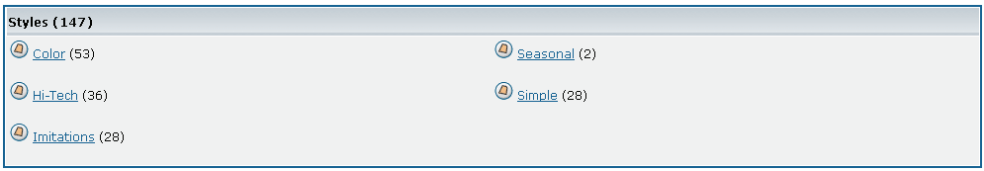


Рис. 12.10. Стили для phpBB, разбитые по группам

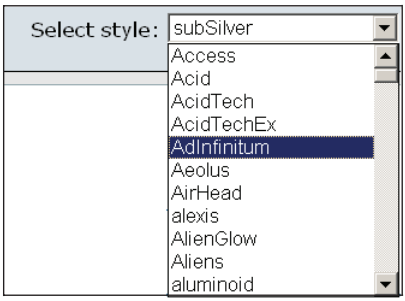


Рис. 12.11. Предварительный просмотр стиля для форума phpBB

Для того чтобы сменить дизайн форума, необходимо скачать и затем установить дополнительные стили или скины. Зайдите на **www.phpbb.com**, выберите раздел **Styles**, далее ссылку **Downloads** и скачайте понравившийся вам стиль из группы элементов **Styles** (рис. 12.10).

Но прежде чем тратить трафик на закачку, вы можете сначала посмотреть, как будет выглядеть форум phpBB после применения к нему выбранного вами стиля. Для этого на **www.phpbb.com** выберите раздел **Styles**, а затем ссылку **Styles demo**, в результате вам станет доступен выпадающий список **Select style** (рис. 12.11), выбрав в котором пункт с именем стиля вы сразу сможете увидеть результат.

Предположим, вы скачали стиль, далее вам необходимо распаковать архив в папку templates, затем зайти в панель администратора и выбрать в разделе **Styles Admin** пункт **Management**. Перейдите по ссылке **Edit** (рис. 12.12) и в выпадающем списке **Template** выберите пункт, имя которого будет совпадать с именем скачанного вами стиля. На рис. 12.13 показан вид форума после установки для него стиля **AdInfinitum** из группы **Simple**.

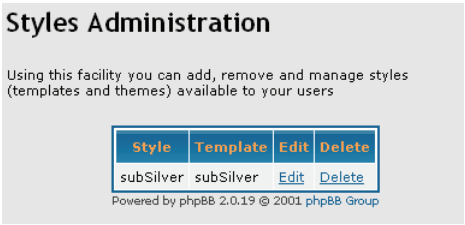


Рис. 12.12. Менеджер стилей

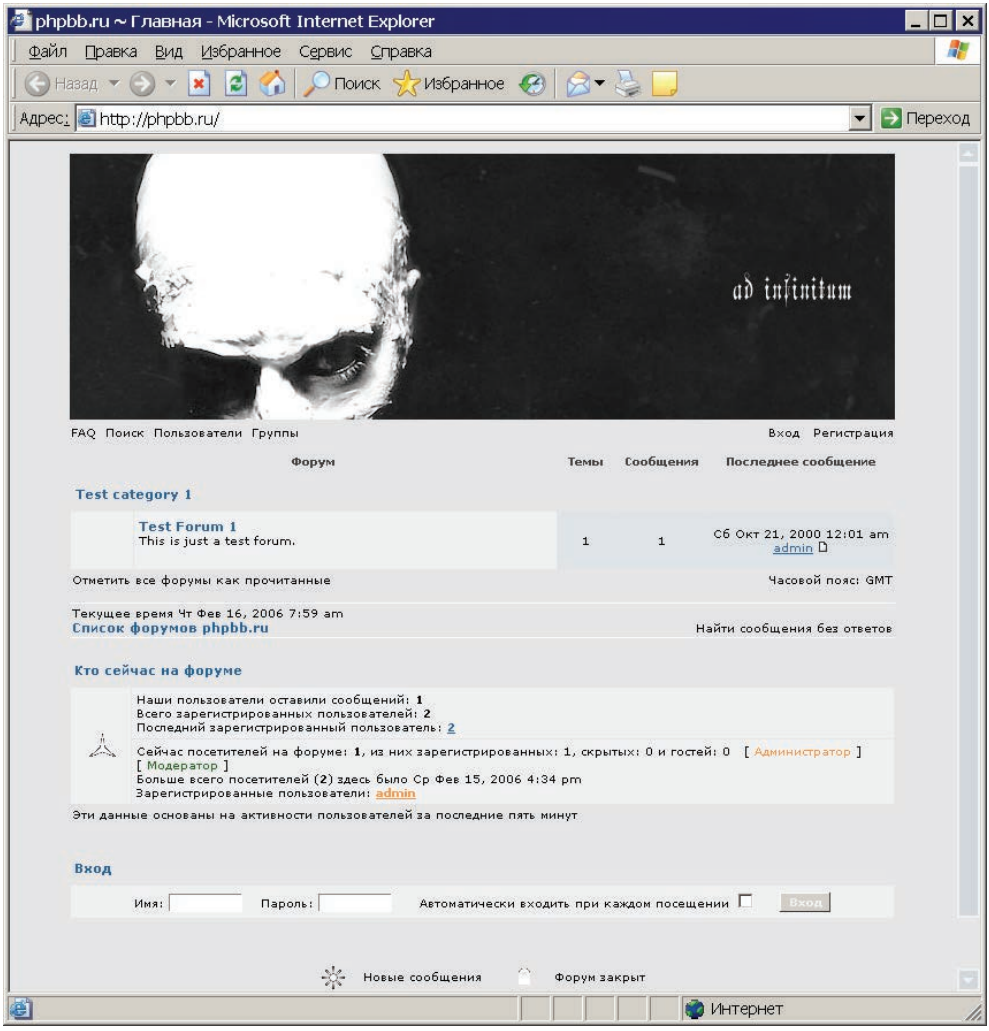


Рис. 12.13. Вид форума phpBB с примененным к нему стилем **AdInfinitum**

Глава 13
Знакомство с Mambo

Не обязательно глубоко вникать в тонкости PHP, быть профессиональным программистом и разработчиком, для того чтобы создавать свои сайты, потому что это можно сделать, не написав ни единой строчки кода. Как это реализовать на практике? Ответ прост — используйте CMS.

CMS (Content Management System) — это система управления контентом (от англ. *content* — содержание) сайта. Она предоставляет удобный механизм наполнения вашей Web-странички информацией, при этом всю техническую сторону берет на себя. То есть вам не требуется даже ничего программировать, вы просто занимаетесь наполнением сайта информацией.

В настоящее время существует достаточно большое количество готовых CMS, причем все они различаются как по функциональным возможностям, так и по цене. В данной главе мы рассмотрим CMS под названием Mambo, которая является одной из самых популярных в мире. Она совершенно бесплатна и переведена на русский язык вместе с документацией, что делает ее еще более привлекательной. Mambo одинакова удобна как для разработки обыкновенной домашней странички, так и для проектов средней сложности.

Функциональные возможности Mambo очень легко расширить, всего лишь установив один или несколько дополнительных компонентов для нее, которые очень легко найти в Интернете. Таким образом, вы можете добавить к своему сайту интернет-магазин, форум и т. д. — и это всего за несколько минут, которые придется потратить на установку и настройку. То же самое касается и внешнего вида вашего сайта — вы можете установить один из готовых шаблонов либо разработать свой собственный, тем самым придав вашему Web-творению уникальность и неповторимость.

О популярности Mambo говорит тот факт, что она переведена более чем на 40 языков.

Предлагаю вам зайти на сайт **www.ru-mambo.ru**, который является официальным сайтом русского сообщества Mambo. Далее зайдите в **Форум** и выберите там раздел **Зацените!** — тут вы можете ознакомиться с готовыми сайтами, которые были

разработаны с помощью Mambo. Советую вам немного попутешествовать здесь, чтобы у вас сложилось общее впечатление о Mambo, как средстве быстрой разработки сайтов.

13.1. Установка

Для начала необходимо скачать последнюю версию дистрибутива Mambo, он весит около 2 Мбайт. Это можно сделать с помощью следующих сайтов:

- **www.mamboforge.net** — официальный сайт Mambo;
- **www.ru-mambo.ru** — официальный сайт русского сообщества Mambo.

Замечание

На обоих сайтах вы можете скачать русскую версию Mambo, а также документацию, переведенную на русский язык.

На русскоязычном сайте также доступны модифицированные версии Mambo, которые уже включают в себя ряд дополнительных, наиболее востребованных компонентов, отсутствующих в базовом варианте. Подробнее об этом можно прочитать, нажав на ссылку **Читать описание**, расположенную напротив каждого варианта дистрибутива (рис. 13.1).

В момент создания этой главы последней версией Mambo была 4.5.2.3 и файл дистрибутива на **www.ru-mambo.ru** назывался **Mambo_RUS_4523_Stable.zip**,

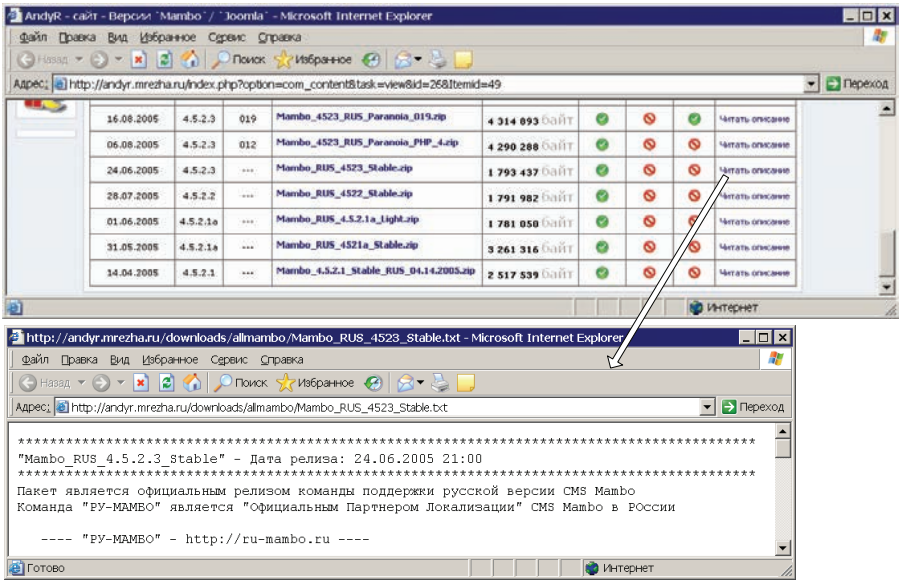


Рис. 13.1. Различные версии Mambo, доступные на **www.ru-mambo.ru**

возможно сейчас доступна более поздняя версия, и если это так, то вам необходимо закачать именно ее, в любом случае вам нужен файл, содержащий в своем имени Stable.

После того как дистрибутив закачан, создайте у себя новый виртуальный хост с именем **mambo.ru** и конечно же папку www в нем. Перезапустите Денвер. Теперь распакуйте скачанный архив дистрибутива Mambo в только что созданный хост, в результате у вас должно получиться следующее — рис. 13.2.

Запустите браузер и введите в нем `mambo.ru`, вы увидите мастер установки Mambo (рис. 13.3).

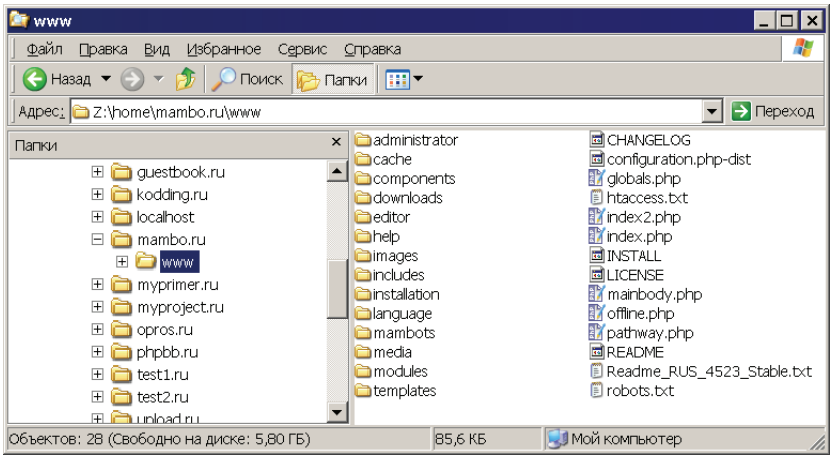


Рис. 13.2. Подготовка к установке Mambo

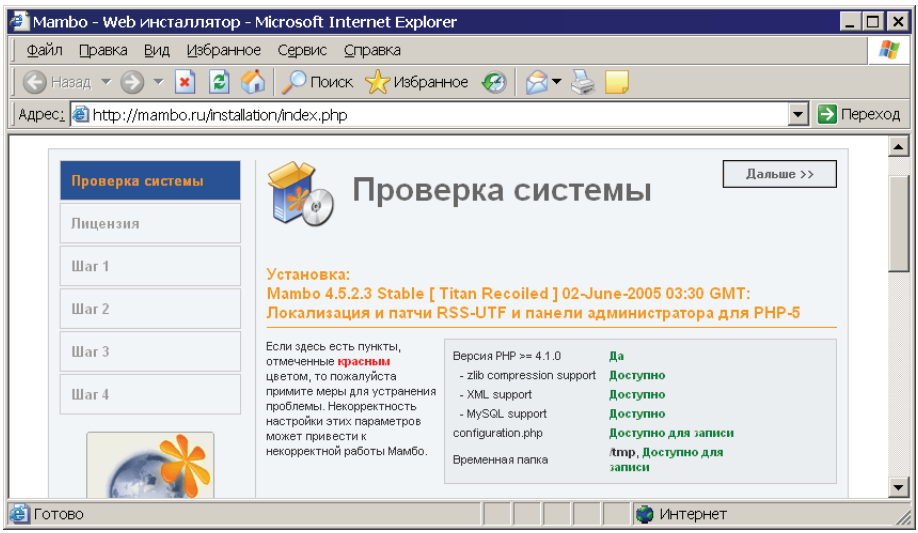


Рис. 13.3. Мастер установки Mambo, этап **Проверка системы**

Рис. 13.4. Настройка Mambo для работы с базой данных MySQL

Нажмите кнопку **Дальше**, после чего появится лицензионное соглашение. Установите флаг **Я принимаю лицензионное соглашение** и нажмите кнопку **Дальше**. Теперь вам необходимо ввести настройки, касающиеся базы данных (рис. 13.4):

- **Host Name** — имя хоста, на котором установлен MySQL, введите `localhost`;
- **MySQL User Name** — имя пользователя, под которым будет работать Mambo с MySQL, введите `root`;
- **MySQL Password** — пароль пользователя, под которым будет работать Mambo с MySQL, оставьте данное поле пустым;
- **MySQL Database Name** — имя базы данных, в которой Mambo будет хранить все свои данные, можете ввести любое, какое вам нравится — главное, чтобы базы с таким именем еще не было.

Нажмите кнопку **Дальше**. Появится окно, где вас спросят, уверены ли вы в правильности введенных данных, нажмите кнопку **OK**. Далее необходимо ввести имя вашего сайта и нажать кнопку **Дальше**, после чего вы перейдете к последнему этапу установки. Здесь особую важность представляет поле **Пароль Админа**, оно будет уже заполнено, но вы можете исправить это значение — пароль вы будете использовать при входе в панель администратора Mambo. Рекомендуется также заполнить поле **Ваш E-mail**. На рис. 13.5 вы можете видеть последний этап мастера установки **Mambo**, с заполненными данными.

