Строганов А.С.

PHP

Ваш первый сайт с использованием PHP-скриптов

ДИАЛОГ-МИФИ

А. С. Строганов

ВАШ ПЕРВЫЙ САЙТ С ИСПОЛЬЗОВАНИЕМ PHP-СКРИПТОВ

МОСКВА • ДИАЛОГ-МИФИ • 2008

УДК 519.682
ББК 32.97
С86

Строганов А. С.
С86 Ваш первый сайт с использованием PHP-скриптов. – М.: Издательство Диалог-МИФИ, 2008. – 288 с.

ISBN 978-5-86404-226-7

Вы решили создать свой сайт, но не знаете с чего начать? Сделать это поможет данная книга. Вы познакомитесь с PHP-программированием, научитесь легко и быстро наполнить свой сайт тысячами страниц, создав при этом вручную всего одну.

Книга написана простым и понятным языком, который доступен даже новичку, никогда не слышавшему о PHP. Освоив материал данной книги, вы сможете создавать сайты любой сложности, писать любые PHP-скрипты для любых целей, используя всю вашу фантазию.

Учебное пособие
Строганов Александр Сергеевич
Ваш первый сайт с использованием PHP-скриптов

Редактор О. А. Голубев
Макет И. М. Чумаковоой

Подписано в печать 15.07.2008.
Формат 60x84/16. Бум. офс. Печать офс. Гарнитура Таймс.
Усл. печ. л. 16,74. Уч.-изд. л. 11,64. Тираж 1 000 экз. Заказ

ООО "Издательство ДИАЛОГ-МИФИ"
115409, Москва, ул. Москворечье, 31, корп. 2. Т.: (495) 320-30-77, 320-43-77
Http://www.dialog-mifi.ru. E-mail:zakaz@dialog-mifi.ru
ООО "ИНСОФТ"
117105, г. Москва, Варшавское ш., д. 37А

ISBN 978-5-86404-226-7

ВВЕДЕНИЕ

Предметом изучения в данной книге является универсальный язык скриптов PHP. Материалы издания будут полезны не только новичкам. Мы с вами построим сайт, используя язык PHP. Самое главное – это понять принципы программирования на PHP и использовать это для создания сайта любой сложности.

Данная книга имеет следующую структуру. В гл. 1 рассматривается общее представление и информация о PHP. В гл. 2 мы установим и сконфигурируем программное обеспечение на компьютер для программирования и тестирования созданных программ-сценариев на PHP. В гл. 3 мы в краткой форме рассмотрим язык разметки HTML для придания хотя бы минимального дизайна страницам сайта. Создадим главную страницу сайта. В гл. 4 мы приступим непосредственно к изучению основ PHP. Вообще изучать PHP мы будем практически в каждой главе и сразу же будем применять полученные знания на практике. Уже в четвертой главе мы создадим счетчик посещений для сайта. В гл. 5 построим страницу для новостей сайта. Вы научитесь отправлять и обрабатывать данные из HTML форм. В гл. 6 создадим страничку сайта с подбором статей, позволим посетителям сайта оценивать этот сайт и оставлять на нем свои отзывы. В гл. 7 научимся создавать простую гостевую книгу на сайте. В гл. 8 рассматривается работа с графическими изображениями. Мы составим страницу для размещения на ней фотографий и других изображений. Дадим возможность посетителям вашего сайта закачивать на страницу свои фотографии или рисунки. В гл. 9 создадим простой интернет-магазин, где посетители могут выбрать себе товар и отправить бланк заказа к вам на электронный адрес. В гл. 10 мы познакомимся с основами электронной коммерции в Интернете. Вы узнаете, что такое электронные деньги, как создавать себе электронные кошельки и пополнять их деньгами. На практике создадим себе несколько электронных кошельков. Создадим страничку для продажи электронных товаров, например, mp3-файлов. В гл. 11 осуществим на практике продажу электронных товаров со своего сайта, используя сервис *Web Merchant Interface* или *Roboxchange*.