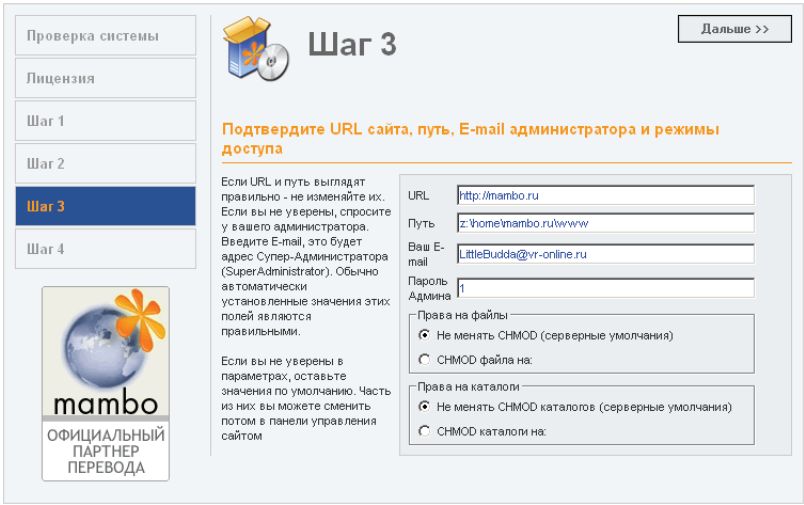


Рис. 13.5. Шаг 3 установки Mambo

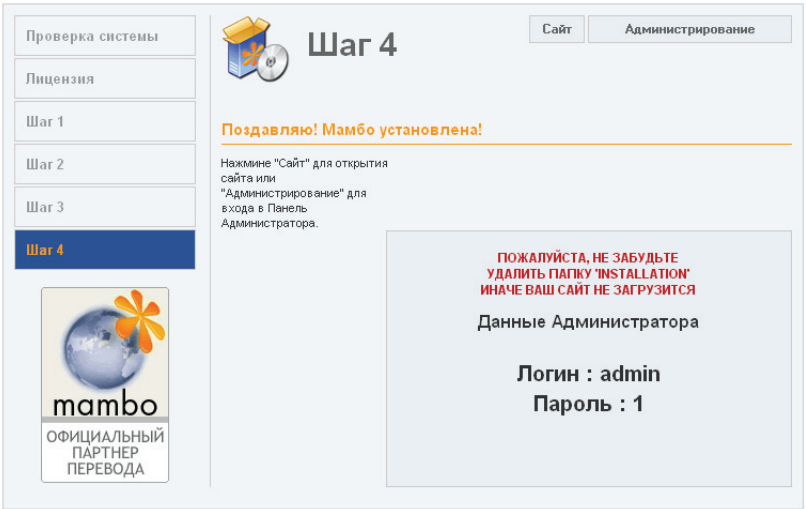


Рис. 13.6. Завершение установки Mambo

Нажмите кнопку **Дальше**. Появится информация о том, что Mambo установлена, а также сведения об учетной записи администратора (рис. 13.6).

Теперь удалите с вашего хоста папку installation. Далее наберите в браузере адрес **mambo.ru/administrator**, после чего вы попадете на форму авторизации, в которой введите логин и пароль администратора (рис. 13.7).

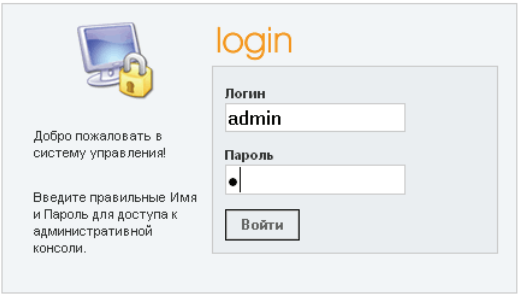


Рис. 13.7. Форма авторизации панели администратора Mambo

Рис. 13.8. Демонстрационный сайт, поставляемый с Mambo

Нажмите кнопку **Войти**, после этого вы попадете в панель администратора Mambo, с помощью которой всего за несколько минут вы создадите свой сайт.

А если вы наберете в браузере адрес **mambo.ru**, то увидите демонстрационный сайт, поставляемый с Mambo (рис. 13.8).

13.2. Основные принципы работы с Mambo

Работа с Mambo очень проста, если вы знаете основные принципы, заложенные в ее основу, которые и будут рассмотрены в данном разделе.

В Mambo вся информация для сайта хранится в сгруппированном виде, разделенная на **Секции** (или **Разделы**), **Категории** и **Объекты**. Предположим, вы разрабатываете сайт для клуба автолюбителей, и у вас есть много материала, посвященного абсолютно различным видам машин. Чтобы упорядочить его в соответствии с правилами Mambo, необходимо будет создать следующую иерархию — рис. 13.9.

В данном случае секция выступает как контейнер для более мелких единиц информации: категорий и объектов. Назначение категорий — придать всей информации дополнительную структурированность, разбив все объекты по общим признакам.

*Когда информацию сложно упорядочить по описанной ранее схеме, то в Mambo она помещается в так называемые **Статичные материалы**. Ярким примером такой информации будет **Свод правил клуба автолюбителей**, т. к. создавать целую иерархию для этой информационной единицы в данном случае не рационально.*

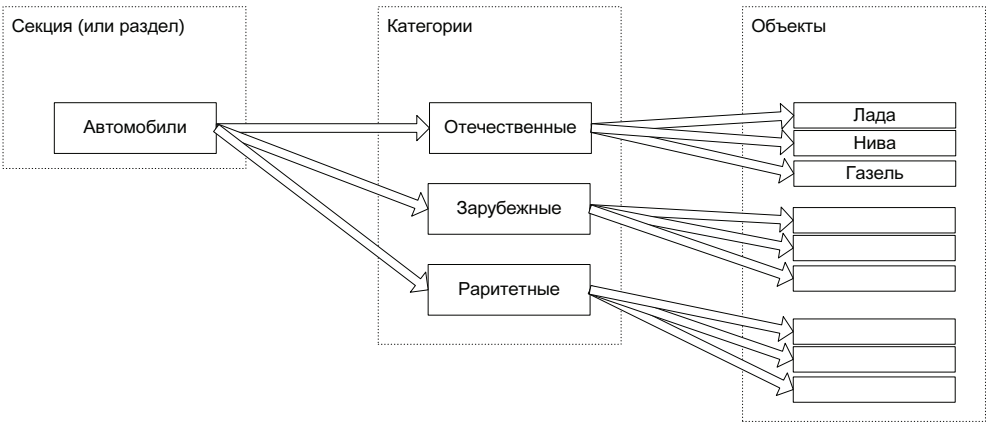


Рис. 13.9. Способ хранения информации для сайта, созданного с помощью Mambo

Сам сайт в Mambo строится с помощью модулей и компонентов. *Модуль* — это небольшой прямоугольный фрагмент вашего сайта, который имеет определенное назначение, например, рис. 13.10.

Компонент — это то, что наделяет ваш сайт возможностью делать те или иные вещи, другими словами можно сказать, что компонент — это мини-программа (рис. 13.11).

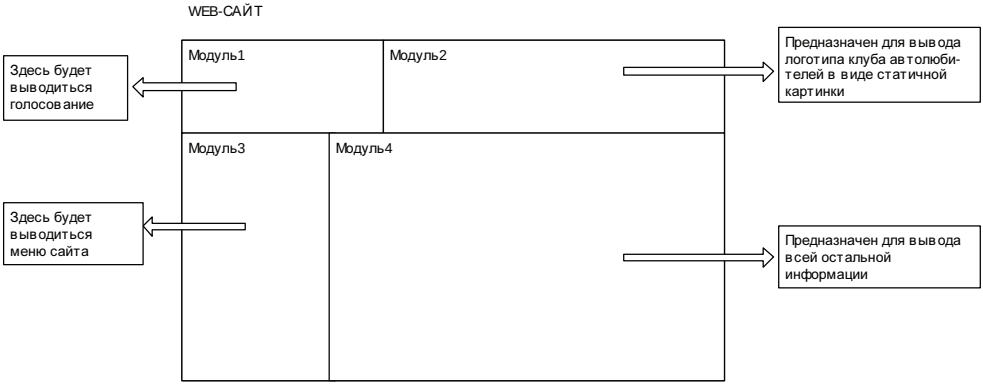


Рис. 13.10. Вариант расположения модулей

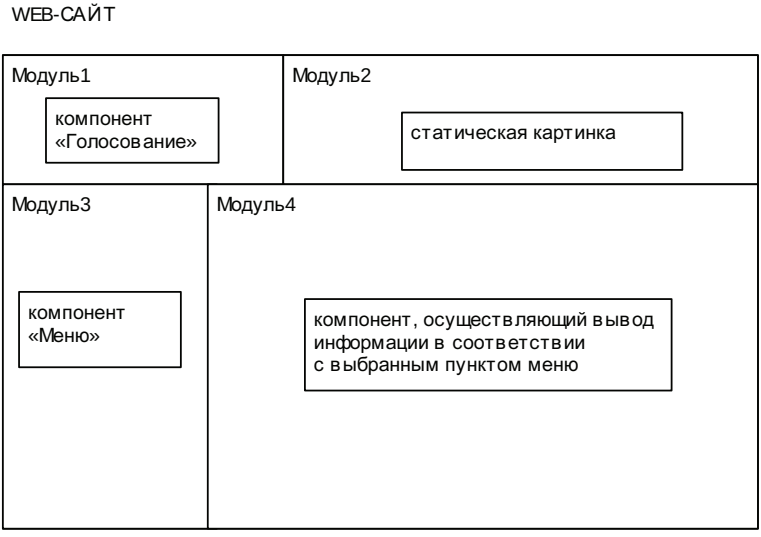


Рис. 13.11. Вариант использования компонентов

Как правило, в стандартный дистрибутив Mambo уже входит набор самых необходимых для создания сайта компонентов. Но ничто не мешает вам установить дополнительные, такие как фотогалерея, интернет-магазин и т. д.

Таким образом, меняя расположение модулей, устанавливая на них различные компоненты, вы можете, как из конструктора, собирать свой сайт, придавая ему те или иные возможности.

За дизайн сайта (его цветовую палитру, фон, шрифты и т. д.) отвечает шаблон, который и определяет конечный вид вашего творения. Изменив шаблон, вы можете придать своему сайту абсолютно новый дизайн, при этом нисколько не меняя содержимое самого сайта.

13.3. Разрабатываем свой сайт

Сейчас мы с вами разработаем сайт клуба автолюбителей. Основной упор будет сделан на практический аспект работы с Mambo, т. к. теоретическая часть очень подробно описана в русскоязычной документации к Mambo.

Итак, вы в панели администратора. Сначала сделаем сайт недоступным для посетителей, т. к. он еще не разработан. Для этого нажмите левой кнопкой мыши на пиктограмме **Конфигурация** (рис. 13.12) или выберите пункт меню **Сайт | Конфигурация**.

Установите переключатель **Сайт в OFF-line** в позицию **Да**. А в поле для ввода многострочного текста **OFF-line сообщение** введите текст, который будет отображен пользователю, обратившемуся к вашему сайту. Нажмите кнопку **Сохранить**. Теперь, если вы наберете в браузере **mambo.ru**, вы увидите следующее — рис. 13.13.

Рис. 13.12. Пиктограмма для доступа к настройкам сайта

Рис. 13.13. Сообщение о том, что сайт закрыт

Обратимся к информационной наполненности сайта. Он будет состоять из нескольких разделов:

- Новости
- Автомобили
- Правила клуба автомобилистов
- О нас
- Форум

Раздел **Автомобили** будет иметь следующую структуру:

- Отечественные:
  - Лада
  - Нива
  - Газель
- Зарубежные
- Раритетные

Выберите пункт меню **Материалы | Разделы** или нажмите левой кнопкой мыши на пиктограмму **Разделы** (рис. 13.14).

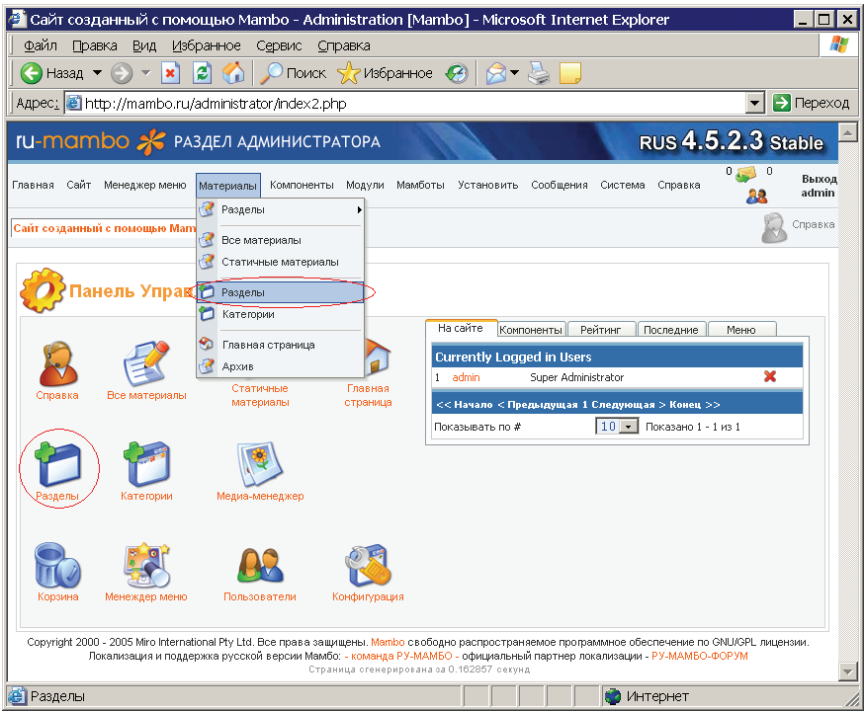


Рис. 13.14. Приступаем к работе с разделами

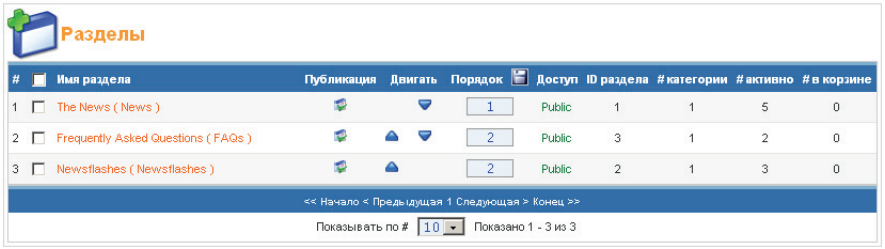


Рис. 13.15. Менеджер разделов

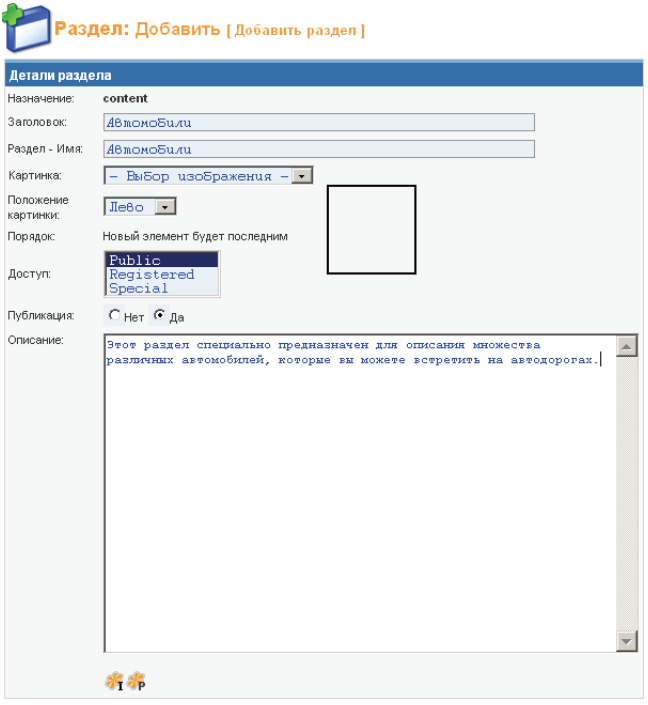


Рис. 13.16. Ввод данных для создаваемого раздела

Вы попадете в менеджер разделов (рис. 13.15).

Как видите, здесь уже присутствует несколько разделов. **The News (News)** мы оставим, а вот два остальных мы сделаем невидимыми. Установите флаг напротив них и нажмите кнопку **Скрыть**. Теперь нажмите кнопку **Добавить**, после чего вы увидите форму создания нового раздела, введите в нее следующую информацию (рис. 13.16):

- в поле **Заголовок** и в поле **Раздел-Имя** введите: `Автомобили`;
- в поле **Описание** введите текст-описание для создаваемого раздела.

Нажмите кнопку **Сохранить**, в результате вы увидите, что в списке разделов появился новый пункт **Автомобили**.

Теперь изменим имя раздела **The News (News)** на русскоязычный вариант — **Новости**. Для этого щелкните левой кнопкой мыши на пункте **The News (News)** и в появившейся форме в поле **Заголовок** и **Раздел-Имя** введите `Новости`, после этого нажмите кнопку **Сохранить**.

Переходим к созданию **Категорий**. Выберите в панели администратора пункт меню **Материалы | Разделы | Автомобили | Добавить/Править Автомобили категории** (рис. 13.17).

Появится менеджер категорий для раздела **Автомобили**, он будет пустой, сейчас мы это исправим. Нажмите кнопку **Добавить**, вы увидите форму точь-в-точь похожую на ту, в которой вы создавали раздел, заполните ее следующим образом (рис. 13.18):

- в поле **Заголовок категории** и в поле **Имя категории** введите: `Отечественные`;
- в поле для ввода многострочного текста **Описание** введите текст-описание для создаваемой категории.

Нажмите кнопку **Сохранить**. После чего создайте точно таким же образом категории **Иностранные** и **Раритетные**.

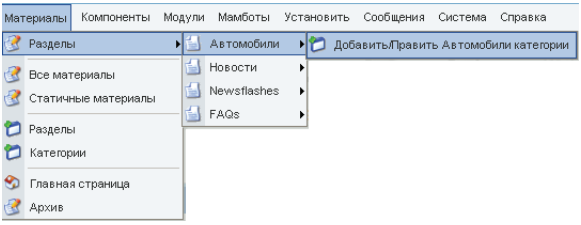


Рис. 13.17. Выбор пункта меню для создания новой категории в разделе **Автомобили**

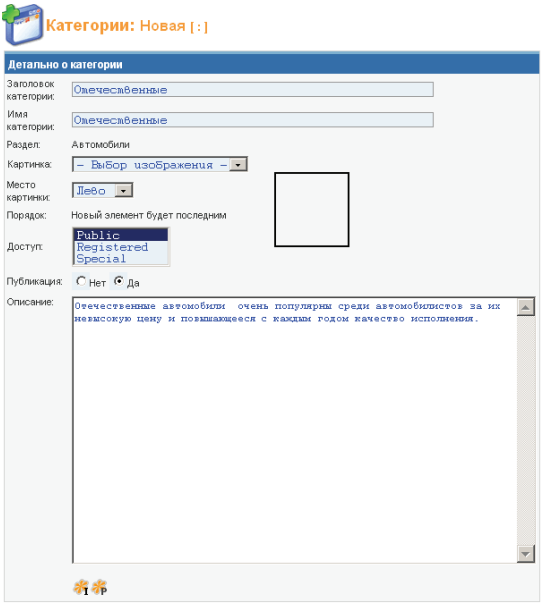


Рис. 13.18. Ввод данных для создаваемой категории

Теперь выберите пункт меню **Материалы | Разделы | Автомобили | Материалы в - Автомобили** (рис. 13.19).

После чего появится менеджер объектов (или материалов). Нажмите кнопку **Добавить**, вы увидите форму добавления нового объекта, в которую необходимо ввести следующие данные (рис. 13.20):

- в поле **Заголовок** и в поле **Алиас заголовка** введите: `Лада`;
- в выпадающем списке **Категория** выберите пункт **Отечественные**;
- в поле для ввода многострочного текста **Текст: (требуется)** введите текст-описание для создаваемого объекта.

Как видите, в данной форме присутствуют два поля для ввода многострочного текста, второе необходимо заполнять, если вы будете использовать так называемые *блоги*. Это способ, при котором сначала выводится часть

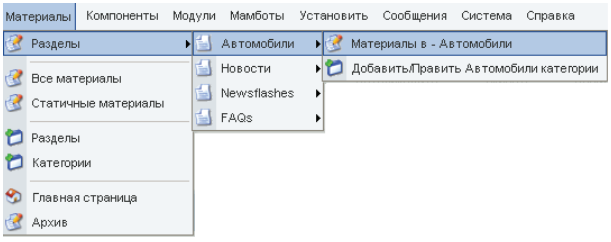


Рис. 13.19. Добавление материалов (объектов)

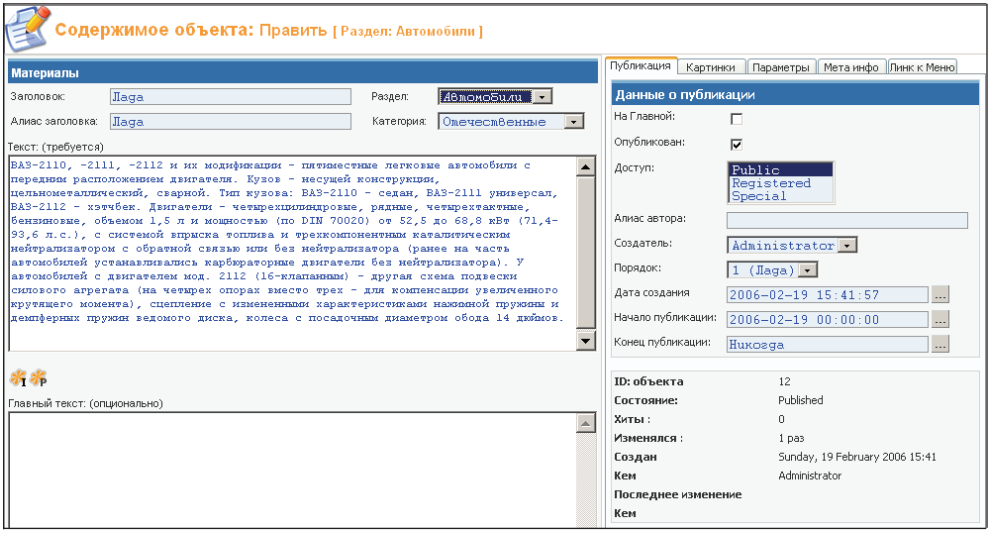


Рис. 13.20. Создание объекта (или материала)

информации по запрошенному пользователем объекту, а также ссылка **Читать далее** или **Подробности**, нажатие на которую приводит к выводу всей информации, которая в этом случае должна быть введена в поле **Главный текст: (опционально)**. Справа вы можете видеть настройки создаваемого объекта, их пока мы трогать не будем. Нажмите кнопку **Сохранить**.

Теперь точно таким же образом добавьте остальные объекты в категорию **Отечественные**, а также заполните по своему усмотрению категории **Иностранные** и **Раритетные**. После этого выберите пункт меню **Разделы | Автомобили | Материалы в - Автомобили** и нажмите левой кнопкой мыши на крестике, расположенном под надписью **На главной** и напротив имени объекта **Лада** (рис. 13.21 и 13.22).

Тем самым мы указываем, что данный объект будет отображаться на главной странице нашего сайта. Давайте проверим это. Снимите блокировку с сайта, установив в конфигурации переключатель **Сайт OFF-Line** в позицию **Нет** и нажав кнопку **Применить**, после чего наберите в браузере **mambo.ru**, вы увидите следующий результат — рис. 13.23.

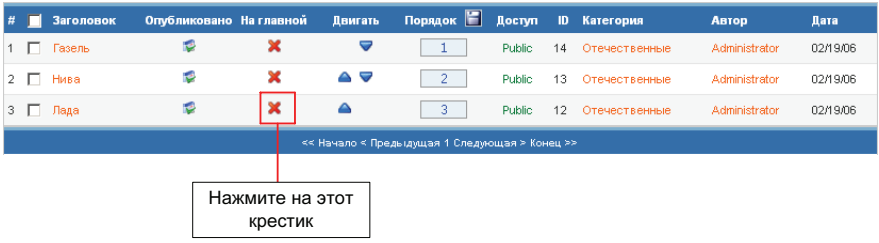


Рис. 13.21. Выбор объекта для публикации на главной странице сайта

Рис. 13.22. Публикация объекта на главной странице сайта

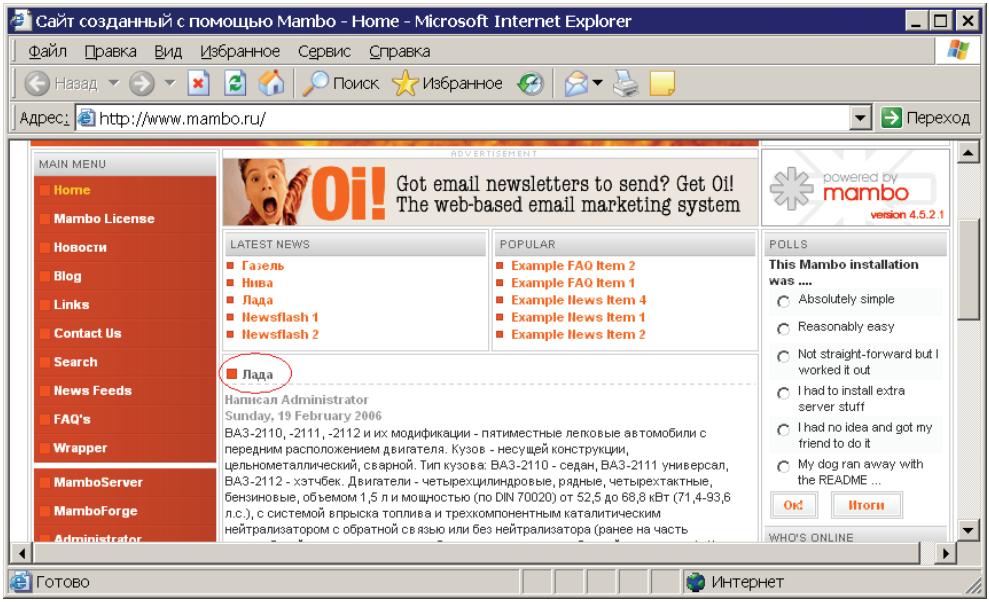


Рис. 13.23. Объект **Лада**, размещенный на главной странице сайта

Теперь поработаем с меню сайта, создадим новый пункт **Автомобили**, назначение которого будет заключаться в выводе иерархии объектов, созданной ранее. Выберите в панели администратора пункт меню **Менеджер меню | Main menu** и нажмите кнопку **Добавить**. Далее необходимо выбрать тип создаваемого пункта меню, установите переключатель **Table - Content Section** (рис. 13.24).

Нажмите кнопку **Следующий**. После чего появится форма для добавления пункта меню, в которую необходимо ввести следующую информацию (рис. 13.25):

- в поле **Name** введите `Автомобили`;
- в списке **Раздел** выберите пункт **Автомобили**.

Нажмите кнопку **Сохранить**. Если вы теперь посмотрите сайт, то заметите, что стал доступен новый пункт, если вы его выберете, то увидите список категорий, если выберете одну из них, то попадете на объект.

Предположим, у нас есть свод правил клуба автолюбителей и его также надо разместить на сайте — добавим его как статичный материал. Для этого

Если вы наведете курсор мыши на значок с изображением буквы ***i****, то сможете увидеть подсказку.*

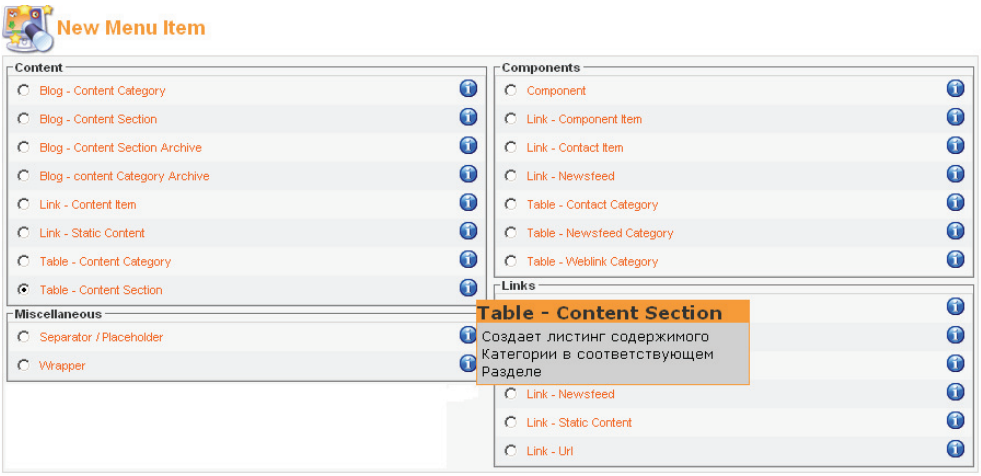


Рис. 13.24. Создание нового пункта меню для сайта

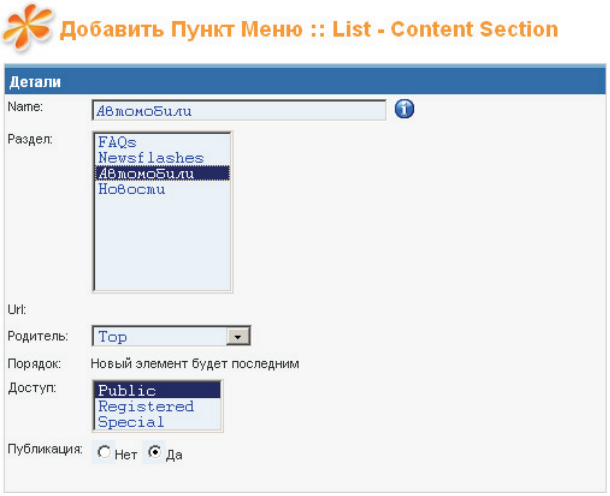


Рис. 13.25. Ввод информации для нового пункта меню

выберите пункт меню **Материалы | Статичные материалы**, после чего нажмите кнопку **Добавить** и заполните форму добавления статичного материала следующим образом:

- в поле **Заголовок** и в поле **Заголовок - псевдоним** введите: `Правила клуба`;
- в поле для ввода многострочного текста **Текст: (обязательно)** введите свод правил клуба автолюбителей.

Теперь привяжем создаваемый объект к пункту меню, для этого справа выберите закладку с именем **Привязка к меню**, в списке **Выбор меню** выберите пункт **main-**

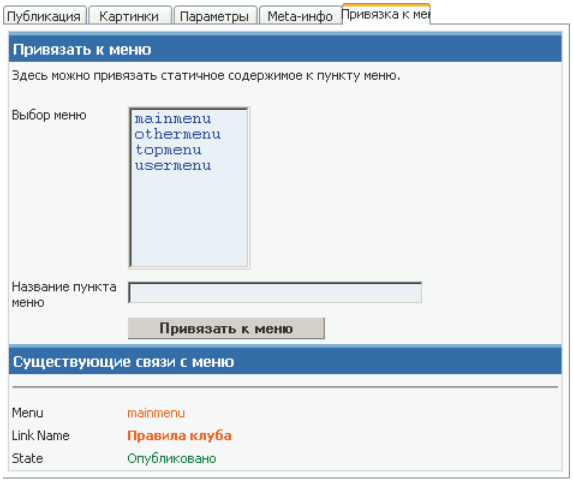


Рис. 13.26. Привязка статического материала (объекта) к меню

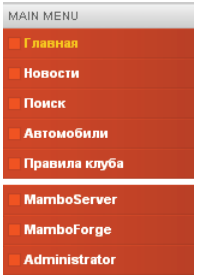


Рис. 13.27. Измененное меню сайта

menu, а в поле **Название пункта** меню введите `Правила клуба`, после чего нажмите кнопку **Привязать к меню**, в результате вы увидите следующее — рис. 13.26.

Нажмите кнопку **Сохранить**. Если вы обратитесь к сайту, то увидите новый пункт, который будет ссылаться на введенные вами правила клуба.

Выберите пункт меню **Менеджер меню | mainmenu**, далее щелкните левой кнопкой мыши на **Home** и измените название данного пункта на русскоязычный вариант — **Главная**, после чего нажмите кнопку **Сохранить**. Точно таким же образом переименуйте пункт **Search** в **Поиск**. Теперь установите флаги напротив пунктов **Mambo License**, **Blog**, **Links**, **Contact Us, News Feeds**, **FAQ's**, **Wrapper** и нажмите кнопку **Скрыть**. Выберите в панели администратора пункт меню **Сайт | Посмотреть сайт | В новом окне**, в результате вы увидите сайт со следующей структурой меню — рис. 13.27.