Все PHP-скрипты подробно разобраны и понятны даже новичку. Разъяснена практически каждая строчка программного кода. Главная цель книги – научиться писать PHP-скрипты и создавать собственные приложения, научиться создавать динамические PHP-страницы для своего сайта и, в конце концов, научиться зарабатывать, используя свои страницы сайта.

Глава 1. САЙТОСТРОЕНИЕ И PHP

1.1. ЗНАКОМСТВО С PHP

Последние десять лет ознаменовались фантастическим развитием Интернета и новых способов общения между людьми. На переднем крае этого явления находится World Wide Web (WWW). Ежедневно в этой новой коммуникационной среде открываются тысячи новых сайтов, а потребителям предлагаются новые виды услуг. Вместе с бурным развитием рынка появился огромный спрос на новые технологии и разработчиков, владеющих ими. Если вы читаете этот абзац, вероятно, вы уже являетесь или скоро станете web-разработчиком. Впрочем, какой бы ни была ваша профессия, вы выбрали эту книгу потому, что слышали о замечательной новой технологии – PHP. В этой главе вы познакомитесь с языком PHP, получите представление о его истории и возможностях, а также основную информацию, необходимую для разработки сайтов с поддержкой PHP. Надеюсь, приведенные примеры пробудят ваш энтузиазм и наглядно покажут, какие перспективы PHP открывает перед вами и вашей организацией. В гл. 2 вы узнаете, как установить и настроить программное обеспечение PHP на компьютерах с Windows.

Что такое PHP? Если вы только начинаете знакомиться с PHP, то вам нужно знать определения. Итак, что же такое PHP? PHP – это широко используемый язык сценариев общего назначения с открытым исходным кодом. Говоря проще, PHP – это язык программирования, специально разработанный для написания web-приложений (сценариев), исполняющихся на web-сервере.

Аббревиатура PHP означает «Hypertext Preprocessor (Препроцессор Гипертекста)». PHP достаточно прост для изучения. Преимуществом PHP является предоставление web-разработчикам возможности быстрого создания динамически генерируемых web-страниц. Важным преимуществом языка PHP перед такими языками, как языки Perl и C заключается в возможности создания HTML-документов с внедренными командами PHP. Значительным отличием PHP от какого-либо кода, выполняющегося на стороне клиента, например, JavaScript, является то, что PHP-скрипты выполняются на стороне сервера. Вы даже можете сконфигурировать свой сервер таким образом, чтобы HTML-файлы обрабатывались процессором PHP, так что клиенты даже не смогут узнать, получают ли они обычный HTML-файл или результат выполнения скрипта.

PHP позволяет создавать качественные web-приложения за очень короткие сроки, получая продукты, легко модифицируемые и поддерживаемые в будущем. PHP прост для освоения и вместе с тем способен удовлетворить запросы профессиональных программистов. Даже если вы впервые услыша-

ли о PHP, изучить этот язык не составит для вас большого труда. Я не сомневаюсь, что, изучив основы PHP в течение нескольких часов, вы уже сможете создавать простые PHP-скрипты. Язык PHP постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков web-программирования, по крайней мере, в ближайшее время.

Возможности PHP очень большие. Главным образом, область применения PHP сфокусирована на написание скриптов, работающих на стороне сервера. Таким образом, PHP способен выполнять все то, что выполняет любая другая программа. Например, обрабатывать данные формы, генерировать динамические страницы. PHP способен выполнять и множество других задач.

Как работает PHP, где он выполняется? PHP выполняется, как я уже говорил, на сервере. Браузер посылает серверу запрос на страницу с PHP-кодом. Сервер отдает эту страницу на исполнение интерпретатору PHP, интерпретатор генерирует HTML-код, отдает серверу, а сервер посылает клиенту. Никакого PHP-кода в браузер не попадает. Это важно! Это значит, что увидеть исходный код PHP-скрипта невозможно! Единственный способ отправить что-то скрипту – это кликнуть по ссылке или нажать на кнопку в форме. Так, чтобы PHP обрабатывал какие-то действия пользователя в браузере – невозможно. PHP остался на сервере, ждать новых запросов с данными для обработки. PHP, но не скрипт! Скрипт, который выполнялся, отдавая пользователю страницу, завершил работу. Все данные, которые были в нем – пропали. Именно поэтому, если какая-то переменная нужна при последующих вызовах скрипта, ее надо этому скрипту передать снова.