Как видите, меню состоит из двух частей:

Main Menu — 1

Other Menu — 2

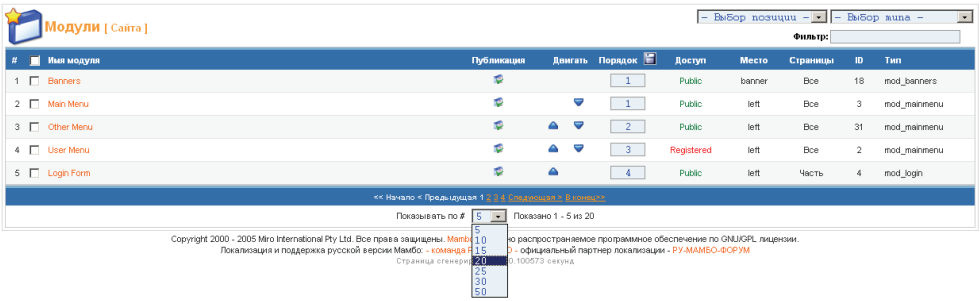


Рис. 13.28. Установка числа отображаемых элементов

Давайте скроем вторую часть — для этого достаточно у модуля, на котором она размещена, установить свойство **Публикация** в **False**. Выберите в панели администратора пункт меню **Модули | Модули сайта**, теперь, чтобы увидеть все модули вашего сайта в выпадающем списке **Показывать по #**, расположенном внизу, выберите пункт **20** (рис. 13.28).

Теперь установите флаг для пункта **Other Menu** и нажмите кнопку **Скрыть**. Войдите в режим предварительного просмотра сайта, вы увидите, что там, где раньше располагалось меню под названием **Other Menu**, образовалось свободное место, давайте разместим в нем изображение автомобиля. Для этого скопируйте в mambo.ru/www картинку, которую вы хотите увидеть на сайте, желательно, чтобы она была небольших габаритов, предположим имя файла картинки picture.jpg. Далее выберите пункт меню **Модули | Модули сайта** и нажмите кнопку **Добавить**, вы попадете на форму добавления модуля, заполните ее следующим образом:

- в поле **Заголовок** введите название создаваемого модуля, например, `Лого`;
- установите переключатель **Показывать заголовок** в позицию **Нет**, чтобы имя модуля не отображалось на сайте;
- в выпадающем списке **Порядок** выберите пункт **3::Other Menu**;
- в поле для ввода многострочного текста **Материалы** введите

```
<img src='picture.jpg'>.
```

Нажмите кнопку **Сохранить** и осуществите предварительный просмотр сайта, вы увидите, что на месте **Other Menu** появилась картинка (рис. 13.29).

Несмотря на то, что сайт получил некую информационную наполненность, этого еще не достаточно — обратимся к баннерам. В Mambo создание баннера совершается в два этапа:

1. Создание клиента баннера.
2. Создание самого баннера.

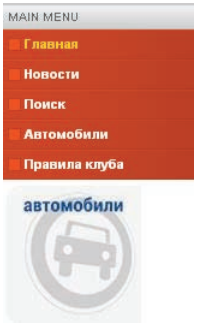


Рис. 13.29. Созданный модуль в действии

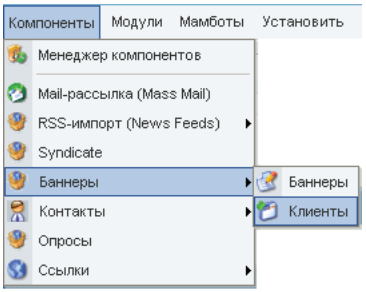


Рис. 13.30. Приступаем к созданию баннера

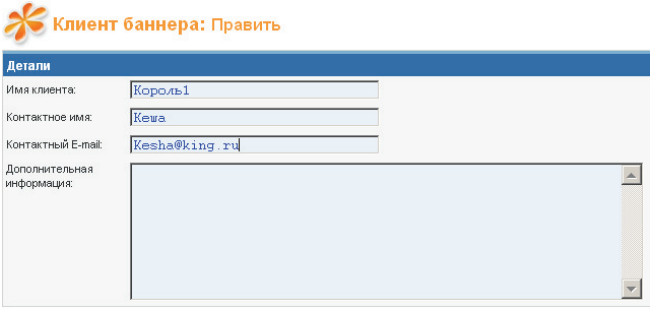


Рис. 13.31. Изменение данных для клиента баннера

В панели администратора выберите пункт меню **Компоненты | Баннеры | Клиенты** (рис. 13.30).

Вы увидите список клиентов, он будет состоять из одного элемента **Miro International Pty**, щелкните на нем левой кнопкой мыши. Вы попадете на форму редактирования данных по клиенту, измените ее следующим образом (рис. 13.31):

- в поле **Имя клиента** введите `Король1`;
- в поле **Контактное имя** введите `Кеша`;
- в поле **Контактный E-mail** введите `Kesha@king.ru`.

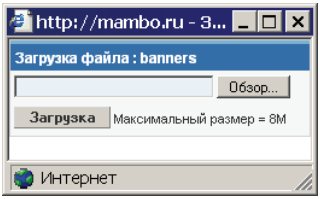


Рис. 13.32. Окно, предназначенное для загрузки файла

Рис. 13.33. Создание нового баннера

Нажмите кнопку **Сохранить**. При желании вы можете добавить еще несколько клиентов. Теперь выберите пункт меню **Компоненты | Баннеры | Баннеры**, вы увидите список баннеров. Скройте уже существующие, т. к. они не понадобятся. Далее нажмите кнопку **Добавить**, вы увидите форму добавления баннера. Загрузим картинку для будущего баннера, лучше чтобы она имела расширение gif или jpeg — нажмите кнопку **Загрузить**, вы увидите окно загрузки файла (рис. 13.32).

Нажмите кнопку **Обзор**, выберите файл и затем нажмите кнопку **Загрузка**, если загрузка была произведена успешно, то вы увидите диалоговое окно, сообщающее вам об этом. Закройте окно загрузки файла и в браузере нажмите кнопку **Обновить**, потому что именно после этого действия файл с изображением можно будет назначить баннеру. В форме добавления баннера введите следующую информацию (рис. 13.33):

- в поле **Имя баннера** введите имя баннера, например `Баннер1`;
- в выпадающем списке **Имя клиента** выберите клиента;
- в выпадающем списке **URL баннера** установите картинку для баннера;
- в выпадающем списке **Показывать баннер** выберите **Да**;

- в поле **URL клика** введите Web-адрес, на который перейдет пользователь после нажатия левой кнопки мыши на баннере.

Нажмите кнопку **Сохранить** и осуществите предварительный просмотр сайта, вы увидите, что теперь там размещен только что созданный баннер.

Обратимся к голосованию — выберите пункт меню **Компоненты | Опросы**, вы попадете в менеджер опросов. Нажмите кнопку **Добавить**, вы увидите форму добавления нового голосования, заполните ее следующим образом:

- в поле **Заголовок** введите вопрос для голосования;
- в поля, расположенные под надписью **Пункты**, введите варианты для голосования;
- в списке **Показывать в меню** выберите, при активизации каких пунктов меню пользователем на сайте он будет видеть голосование.

На рис. 13.34 показан вариант создания нового голосования.

Нажмите кнопку **Сохранить**, вы увидите следующий результат — рис. 13.35.

Как видите, в поле **Публикация** напротив созданного вами опроса стоит крестик, это значит, что пользователь не будет видеть его на сайте. Выполните щелчок левой

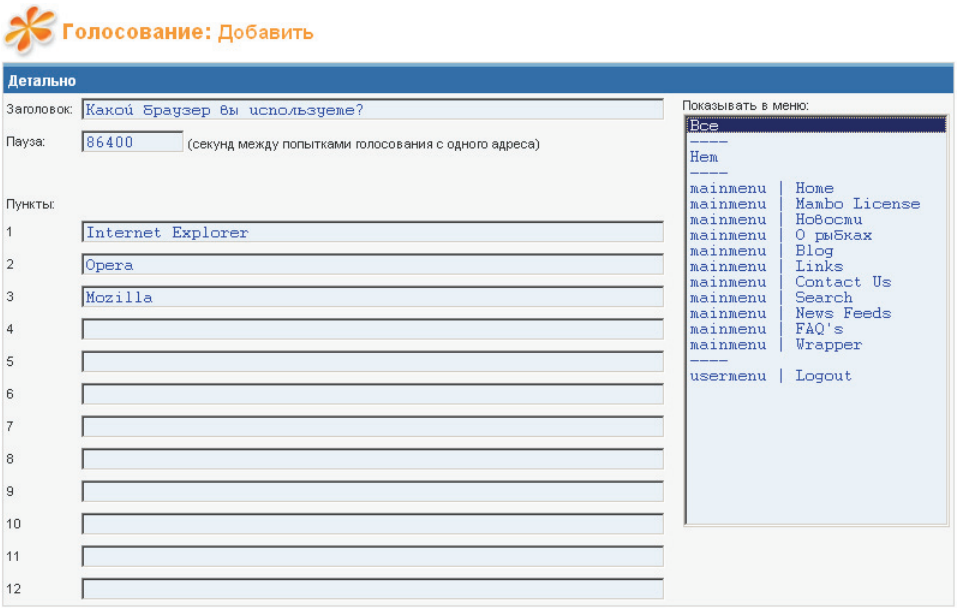


Рис. 13.34. Вариант заполнения данных для создания голосования

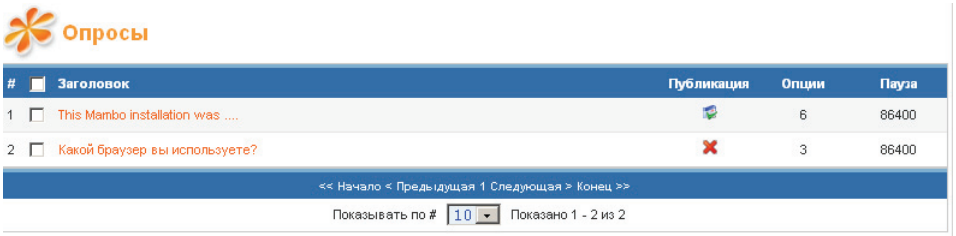


Рис. 13.35. Результат добавления нового голосования

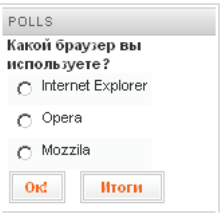


Рис. 13.36. Созданное голосование в действии

ФОРУМ
[лат. forum] -
1) площадь в древнем Риме,
2) место выступлений, высказываний;
3) собрание, съезд.
Словарь иностранных слов

кнопкой мыши на крестике — он сменится на значок публикации, для первого же опроса наоборот установите крестик. Если вы осуществите предварительный просмотр сайта, то увидите только что созданное голосование (рис. 13.36).

Теперь подключим к нашему сайту форум, разработанный в *главе 10*. Выберите пункт меню **Менеджер меню | mainmenu**, после этого нажмите кнопку **Добавить**, далее установите тип создаваемого пункта меню равным **Wrapper**, нажмите кнопку **Следующий**, появится форма добавления нового пункта меню, заполните ее следующим образом (рис. 13.37):

- в поле **Имя** введите `Форум`;
- в поле **Куда ведет** введите `www.forum.ru`;
- в выпадающем списке **Родитель** оставьте значение по умолчанию, равное **Top**.

Нажмите кнопку **Сохранить** и осуществите предварительный просмотр сайта, вы увидите следующий результат — рис. 13.38.

Как видите, форум, разработанный в *главе 10*, спокойно себя чувствует на сайте, сконструированном с помощью Mambo.

Теперь выберите в панели администратора пункт меню **Модули | Модули сайта**, щелкните левой кнопкой мыши на модуле **Main Menu** и в выпадающем списке **Место** выберите **right**, после чего нажмите кнопку **Сохранить** и посмотрите на сайт. Вы увидите, что меню будет расположено справа. Точно так же зайдите в форму редактирования модуля **Polls** и в выпадающем списке **Место** установите пункт **left**, далее

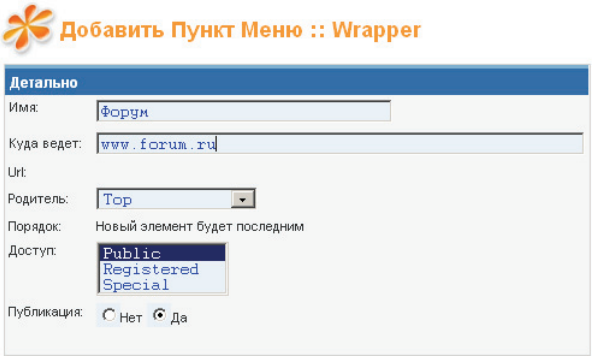


Рис. 13.37. Создание пункта меню типа **Wrapper**

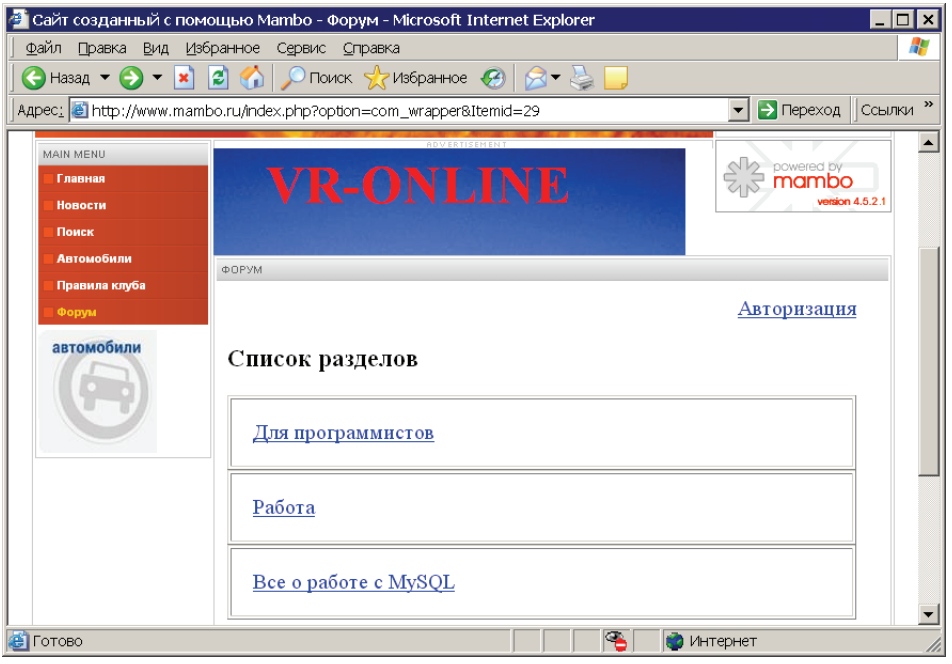


Рис. 13.38. Форум, встроенный в Mambo-сайт

нажмите кнопку **Сохранить**, после чего голосование будет размещено в левой части вашего сайта.

Изменим немного внешний вид сайта, а именно поменяем оранжевую картинку, изображенную на рис. 13.39, на свою.

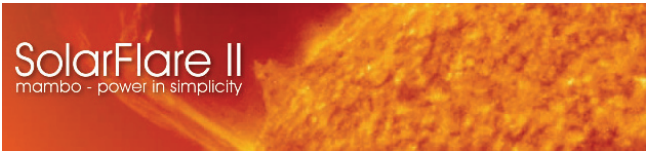


Рис. 13.39. Элемент дизайна стандартного шаблона Mambo

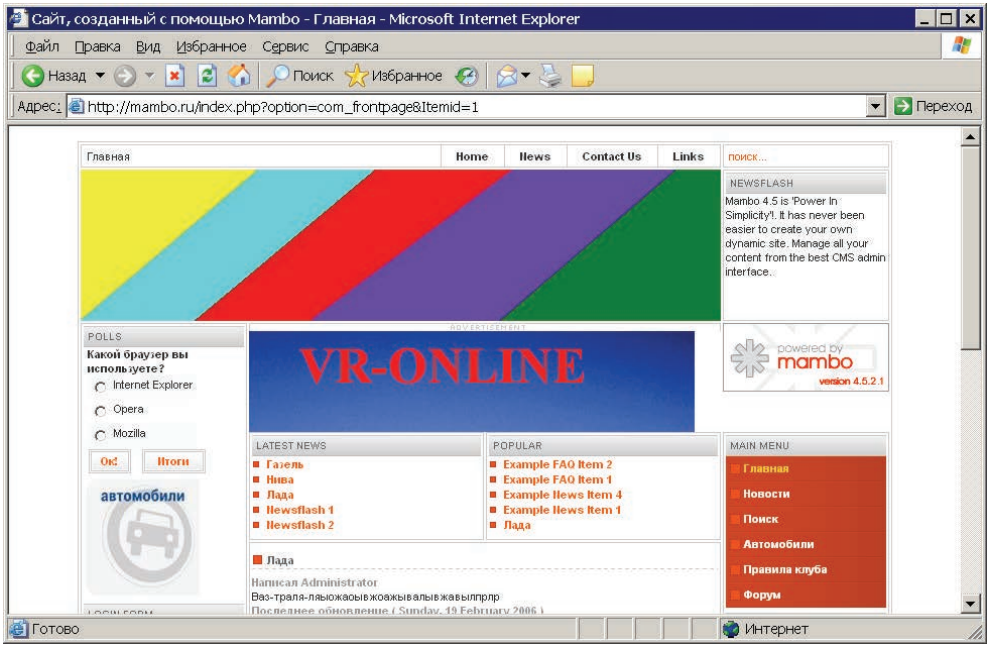


Рис. 13.40. Легкие изменения, внесенные в стандартный шаблон Mambo

Для этого, с помощью проводника, обратитесь по следующему пути: **mambo.ru\www\templates\rhuk_solarflare_ii\images**, вместо файла header_short.jpg сохраните свою картинку с такими же размерами, как и у оригинала. На рис. 13.40 вы можете видеть результат этого действия.

13.4. Устанавливаем модуль для Mambo

В этом разделе установим дополнительный модуль для Mambo, который осуществляет вывод прогноза погоды с сайта **gismeteo.ru**, дистрибутив данного модуля можно взять с сайта **ru-mambo.ru**, раздел **Файлы | Модули** (рис. 13.41).

Дистрибутив модуля — это обычный ZIP-архив. Весь процесс инсталляции модуля описан в XML-файле, находящемся в этом архиве. В панели администратора выберите пункт меню **Модули | Менеджер модуля**, вам станет доступен менеджер модулей, нажмите в нем на кнопку **Обзор**, расположенную напротив поля **Файл пакета** (рис. 13.42).

В появившемся окне **Выбор файла** выберите файл mod_gisweather_032.zip, который вы скачали, и нажмите кнопку **Открыть**, далее нажмите кнопку **Загрузить & Установить**. После того как модуль будет установлен, вы увидите следующее — рис. 13.43.


GisWeather v. 0.3.2
Версия Mambo: 4.5.2
Автор: Денис Киселев
Описание: Модуль вывода погодного информера с сайта gismeteo.ru.
Параметры:
1. Возможность включения системы кеширования.
2. Настройка на любой город доступный в каталоге Гисметео.
3. Настройка одного из 10 вариантов вывода изображения информера.
Сайт: http://ru-mambo.ru
Скачать: mod_gisweather_032.zip (6.96 kb)

Рис. 13.41. Описание модуля **GisWeather**

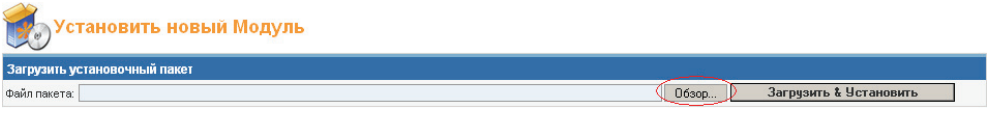


Рис. 13.42. Выбор дистрибутива модуля для его установки

Рис. 13.43. Сообщение о завершении установки модуля

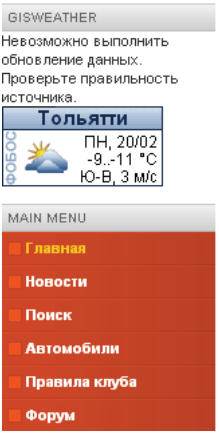


Рис. 13.44. Модуль **GisWeather** в работе

Нажмите на ссылку **Продолжить**. Теперь выберите пункт меню **Модули | Модули сайта** и отредактируйте появившийся модуль с именем **GisWeather** таким образом, чтобы он располагался справа, также установите публикацию для него. На рис. 13.44 вы можете видеть модуль **GisWeather** в работе.

13.5. Устанавливаем компонент для Mambo

Май 200?

25	Прийти
26	Увидеть
27	Победить!

Сейчас мы установим дополнительный компонент **ExtCalendar**, который предназначен для организации на сайте гостевой книги, его дистрибутив можно взять с сайта **ru-mambo.ru**, раздел **Файлы | Прочее | ExtCalendar**. После закачки файла ExtCalendar07_RUS.zip (на момент написания книги архив компонента имел именно такое название) его надо будет распаковать, т. к. он состоит из двух компонентов. Далее в панели администратора выберите пункт меню **Компоненты | Менеджер компонентов**, в появившемся менеджере компонентов нажмите кнопку **Обзор**, расположенную напротив поля **Файл пакета** (рис. 13.45).

Вам станет доступно окно **Выбор файла**, в котором необходимо выбрать файл ExtCalendar_Component_v0.7_RUS.zip и нажать кнопку **Открыть**. Далее нажмите кнопку **Загрузить & Установить**. После того как компонент будет установлен, вы увидите следующее — рис. 13.46.

Нажмите на ссылку **Продолжить**. Далее в панели администратора выберите пункт меню **Менеджер меню | mainmenu**, нажмите кнопку **Добавить**, установите пере-

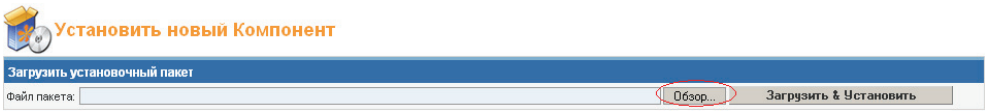


Рис. 13.45. Выбор дистрибутива компонента для его установки

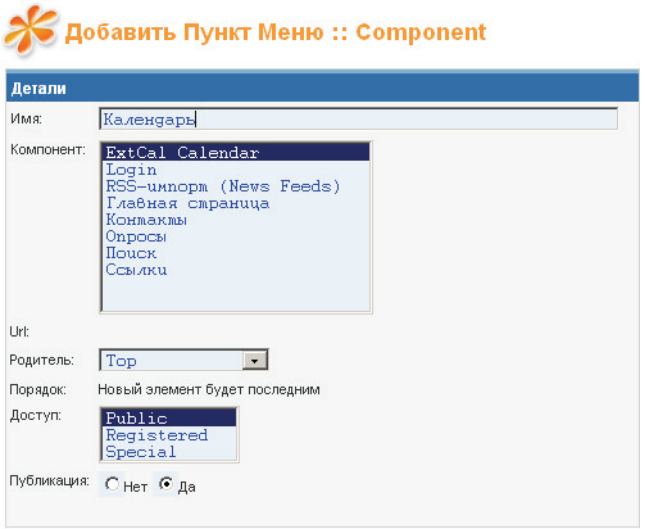


Рис. 13.46. Сообщение о завершении установки компонента

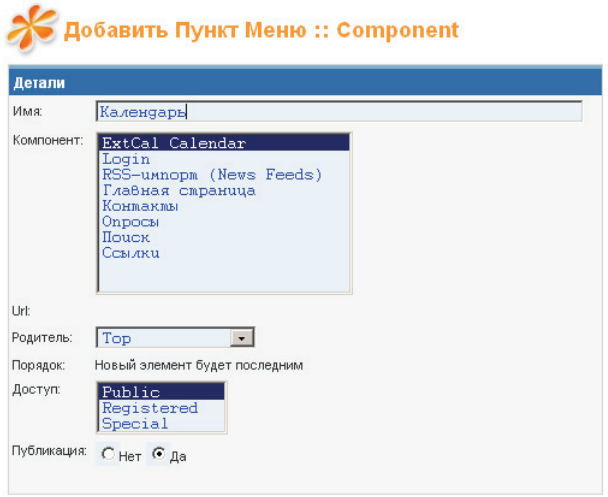


Рис. 13.47. Создание пункта меню, ссылающегося на компонент

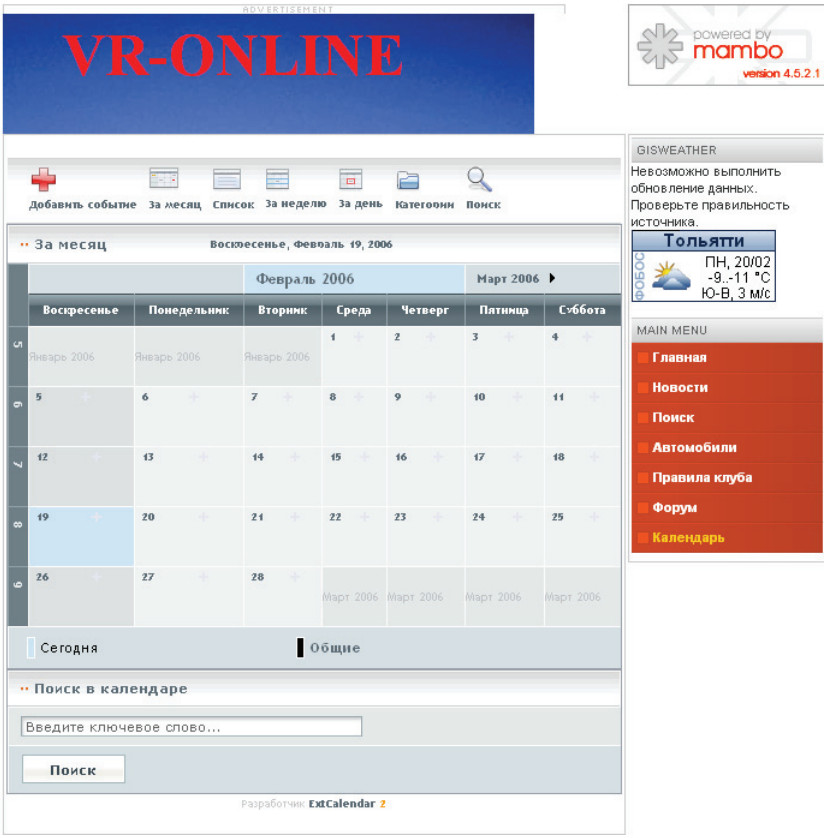


Рис. 13.48. Компонент **ExtCal Calendar** в действии

ключатель в позицию **Component** и нажмите кнопку **Следующий**, после чего в форме добавления нового пункта меню введите следующую информацию (рис. 13.47):

- в поле **Имя** введите `Календарь`;
- в списке **Компонент** выберите пункт **ExtCal Calendar**.

Нажмите кнопку **Сохранить**. Теперь с новым компонентом можно работать на сайте через только что созданный пункт **Календарь**. На рис. 13.48 вы можете увидеть компонент **ExtCal Calendar** в действии.

Шаблоны [Сайта]

Посмотр шаблона

#		Имя	По умолчанию	Назначено	Автор	Версия	Дата	URL автора
1	○	mambospan_glass			Shashikiran Umakanth	1.0.	08/12/05	http://mambo.medspan.info
2	○	Purple Heart			Ben Brown	1.0	09/16/2004	www.newhopeowasso.com
3	◉	rhuk_solarflare_ii	✔		rhuk	2.01	11/02/04	http://www.mambodev.com
4	○	starwars_force			Dragon Company & Mijean	1.0	2005/05/18	http://www.dragon-company.com

<< Начало < Предыдущая 1 Следующая > Конец >>

Рис. 13.49. Менеджер шаблонов

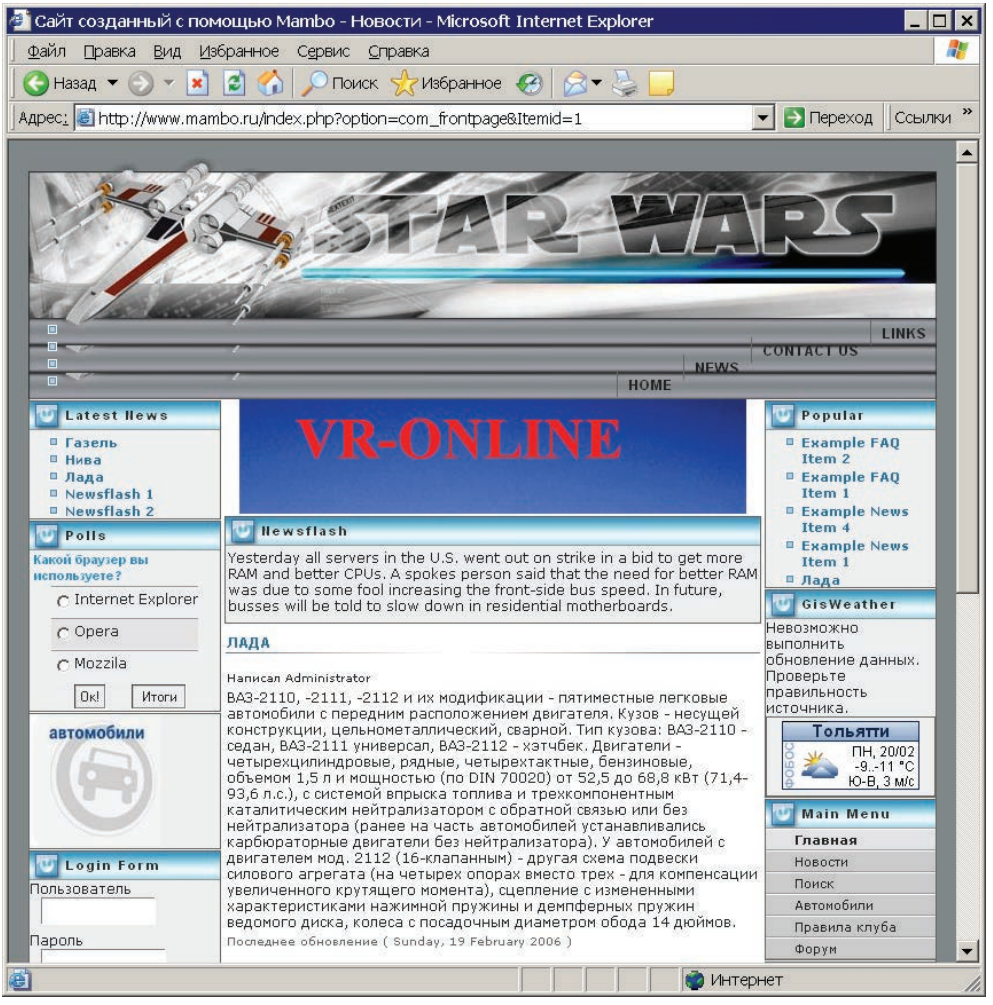


Рис. 13.50. Шаблон **starwars_force** в действии

13.6. Устанавливаем шаблоны для Mambo

Для того чтобы установить шаблон, предварительно надо его скачать. Шаблон — это обычный ZIP-архив. Для установки шаблона необходимо в панели администратора выбрать пункт меню **Сайт | Шаблоны | Добавить**, далее выбрать архив шаблона и произвести инсталляцию.

Для того чтобы назначить шаблон сайту, необходимо выбрать пункт меню **Сайт | Шаблоны | Шаблоны сайта**, после чего вы попадете в менеджер шаблонов (рис. 13.49).

Установите переключатель напротив того шаблона, который нужно применить к сайту, и нажмите кнопку **По умолчанию**. Если же вы хотите применить шаблон только к определенному разделу сайта, то после выбора шаблона нажмите кнопку **Назначить**, затем выберите, с каким пунктом меню нужно ассоциировать данный шаблон, и нажмите кнопку **Сохранить**. Таким образом вы можете создать уникальный дизайн для каждой странички вашего Web-сайта в отдельности.

На рис. 13.50 вы можете видеть как будет выглядеть разработанный в данной главе сайт с примененным к нему шаблоном **starwars_force**.

13.7. В заключение

Если вы хотите более глубоко изучить Mambo, то в этом вам поможет ресурс **http://e-planet.ru**. На данном сайте вы сможете подписаться на электронные уроки (**http://e-planet.ru/hosting/mambo/**), посвященные Mambo: весь курс состоит из 7 уроков, в каждом из которых очень подробно и доступно описывается работа с Mambo, но это еще не все. Также на сайте содержатся видеоролики (**http://e-planet.ru/hosting/mambo/**), их более 40, в которых наглядно рассматриваются практические аспекты работы с Mambo.

Различные элементы к Mambo (компоненты, шаблоны, модули и т. д.) можно найти на следующих ресурсах:

- **http://mamboserver.ru**
- **http://mambasana.ru**
- **http://mamboportal.com**
- **http://www.freemambo.com**
- **http://mamboserver.com**
- **http://forum.mamboserver.com**

Глава 14
Закачиваем сайт на хостинг

Итак вы создали свой сайт и готовы разместить его в Интернете. Все, что осталось сделать, — это найти подходящий хостинг и закачать туда ваш Web-ресурс. В данной главе рассматривается ряд вопросов, знание которых поможет вам в этом.