1.2. КРАТКАЯ ИСТОРИЯ

История PHP начинается с 1995 г., когда независимый программист-контрактник по имени Расмус Лердорф (Rasmus Lerdorf) написал сценарий Perl/CGI для подсчета количества посетителей сайта, прочитавших его онлайновое резюме. Его сценарий решал две задачи: регистрацию данных посетителя и вывод количества посетителей на web-странице. Развитие WWW еще только начиналось, никаких специальных средств для решения этих задач не было, и к автору хлынул поток сообщений с вопросами. Лердорф начал бесплатно раздавать свой инструментарий, названный Personal Home Page (PHP) или Hypertext Processor (гипертекстовый процессор). Шумный успех инструментария PHP заставил Лердорфа приступить к разработке расширений PHP. Одно из расширений преобразовывало данные, введенные на форме HTML, в символические переменные, что позволяло экспортировать их в другие системы. Чтобы добиться поставленной цели, Лердорф решил в дальнейших разработках перейти с Perl на C. Расширение существующего инструментария PHP привело к появлению PHP 2.0, или PHP-FI (Personal Home Page – Form Interpretator). В усовершенствовании версии 2.0 принима-

ли участие программисты со всего мира. Новая версия PHP пользовалась исключительной популярностью, и вскоре образовалась основная команда разработчиков. Они сохранили исходную концепцию внедрения программного кода прямо в HTML и переписали заново механизм лексического анализа, что привело к появлению PHP 3.0. К моменту выхода версии 3.0 в 1997 г. свыше 50 000 пользователей применяли PHP для улучшения своих web-страниц. В 1997 г. было решено, что сокращение PHP должно означать не «Personal Home page», а «PHP Hypertext Processor».

В течение следующих двух лет стремительное развитие PHP продолжалось. В язык добавлялись сотни новых функций, а количество пользователей стремительно росло. В начале 1999 г. служба Netcraft (http://www.netcraft.com) сообщила о том, что, по минимальным оценкам, число пользователей PHP превысило 1 000 000, в результате чего PHP стал одним из самых популярных сценарных языков в мире.

В начале 1999 г. было объявлено о предстоящем выходе PHP 4.0. Хотя одной из сильнейших сторон PHP была эффективность выполнения сценариев, при первоначальных разработках не предполагалось, что на базе PHP будут строиться крупномасштабные приложения. По этой причине была начата работа над более устойчивым механизмом лексического анализа, больше известным под названием Zend (http://www.zend.com). Работа шла быстро и завершилась 22 мая 2000 г. выпуском PHP версии 4.0. В настоящее время, все более популярной становится PHP версии 5.0. Программное обеспечение PHP установлено более чем в 4 млн доменов. Будущее PHP выглядит светлым, поскольку продукт продолжает активно использоваться как на крупных web-сайтах, так и на компьютерах отдельных пользователей.

PHP лучше всего охарактеризовать как работающий на стороне сервера встроенный язык сценариев Web, позволяющий разработчикам быстро и эффективно строить динамические web-приложения. С позиций грамматики и синтаксиса PHP напоминает язык программирования C, хотя разработчики не постеснялись включить в него средства из других языков, в том числе из Perl, Java и C++. Среди ценных заимствованных возможностей – поддержка регулярных выражений, мощные средства работы с массивами, объектно-ориентированная методология и обширная поддержка работы с базами данных.