14.1. Бесплатный хостинг от Holm.ru

Если вы хотите, чтобы ваш сайт был доступен общественности, то не обязательно для этого покупать хостинг, можно воспользоваться бесплатным, заодно вы оцените свои силы в сайтостроении. В данном разделе будет рассмотрена работа с **holm.ru** — ресурсом, который предоставляет бесплатные хостинги.

Зайдите на сайт **www.holm.ru**, вам нужно найти форму с названием **Бесплатный хостинг** (рис. 14.1).

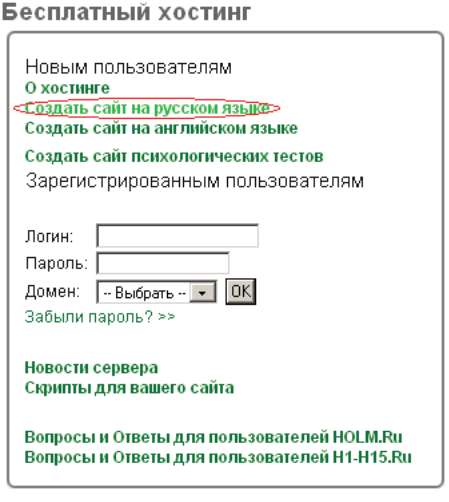


Рис. 14.1. Начало регистрации на бесплатном хостинге **holm.ru**

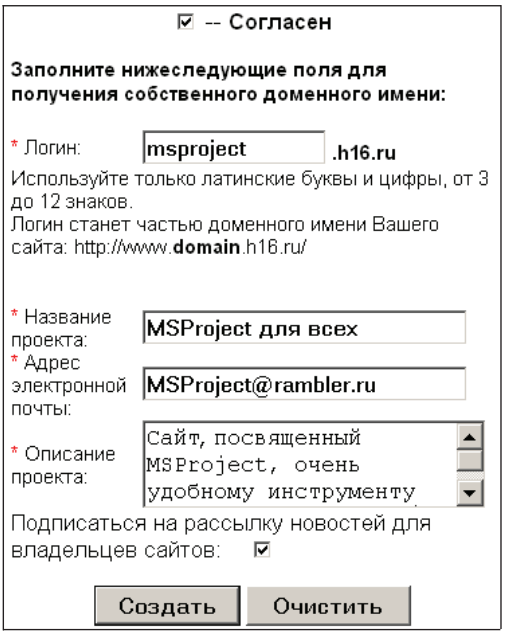


Рис. 14.2. Вариант заполнения формы для регистрации сайта на **holm.ru**

Щелкните левой кнопкой мыши на ссылке **Создать сайт на русском языке**. Вы попадете на форму регистрации, где вам предложат ознакомиться с условиями предоставления услуг и ввести некоторые данные о себе. Установите переключатель **Согласен**, в поле **Логин** введите имя вашего сайта, заполните поле **Название проекта**, в поле **Адрес электронной почты** введите свой e-mail, заполните поле для ввода многострочного текста **Описание проекта**. На рис. 14.2 показан вариант заполнения формы регистрации сайта.

Нажмите кнопку **Создать**. Появится новая форма, где вам будет необходимо в специальное поле ввести цифрами контрольное число, которое изображено в виде текста (рис. 14.3).

Сделайте это и нажмите кнопку **Создать**. После чего вы увидите надпись **Домен зарегистрирован**. Через некоторое время к вам на почтовый ящик придет электронное письмо, которое будет содержать данные доступа к серверу и некоторую полезную информацию, например, правила, которые необходимо соблюдать, чтобы ваш сайт не был удален (рис. 14.4).

Теперь вы можете зайти на свой хостинг, используя логин и пароль, указанные в письме (рис. 14.5).

Будьте особо внимательны при выборе значения выпадающего списка ***Домен****.*

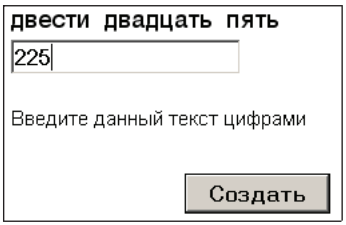


Рис. 14.3. Ввод контрольного числа

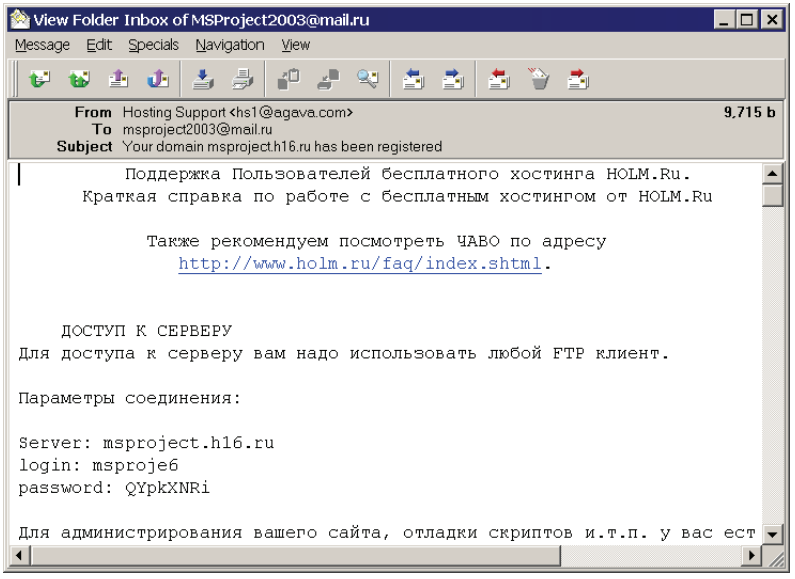


Рис. 14.4. Письмо, в котором содержатся данные для доступа к серверу **holm.ru**

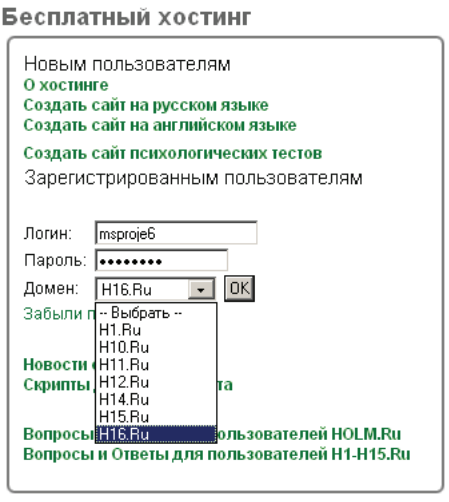


Рис. 14.5. Вход на свой хостинг

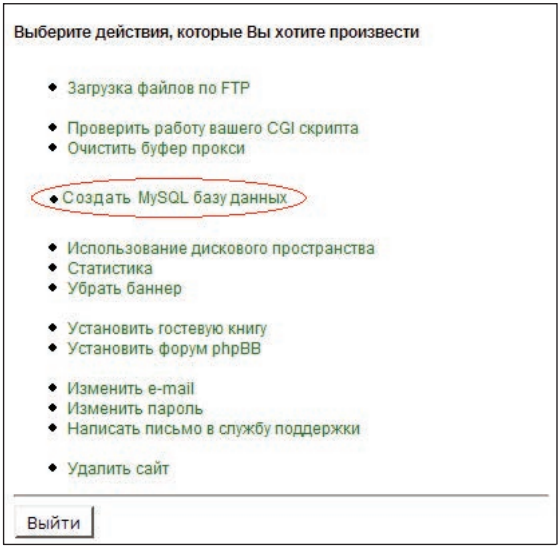


Рис. 14.6. Меню, доступное после авторизации на **holm.ru**

После того как вход осуществлен, вам будет доступно небольшое меню (рис. 14.6).

Давайте создадим на только что полученном хостинге базу MySQL — необходимость в ее использовании возникает очень часто. Выберите пункт **Создать MySQL базу данных**, после чего вы попадете на форму, в которой увидите надпись **Вы запросили создание базы данных** и кнопку **Создать**, на которую вам необходимо нажать. После чего появится информация о только что созданной базе данных (обязательно сохраните ее) — рис. 14.7.

Замечание

Как видите, с помощью данного меню вы можете даже установить форум phpBB, делается это очень удобно — всю заботу берет на себя специальный мастер.

Для того чтобы закачать сайт на хостинг, подойдет любой FTP-клиент. Рассмотрим реализацию этого действия с помощью браузера Internet Explorer, запустите его, затем введите `ftp://адрес_вашего_сайта` (например, `ftp://msproject.h16.ru`), после чего на экране появится окно, наподобие следующего — рис. 14.8.

Нажмите кнопку **OK**, далее выберите пункт меню **Файл | Войти как**, на экране появится окно **Вход** (рис. 14.9).

Введите в него имя пользователя и пароль, информацию о которых вы найдете все в том же электронном письме, далее нажмите кнопку **Вход**. После чего вам только останется скопировать ваш сайт на сервер.

Стоит отметить, что **holm.ru** — не единственный ресурс, где можно получить бесплатный хостинг. Также это можно сделать на **hut.ru** (рис. 14.10), **siteburg.com** и на множестве других, которые очень легко найти, введя в любом поисковом сервере фразу "бесплатный хостинг".

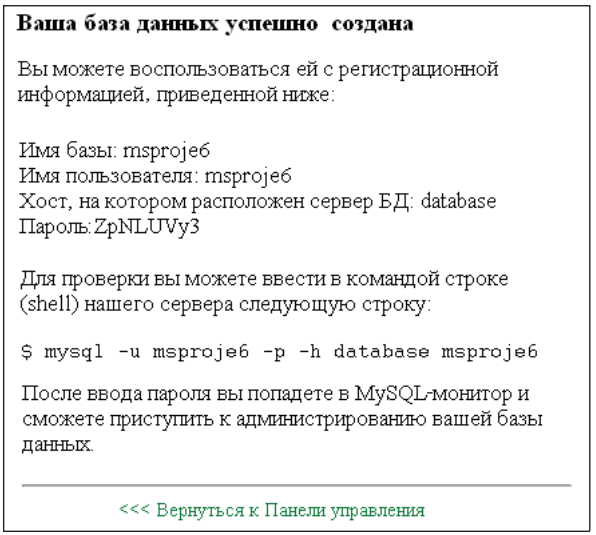


Рис. 14.7. Создание MySQL-базы на хостинге от **holm.ru**

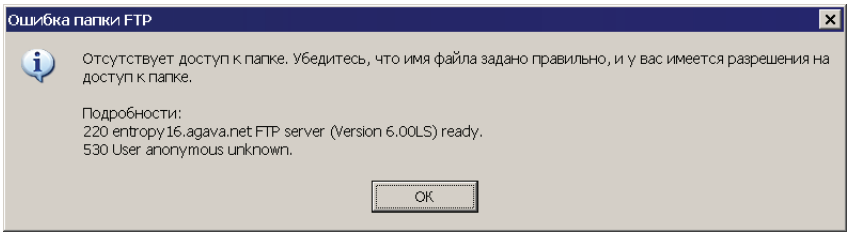


Рис. 14.8. Неудачная попытка подключения к FTP-серверу

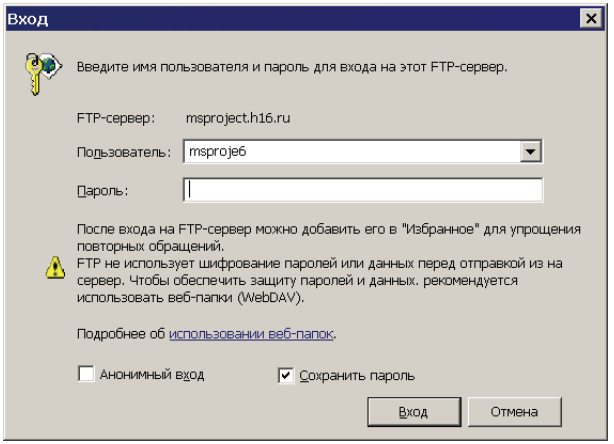


Рис. 14.9. Окно **Вход**

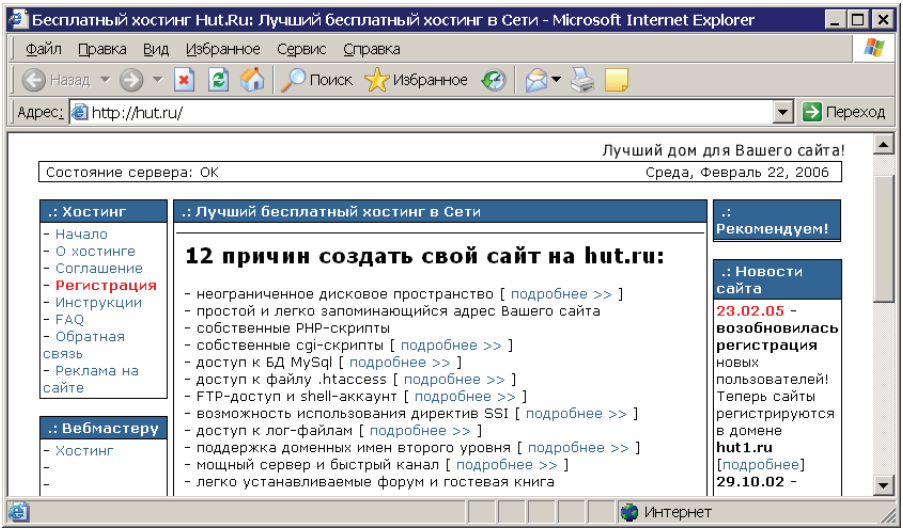


Рис. 14.10. Бесплатный хостинг от **hut.ru**

14.2. cPanel

При работе с платным хостингом закачка вашего сайта также осуществляется через любой FTP-клиент, в качестве которого можно использовать и обычный браузер. Существенное различие заключается в том, что вам доступна специальная панель управления, позволяющая взаимодействовать с вашим сайтом через Web-интерфейс. Одной из самых популярных среди них является cPanel. Она предоставляет пользователю огромное количество функций: начиная от работы с учетными записями электронной почты и заканчивая администрированием MySQL.

Официальным сайтом cPanel является **http://www.cpanel.net**, там же можно скачать документацию по данной системе (**http://www.cpanel.net/docs.htm**).

Итак, вы приобрели хостинг и закачали на него сайт. О том где находится cPanel, вам сообщит провайдер, у которого была совершена покупка, обычно это `имя_сайта\cpanel`. Первое, что вам необходимо будет сделать, — это пройти авторизацию (рис. 14.11).

После того как авторизация пройдена (имя пользователя и пароль вам также предоставит провайдер). Вы увидите что-то наподобие следующего — рис. 14.12.

Слева вы можете видеть статистическую информацию, а справа — набор иконок, каждая из которых предназначена для активизации той или иной функции, предоставляемой cPanel.

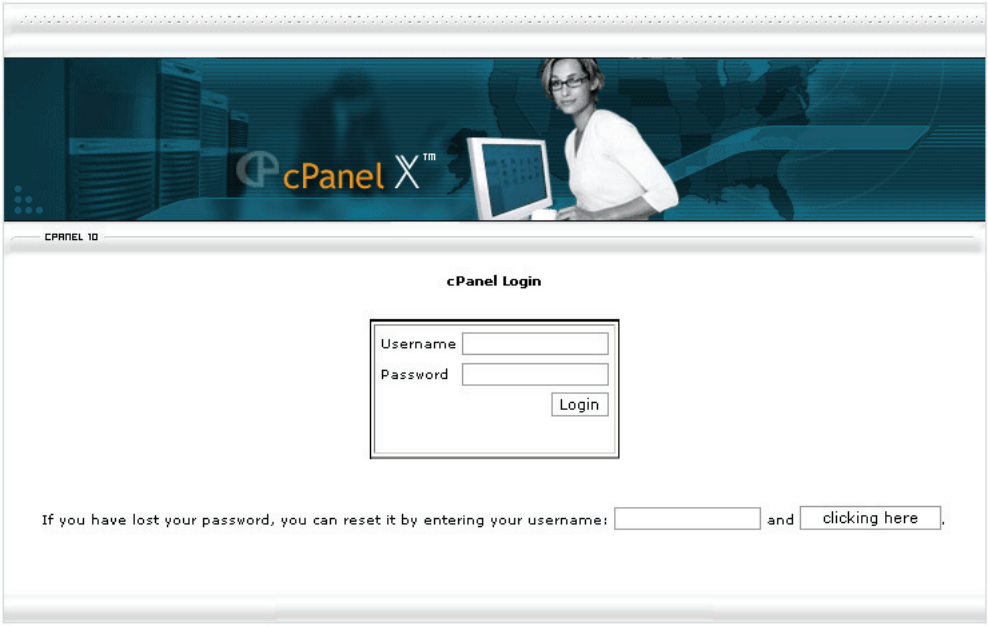


Рис. 14.11. Форма авторизации для входа в cPanel

Рассмотрим одно из основных действий — это создание базы данных. Нажмите левой кнопкой мыши на пиктограмме с текстом **Базы данных MySQL** (рис. 14.13).

Вы попадете в менеджер баз данных. В поле **Db** (рис. 14.14) введите имя создаваемой базы.

Нажмите кнопку **Add Db**, вы увидите сообщение о том, что база создана (рис. 14.15).

Нажмите левой кнопкой мыши на ссылку **Go Back**. Теперь вы можете видеть информацию о созданной базе (рис. 14.16).

Как видите, к имени базы данных автоматически добавился префикс, как правило, изменить его нельзя. Также рядом с именем только что созданной базы можно увидеть ссылки — кнопки, предназначенные для работы именно с этой базой данных (например, нажатие на ссылку **Delete** приведет к тому, что база данных `kalipsoi_TESTBASE` будет удалена).

Теперь необходимо создать пользователя MySQL, под учетной записью которого будет осуществляться работа с созданной базой. Для этого предназначена форма **Users**, состоящая из двух полей: **UserName** и **Password** (рис. 14.17).

Общая информация об аккаунте:

Тарифный план	NOVA
Shared Ip Address	81.177.4.39
Поддоменов	0 / unlimited
Паркованных доменов	0 / 6
Дополнительных доменов	0 / 6
Баз данных Mysql	1 / 5
Postgresql Databases	0 / 5
Использовано места	37.21 Мегабайты
Использовано места под SQL	0.27 Мегабайты
Доступно места	602.79 Мегабайты
Использовано трафика	483.48 Мегабайты
Емайл аккаунтов:	2 / unlimited
Переадрсеторы почты	5
Автоответчики	0
Фильтров почты	0
ФТП аккаунты	0 / unlimited

Общая информация о сервере:

Операционная система	Linux
Форматирование диска (все данные будут утеряны !!!)	посмотреть
Версия Kernel	2.6.11-1.1369_FC4
Machine Type	x86_64
Версия Apache	1.3.34 (Unix)
Версия PERL	5.8.6
Путь к PERL	/usr/bin/perl
Путь к sendmail	/usr/sbin/sendmail
Модули Perl	посмотреть
Версия PHP	4.4.1
Версия MySQL	4.1.14-standard
Версия cPanel	10.8.1-RELEASE 113
Theme	cPanel X v2.5.0
Документация	посмотреть
cPanel Pro	1.0 (RC36)

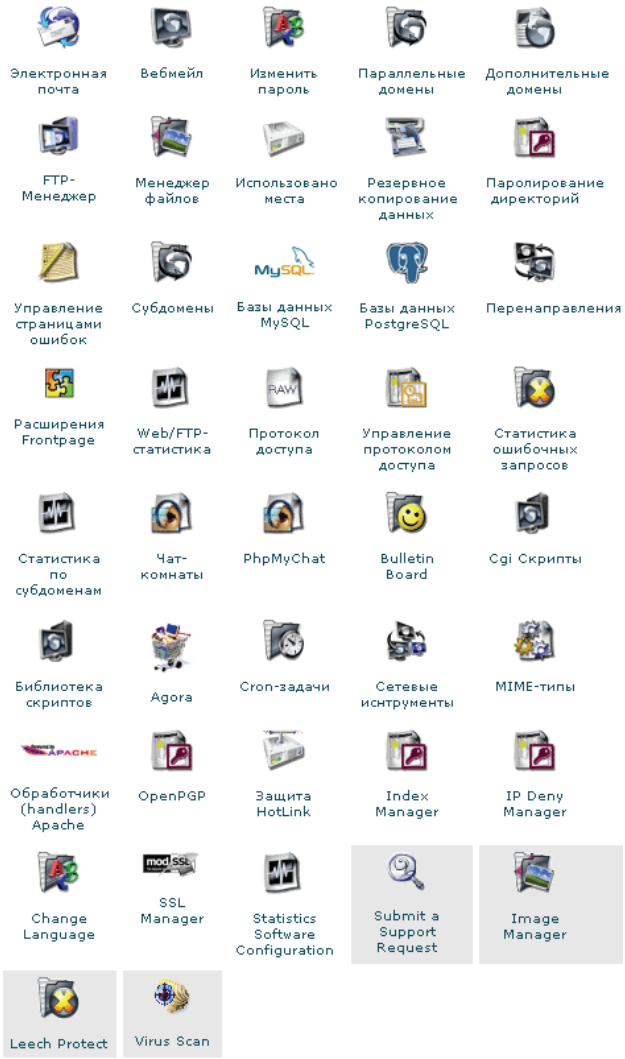


Рис. 14.12. cPanel

Рис. 14.13. Пиктограмма, нажатие которой позволит перейти к работе с базой данных MySQL

Рис. 14.14. Поле для ввода имени создаваемой базы данных

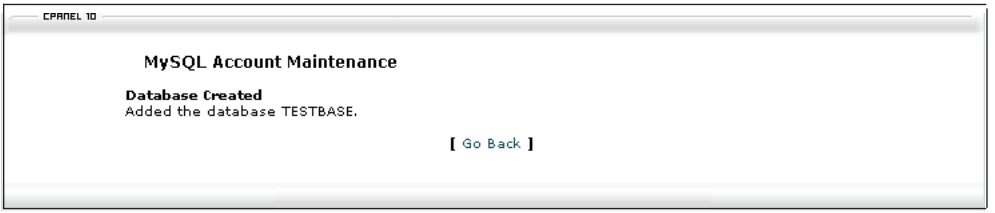


Рис. 14.15. Пример создания базы данных с именем `TESTBASE`

Рис. 14.16. Информация о созданной базе данных

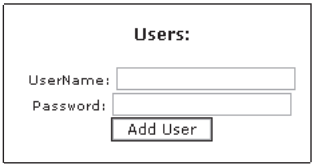


Рис. 14.17. Форма создания учетной записи пользователя MySQL

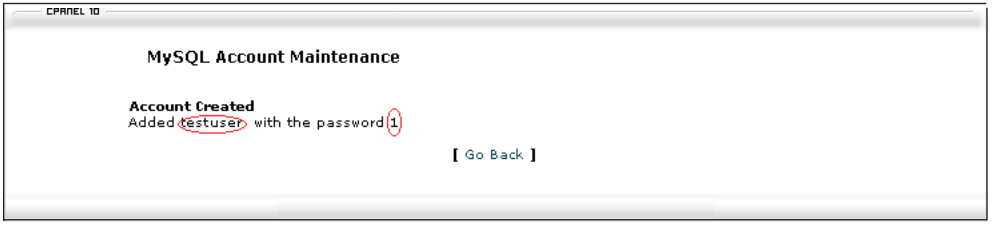


Рис. 14.18. Пример создания учетной записи пользователя MySQL с именем `testuser` и паролем `1`

Рис. 14.19.
Информация о созданной учетной записи

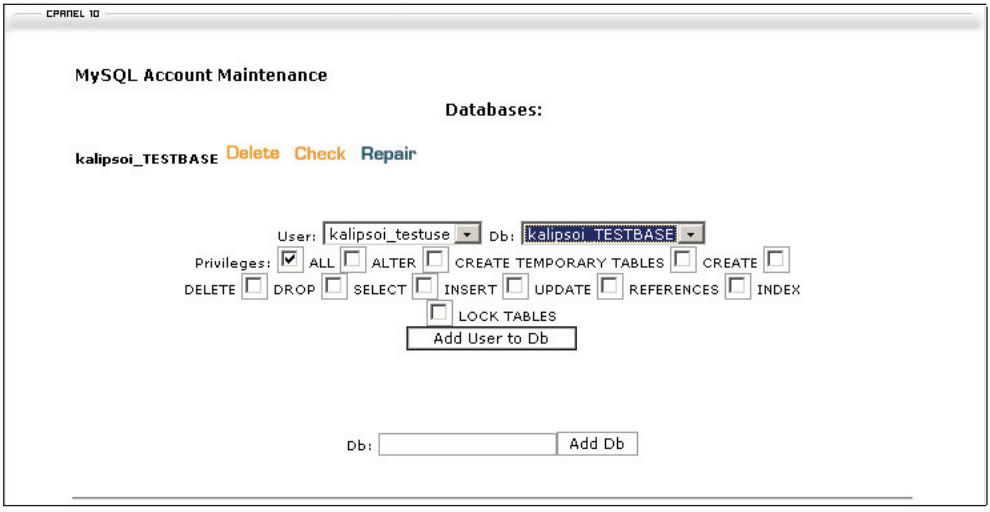


Рис. 14.20.
Подключение пользователя к базе данных с заданными правами доступа

Введите имя и пароль для пользователя, учетную запись которого вы хотите создать. Далее нажмите кнопку **Add User**, после чего появится сообщение о том, что действие успешно совершено (рис. 14.18).

Нажмите левой кнопкой мыши на ссылку **Go Back**. Теперь вы можете увидеть информацию о созданной учетной записи (рис. 14.19).

Как видите, к имени пользователя также был добавлен префикс, к тому же имя пользователя было обрезано (вместо `test_user` получился `kalipsoi_testuse`), поэтому не удивляйтесь, если в результате получается не совсем то, что вы ожидали увидеть.

Остался последний штрих — подключить созданного пользователя к базе данных с заданными правами доступа. Для этого в выпадающем списке **User** выберите пользователя, а в выпадающем списке **Db** — имя базы данных. С помощью флагов **Privileges** установите права работы пользователя с базой (рис. 14.20).

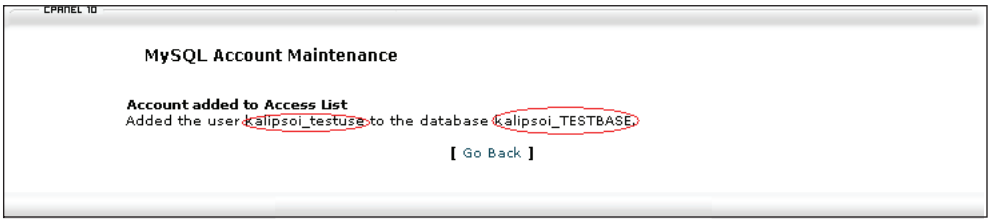


Рис. 14.21. Успешное подключение пользователя к базе данных

Рис. 14.22. Информация о параметрах подключения пользователя к базе данных

Нажмите кнопку **Add User to Db**, после чего появится сообщение о том, что действие успешно совершено (рис. 14.21).

Нажмите левой кнопкой мыши на ссылку **Go Back**. В результате под названием базы данных вы увидите информацию о пользователе, который теперь может работать с ней (рис. 14.22).

Заключение

Я надеюсь, что вы нашли для себя в данной книге полезную и интересную информацию. Все ваши пожелания и комментарии рад буду увидеть на форуме **http://www.vr-online** в разделе **Книги от LittleBudda**, там же можно всегда со мной пообщаться и задать любой интересующий вас вопрос. Я буду благодарен за любые ваши замечания, их я смогу учесть в дальнейшей своей работе.

Связаться с автором можно через издательство "БХВ-Петербург": **mail@bhv.ru**.

Приложение 1
Настройка PHP

В данном приложении будут рассмотрены основные директивы, влияющие на работу PHP. Как вы уже знаете, они находятся в файле php.ini. В случае использования Денвера, его можно найти по следующему пути: usr\local\php\php.ini. Для удобства описанные директивы разбиты на группы. Стоит отметить, что значения директив чувствительны к регистру, поэтому, например, `file_uploads=On` и `file_uploads=on` не являются одним и тем же.

Отправка писем с помощью функции *mail()*:

- `SMTP` — имя или IP-адрес SMTP-сервера (директива используется только для WINDOWS);
- `smtp_port` — номер порта, с которым работает SMTP-сервер (директива используется только для WINDOWS);
- `sendmail_from` — от кого отправлено письмо (директива используется только для WINDOWS);
- `sendmail_path` — путь, по которому располагается программа sendmail, предназначенная для отправки писем (директива используется в операционных системах, отличных от WINDOWS; при ее использовании три описанных ранее директивы — `SMTP`, `smtp_port`, `sendmail_from` — должны быть закомментированы; это делается с помощью точки с запятой, которую указывают перед именем директивы; то же самое верно и для обратного случая: когда `sendmail_path` не используется, ее надо закомментировать).

Закачка файлов:

- `file_uploads` — разрешает/запрещает загрузку файлов, это достигается установкой значений `On` или `Off` (также возможно использование их альтернативных вариантов `1` или `0`, `Yes` или `No`, `True` или `False`);
- `upload_tmp_dir` — директория, в которую будут помещаться закачанный на сервер файл для временного хранения;
- `upload_max_filesize` — максимальный размер файла, который может быть закачан на сервер (эта директива зависит от значения директивы `post_max_size`, значение последней будет учитываться в первую очередь);
- `post_max_size` — ограничение на объем данных, передаваемых методом `POST`.

Запреты:

- `disable_functions` — содержит имена запрещенных к выполнению функций, которые перечисляются через запятую, например, `disable_functions = fopen, fgets` приведет к тому, что функции `fopen()` и `fgets()` работать не будут.

Сессии:

- `session.save_path` — место хранения сессий на сервере, например `session.save_path = /tmp`;
- `session.gc_maxlifetime` — время жизни сессии, указывается в секундах, например `session.gc_maxlifetime = 300` будет означать, что сессия будет действительна в течение 5 минут (60 сек * 5 мин);
- `session.use_trans_sid` — будет ли идентификатор сессии передаваться в GET-запросе, если у пользователя в браузере отключены COOKIES, для данной директивы возможны следующие значения: `On` и `Off` (или их альтернативные варианты `1` и `0`, `Yes` и `No`, `True` и `False`).

Время работы скрипта:

- `memory_limit` — максимальный объем памяти, выделяемый скрипту;
- `max_execution_time` — максимальное время работы скрипта, указывается в секундах, данный параметр можно установить в программе с помощью функции `set_time_limit(секунд)`;
- `max_input_time` — максимальное время, выделяемое сервером на прием `POST`-данных.

Вывод ошибок:

- `error_reporting` — с помощью данной директивы устанавливается уровень контроля ошибок в PHP, далее представлены основные варианты уровня контроля:
  - `E_ALL` — сообщать обо всех предупреждениях и ошибках (самый предпочтительный на этапе разработки и отладки программы);
  - `E_ERROR` — критические ошибки времени выполнения;
  - `E_WARNING` — предупреждения времени выполнения;
  - `E_NOTICE` — замечания времени выполнения (например, если в программе используется неинициализированная переменная);
  - `E_CORE_ERROR` — критические ошибки в момент старта PHP;
- `display_errors` — определяет, будет ли информация о предупреждениях и ошибках выводиться в браузер пользователю, возможны следующие варианты: `On` и `Off` (или их альтернативы `1` и `0`, `Yes` и `No`, `True` и `False`);
- `log_errors` — определяет, будут ли сообщения об ошибках и предупреждениях сохраняться в LOG-журнале, возможны следующие варианты: `On` и `Off` (или их альтернативы `1` и `0`, `Yes` и `No`, `True` и `False`), в случае использования Денвера файл LOG-журнала именуется error.log и располагается в том же каталоге, что и папка www для вашего сайта (рис. П1.1).