При написании приложений, выходящих за рамки традиционной, статической методологии разработки web-страниц (т. е. HTML), PHP также может послужить ценным инструментом для создания и управления динамическим содержанием, который используется наряду с JavaScript, стилями, WML (Wireless Markup Language) и другими полезными языками. Благодаря наличию сотен стандартных функций PHP в состоянии решить практически любую задачу, которая может прийти в голову разработчику. В нем имеется обширная поддержка создания графики и операций с ней, математических вычислений, средств электронной коммерции и таких популярных техноло-

гий, как XML (Extensible Markup Language), ODBC (Open Database Connectivity) и Macromedia Shockwave. Широкий выбор возможностей избавляет от необходимости рутинной и непростой работы по подключению сторонних модулей, поэтому многие разработчики со всего мира останавливают свой выбор на PHP.

Одним из главных достоинств PHP является тот факт, что он внедряется прямо в HTML-код, поэтому программисту не приходится писать программу с множеством команд для простого вывода HTML. Код HTML и PHP можно чередовать по мере необходимости. PHP позволяет написать фрагмент следующего вида:

```
<html>
<title><? print "Hello world!"; ?></title>
</html>
```

Сообщение *"Hello world!"* выводится в заголовке web-страницы. Интересно то, что команда *print* внутри конструкции, которая обычно называется экранирующими последовательностями PHP (<?...?>), представляет собой законченную программу. Ни длинного кода инициализации, ни включения библиотек – программа состоит лишь из того кода, который непосредственно решает поставленную задачу! Если на данный момент вы не понимаете приведенный выше код, то поймете позже, дойдя до гл. 4.

Конечно, для выполнения сценариев PHP необходимо предварительно установить и настроить программное обеспечение PHP на сервере (или на вашем компьютере). Этот процесс описан в гл. 2. Но прежде чем браться за процесс установки, мы познакомимся с некоторыми характеристиками PHP. Этой теме посвящен следующий раздел данной главы.

1.3. ХАРАКТЕРИСТИКИ PHP

Как вы, вероятно, уже поняли, главным фактором при проектировании языка PHP является практичность. PHP должен предоставить программисту средства для быстрого и эффективного решения поставленных задач. Практический характер PHP обусловлен пятью важными характеристиками:

- традиционностью;
- простотой;
- эффективностью;
- безопасностью;
- гибкостью.

Существует еще одна «характеристика», которая делает PHP особенно привлекательным: он распространяется бесплатно!

1.3.1. Традиционность

Язык PHP кажется знакомым программистам, работающим в разных областях. Многие конструкции языка позаимствованы из Си Perl, а нередко код PHP практически неотличим от того, что встречается в типичных программах C или Pascal. Это заметно снижает начальные усилия при изучении PHP.

1.3.2. Простота

Сценарий PHP может состоять из 10 000 строк или из одной строки – все зависит от специфики вашей задачи. Вам не придется подгружать библиотеки, указывать специальные параметры компиляции или что-нибудь в этом роде. Механизм PHP просто начинает выполнять код после первой экранирующей последовательности или тега <? и продолжает выполнение до того момента, когда он встретит парную экранирующую последовательность ?>. Если код имеет правильный синтаксис, он исполняется в точности так, как указал программист.

1.3.3. Эффективность

Эффективность является исключительно важным фактором при программировании для многопользовательских сред, к числу которых относится и WWW. В PHP 4.0 был реализован механизм выделения ресурсов и обеспечена улучшенная поддержка объектно-ориентированного программирования, а также средства управления сеансом. В последней версии появился и механизм подсчета ссылок (reference counting), предотвращающий выделение лишней памяти.

1.3.4. Безопасность

PHP предоставляет в распоряжение разработчиков и администраторов гибкие и эффективные средства безопасности. В PHP реализованы механизмы безопасности, находящиеся под управлением администраторов. При правильной настройке PHP это обеспечивает максимальную свободу действий и безопасность. PHP может работать в так называемом безопасном режиме (safe mode), который ограничивает возможности применения PHP пользователями по ряду важных показателей. Например, можно ограничить максимальное время выполнения и использование памяти (неконтролируемый расход памяти отрицательно влияет на быстродействие сервера). Администратор также может устанавливать ограничения на каталоги, в которых пользователь может просматривать и исполнять сценарии PHP, а также использовать сценарии PHP для просмотра конфиденциальной информации на сервере.