Если у вас есть доступ к php.ini, то вы просто можете открыть его и изменить ту или иную директиву. Но, как правило, на реальном хостинге доступ к этому файлу вам, конечно, закрыт. В этом случае для работы с настройками PHP предусмотрена специальная функция `ini_set()`, синтаксис которой следующий:

```
ini_set(имя_директивы, значение_директивы);
```

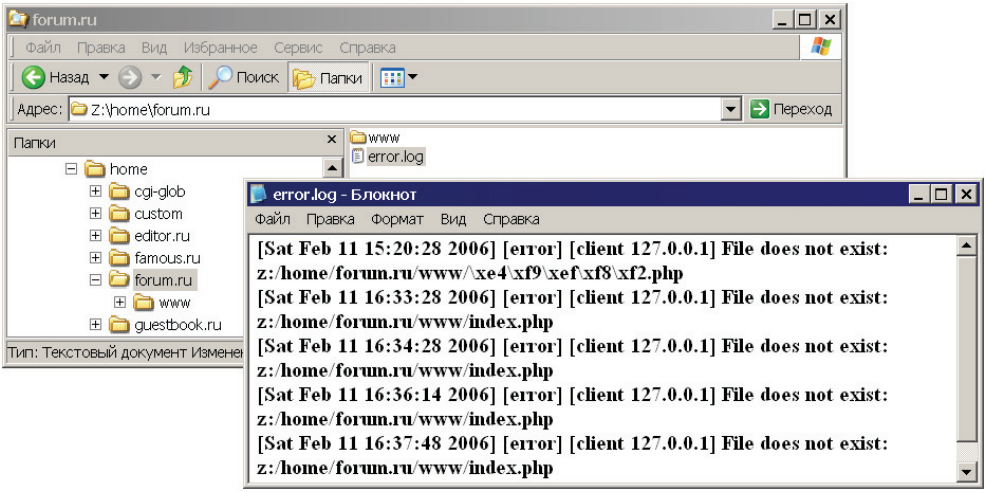


Рис. П1.1. LOG-журнал для форума, разработанного в *главе 10*, открытый в Блокноте

Рассмотрим следующую небольшую программу:

```
<?php
ini_set("max_execution_time","1");
sleep(5);
echo "Прошло 5 секунд";
?>
```

В результате ее выполнения вы никогда не увидите текст `"Прошло 5 секунд"`, т. к. с помощью `ini_set()` время выполнения скрипта ограничивается одной секундой и в связи с этим вам будет сообщено об ошибке (рис. П1.2).

А вот использование функции `ini_set()` для директивы `disable_function` не приведет ни к чему, т. к. эту директиву можно изменить только через php.ini. Более подробную информацию о том, какие директивы можно менять с помощью `ini_set()`, смотрите в официальной документации.

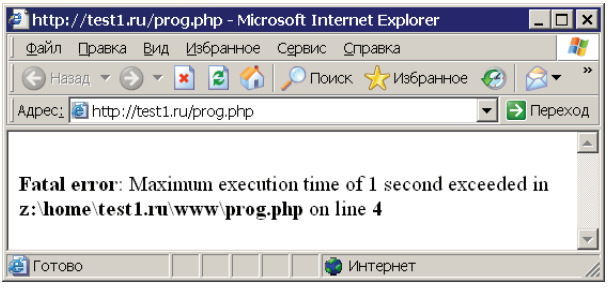


Рис. П1.2. Результат работы программы с ограниченным временем выполнения

Приложение 2. Список сайтов, связанных с PHP

В этом приложении даются полезные ссылки на ресурсы, связанные с PHP, Web-программированием и разработкой сайтов.

http://phpfaq.ru — FAQ по PHP.

http://sql.ru — здесь можно найти статьи по PHP и MySQL, а также отличный форум по PHP.

http://phpinside.ru — сайт редакции журнала PHPInside, здесь можно найти подборку статей, посвященных PHP.

http://www.phpbbguru.net — сайт посвящен популярному форуму phpBB.

http://webscript.ru —Web-программирование для всех (статьи, скрипты, форум).

http://hotscripts.com — библиотека самых разнообразных скриптов.

http://cmslist.ru — информация о различных системах управления сайтами (описания, сравнения и т. д.).

http://phpclub.ru — отличный сайт, посвященный PHP, на нем есть большая коллекция готовых PHP-скриптов и статей, хороший форум и электронный журнал для Web-разработчиков PHPInside, в котором можно найти много ценной информации.

http://www.php.net/links.php — по этой ссылке можно найти большое количество сайтов, посвященных PHP, начиная от готовых скриптов и заканчивая предложениями о работе.

http://www.devshed.com — на этом ресурсе можно найти статьи на английском языке, посвященные Web-программированию и тому, что с ним связано.

www.spravkaweb.ru — справочники и учебники по WEB.

http://greatweb.ru — на этом ресурсе интерес представляет раздел **Веб-мастеру**.

http://phpmanual.narod.ru — руководства, посвященные PHP.

http://xpoint.ru — форумы профессиональных Web-разработчиков.

http://phpconf.ru — официальный сайт PHP-конференции.

http://www.free-lancer.ru — вакансии по удаленной работе, а также коллекция скриптов.

http://woweb.ru — полезные вещи для Web-мастера (документация, скрипты, HTML-шаблоны и т. д.).

http://manlix.ru — статьи по Web-программированию, а также небольшая коллекция скриптов.

http://php.com.ua — общественный, некоммерческий проект независимой украинской группы PHP-разработчиков, созданный и поддерживаемый группой энтузиастов со всех регионов страны (здесь вы можете найти подборку статей по Web-программированию, коллекцию скриптов, на сайте есть форум).

http://www.softtime.ru — здесь вы можете найти учебник по PHP, подборку статей, а также посетить форум.

http://codenet.ru — все для программиста.

http://citforum.ru — море аналитической информации (среди нее можно найти неплохие статьи по PHP).

http://weblancer.net — интернет-биржа проектов.

Предметный указатель

A

Action 99
And 250, 252
Apache 29
Array 68
As 157
Auto_Increment 233

B

Bat 134
Break 206

C

Chr() 153
CMS, Content Management System 305
Cookie 184
Count() 168, 178
cPanel 340

D

Date() 149
Define() 65
DELETE 245
Die() 83

E

Echo 47
Edit 101
Exit 145
Explode() 213

F, G

FCKEditor 285
Fclose() 82
Feof() 140
Fgets() 81
File() 167
Filesize() 148
Firewall 35
Flock() 80, 81, 82
Fopen() 77, 78
Foreach 156
FORM 98
Fputs() 82
FROM 242
Fseek() 183
Ftell() 182
Ftruncate() 82
Gettype() 63

H

Header() 143
HTML-комментарии 46
Htmlspecialchars() 146
HTTP sniffer 124
HTTP-протокол 124

I

IDE 17
ieHTTPHeaders 121
In_Array() 211
Ini_Set() 350

Input 102
INSERT 243

M, N

Mail() 129
Mambo 305
Md5() 160
Method 100
Microsoft IIS 35
Microtime() 212
Move_uploaded_file() 211
MySQL 29, 223
Mysql_Connect() 246
Mysql_Db() 246
Mysql_Error() 248
Mysql_Fetch_Row() 254, 256
Mysql_Insert_Id() 281
Mysql_Num_Rows() 256
Mysql_query() 252, 254
Notepad++ 25

O

Or 83
ORDER BY 242
Outlook 134

P

Password 112
Pathinfo() 211
Perl 33
PHP 5, 29
PHP Designer 21
PHP Expert Editor 17, 18
PHP как внешняя программа 35
PHP как модуль
Web-сервера 35
Php.exe 10
Php.ini 41
PHP/FI 5
phpBB 297
phpMyAdmin 30, 224
POST-форма 111
Primary Key 234
Print_R() 69, 71, 202

R

Random() 219
Rename() 196
Require_once() 85

S, T

SELECT 242
Session_Destroy() 173
Session_Start() 168, 170
Set_Time_Limit() 350
Setcookie() 184, 185
SQL 222
SQL injection 251
SQL-запрос 229
Str_Replace() 152
Submit 103
Substr() 154
Switch() 205, 262
Time() 184

U

UPDATE 245
Upload_Max_Filesize 208
Upload_tmp_dir 208

W, X

Web-форма 98
Web-сервер 29
WHERE 242, 243
While 139, 256
XML-файл 93

А

Админка 160
Алгоритм MD5 160
Альтернативная запись if 146
Аналог таблицы 223
Атака типа SQL injection 251

Б

База данных 221, 223
Бесплатный хостинг 335, 338
Бинарный режим работы с файлом 77
Блог 317
Блокировка файла 80
Блокнот 17

В

Варианты блокировки файла 81
Виртуальный диск 33, 37
Вложенные формы 101
Вставка кода в HTML-тег 165
Входные данные 63
Выбор базы данных 246
Вывод данных в одну строку 57
Вывод символа $ 55
Выпадающий список 115
Выражение 206

Г

Главное правило организации цикла 139

Д

Данные формы 99
Два режима работы с файлом 77
Двумерный массив $_FILES 199
Денвер 30
Дескриптор файла 77
Директивы 349
Документация по cPanel 340
Документация по PHP 6

З

Заголовок Set-Cookie 187
Загрузка файла 198
Задание пути установки Денвера 33
Закачка сайта 338
Закрывающий тег ?>, 47
Закрытие файла 82
Запись 223
  данных в файл 82
Запрос 222
Запуск Денвера 35
Защита от лишних <Enter> 153
Защитный механизм 83
Знак ! 140
Значение константы 66
Значения массива 68

И

Изменение скрытого поля 120
Имя формы 99
Инициализация сессии 168, 170
Инструменты Денвера 37
Интерфейс гостевой 136
Информер 91—93, 95
Исключительная блокировка 81
Использование:
  GET POST одновременно 111
  двух видов кавычек 133
  констант 206
  точки 57

К

Как работает PHP 7
Ключ 68
Кнопка 103
  в виде картинки 104
Кодинг 137
Команда echo 47
Комментарии на PHP 47
Константа 65
  FALSE 67
  PHP_VERSION 67
  TRUE 67
Конфигурационные файлы 40
Кроссплатформенный язык 6

Л

Лог-журнал 73
Логическая конструкция 83
Логический оператор 83, 250, 252
Логическое выражение 83, 250

М

Массив 68
  $_COOKIE 185
  $_FILES 199
  $_SERVER 71
  $_SESSION 170
  $HTTP_POST_FILES 208
Метод:
  GET 109
  POST 110

Н

Назначение формы 97, 101
Накрутка голосования 183
Настройка Apache, PHP и MySQL 40
Неудачный пример написания программы 53

О

Обмен HTTP-заголовками 124
Обозначение переменной 54
Обращение к элементу массива 69
Объединение 57
  обработчика 131
  переменной 55, 56
  формы 98
Ограничение сохраняемой информации 154
Основные директивы 349
Основные понятия базы данных 223
Основные принципы работы с Mambo 311
Остановка Денвера 37
Открывающий тег <?php 47
Отладочная заглушка 133
Отличительная черта функции 62
Отправка формы 99
Отправка электронного письма 129
Отрицание в выражении сравнения 178
Отрицание не 140
Официальный сайт:
  cPanel 340
  FCKEditor 285
  Mambo 306
  MySQL 223
  phpBB 297
  phpMyAdmin 224
  русского Mambo 306
Ошибка 404 8

П

Пакет 199
Панель администратора 160
Папка tmp 171
Первичный ключ 234
Перевод строки 148
Перевод строки в Windows 152
Перезапуск Денвера 37
Переменная 54
Переменные окружения 71
Перенаправление 143, 144
Подключение:
  к MySQL 246
  модуля 85
  своего сайта к системе статистики 89
Подробное описание HTTP 1.1 124
Подстановка недостающего символа 153
Поле 223
  ввода 101
Получение данных формы 108
Понятие:
  категории в Mambo 311
  компонента в Mambo 312
  модуля 86
  модуля в Mambo 312
  объекта в Mambo 311
  раздела в Mambo 311
  секции в Mambo 311
  шаблона в Mambo 313

Построение SQL-запроса 242
Правильное название
для переменной 60
Предопределенные константы 67
Преимущество использования БД 222
Премодерация 155
Префикс БД 341
Пример SQL injection 251
Принцип оценки быстродействия 212
Принцип работы echo 50
Присвоение переменной значения 55
Проверка данных 132
Просмотр скрытой информации
для браузеров 121
Просмотр скрытой информации
через WEB 122
Просмотр статистики по сайту 88
Простое определение массива 68
Простые типы данных 55

P

Работа:
препроцессора 14
с двумерным массивом 199
с меню в Mambo 319
с платным хостингом 340
функции mail() 133
Разделяемая блокировка 81
Размещение информера 92
Разработка счетчика 76
Разработка функции 214
Распространенная ошибка
начинающих 151
Расширение PHP-программ 11
Регистрозависимый язык 59
Регулярные выражения 95
Режим запуска Денвера 34
Режим работы с файлом 78

C

Самые распространенные
скрипты 127
Сердце fckeditor 293
Сессия 168
Символ 148, 152
& 109
? 109
\n 148
Система:
статистики по сайту 88
управления базой данных 223
управления контентом 305
Скин 303
Скрытое поле MAX_FILE_SIZE 203
Смена дизайна phpBB 303
Снятие блокировки 81, 82
Создание:
checkbox 114
переключателя 113
базы данных в cPanel 341
баннера в Mambo 322
БД 229
виртуального сайта 40
голосования в Mambo 325
категории в Mambo 316
кнопки 103
константы 65
массива 68
объекта в Mambo 317
переключателя 113
пользователя MySQL в cPanel 341
поля ввода многострочного
текста 118
поля для ввода пароля 112
пустого массива 68
раздела в Mambo 314
скрытого поля 119
списка 115
статичного материала в Mambo 320
таблицы 231
флага 114
формы 98
функции 214
Сокращенная запись создания
формы 100
Состав Денвера 39
Специальный элемент
типа file 198, 199

Спецификация 124
Список с возможностью
множественного выбора 115
Список с фиксированным числом
видимых элементов 115
Способ:
отправки формы 100
работы с файлом 78
установки PHP 35
Стандартные счетчики 88
Стандартные функции 63
Столбец 221, 222
Строка 221
Структура заголовка 144
СУБД 29, 223
Счетчик 75

Т

Таблица 223
Текстовый режим работы с файлом 77
Тело:
запроса 110
функции 214
цикла 139
Тестирование программы 152
Тип данных 55
resource 78
для столбца 231
Точка с запятой 48

У

Удаление данных из файла 82
Удаление переменной 55, 61
Удобства метода GET 109
Указание на родительский
каталог 167
Указатель на результат 252, 254
Указатель текущей позиции 81, 140
Уничтожение сессии 173
Упрощенная схема работы
с файлом 78
Условная конструкция 129
Установка:
Mambo 306
компонента для Mambo 330
модуля для Mambo 329
ограничения на размер
закачиваемого файла 203
шаблона для Mambo 334
Утилиты Денвера 37
Учебник по HTML 7

Ф

Форма 97, 98
для загрузки файла 198
обратной связи 127
Форум phpBB 297
Функция 62

Х

Хеш 160

Ц

Цензура 155
Цикл 139
while 256
с предусловием 139

Ч

Чтение строки из файла 81

Э

Элемент return 214

Я

Язык SQL 222
Ярлык:
Restart Servers 37
Start Servers 35
Stop Servers 37
Ячейка 221, 222

PHP — ЭТО ПРОСТО

ПРОГРАММИРУЕМ ДЛЯ WEB-САЙТА

Прочтите книгу, у вас все получится!

Шкрыль Андрей Александрович,
профессиональный программист, имеющий большой практический опыт в разработке Web-проектов, принимал участие в создании крупных биллинговых систем. Автор книги «Разработка клиент-серверных приложений в Delphi».

- Что нужно знать для успешной работы с PHP?
- Почему вместе с PHP чаще всего употребляют слова MySQL и APACHE?
- Чем PHP лучше для создания сайтов по сравнению с другими языками программирования?
- Что такое CMS и как создать свой сайт всего за несколько минут, не вникая в тонкости Web-программирования?
- Какой комплект программного обеспечения необходим для нормальной разработки Web-сайтов?
- Как использовать общедоступные интернет-сервисы, например, разместить на своем сайте прогноз погоды и курс валют?
- Как закачать свой сайт на реальный хостинг?

На все эти вопросы и не только на них вы найдете ответы в данной книге. Изложение ведется с позиции начинающего программиста. Каждый шаг сопровождается практическими примерами, полезными советами, а также многочисленными схемами и рисунками, что позволит читателю быстро и легко освоить тонкости, связанные с Web-программированием. Книга будет полезна не только начинающим, но и всем тем, кто хочет повысить уровень своих знаний в области создания интерактивных Web-сайтов.

«…На сегодняшний день действительно не существует книги, которая бы охватывала несколько готовых популярных программных решений, а так же освещала работу с бесплатными хостингами …»

М. В. Кузнецов, автор бестселлеров «Самоучитель PHP 5», «PHP 5. Практика разработки Web-сайтов», «PHP 5 на примерах» и др.

ISBN 5-94157-905-5

9 785941 579051

БХВ-ПЕТЕРБУРГ
194354,
ул. Есенина, 5Б
E-mail: mail@bhv.ru
internet: www.bhv.ru
тел./факс: (812) 591-6243