В стандартный набор функций PHP входит ряд надежных механизмов шифрования. Исходный текст сценариев PHP, например, нельзя просмотреть в браузере, поскольку сценарий компилируется до его отправки по запросу пользователя. Реализация PHP на стороне сервера предотвращает похищение нетривиальных сценариев пользователями.

1.3.5. Гибкость

Поскольку PHP является встраиваемым (embedded) языком, он отличается исключительной гибкостью по отношению к потребностям разработчика. Хотя PHP обычно рекомендуется использовать в сочетании с HTML, он с таким же успехом интегрируется и в JavaScript, WML, XML и другие языки. Кроме того, хорошо структурированные приложения PHP легко расширяются по мере необходимости (впрочем, это относится ко всем основным языкам программирования).

Нет проблем и с зависимостью от браузеров, поскольку перед отправкой клиенту сценарии PHP полностью компилируются на стороне сервера. В сущности, сценарии PHP могут передаваться любым устройствам с браузерами, включая сотовые телефоны, электронные записные книжки, пейджеры и портативные компьютеры, не говоря уже о традиционных PC. Программисты, занимающиеся вспомогательными утилитами, могут запускать PHP в режиме командной строки.

Поскольку PHP не содержит кода, ориентированного на конкретный web-сервер, пользователи не ограничиваются определенными серверами (возможно, незнакомыми для них). Apache, Microsoft IIS, Netscape Enterprise Server, Stronghold и Zeus – PHP работает на всех перечисленных серверах. Поскольку эти серверы работают на разных платформах, PHP в целом является платформенно-независимым языком и существует на таких платформах, как UNIX, Solaris, FreeBSD и Windows 95/98/NT.

Наконец, средства PHP позволяют программисту работать с внешними компонентами, такими как Enterprise Java Beans или COM-объекты Win32. Благодаря этим новым возможностям PHP занимает достойное место среди современных технологий и обеспечивает масштабирование проектов до необходимых пределов.

1.3.6. Бесплатное распространение

Стратегия Open Source наделала немало шуму в программной отрасли. Распространение исходных текстов программ в массах оказало несомненно благотворное влияние на многие проекты, в первую очередь – Linux, хотя и успех проекта Apache сильно подкрепил позиции сторонников Open Source. Сказанное относится и к истории создания PHP, поскольку поддержка пользователей со всего мира оказалась очень важным фактором в развитии проекта PHP.

Принятие стратегии Open Source и бесплатное распространение исходных текстов PHP оказало неоценимую услугу пользователям. Вдобавок отзывчивое сообщество пользователей PHP является своего рода «коллективной службой поддержки», и в популярных электронных конференциях можно найти ответы даже на самые сложные вопросы.

1.4. СТРОИТЕЛЬСТВО САЙТА НА PHP

Итак, вы решили создать свой сайт. Вы знаете, что такое HTML-страница. С помощью HTML-кода можно оформить страницу на свой вкус, не прибегая к PHP-коду. Бродя по просторам Интернета, вы сталкивались с сайтами, имеющими огромный контент (фотографии, картинки, музыку, программы и т. п.) и содержащие огромное число страниц. Конечно, если сайт небольшой, состоящий из 1–2 страниц, то составить такой сайт, используя только HTML-код, несложно. Но если вы захотите разместить на сайте, например, 1000 фотографий, то реализация такой задачи становится очень долгой и сложной. Ну, во-первых, все фотографии на одной HTML-странице вы размещать не будете, иначе данная страница будет открываться в браузере посетителя очень и очень долго. Посетитель просто покинет вашу страницу, не дождавшись загрузки. Вам придется делать сотню HTML-страниц под фотографии, на каждой странице писать ссылки к каждой фотографии. Не забудьте, кстати, сделать мини-копию каждой фотографии (не будете же вы сразу выводить полные оригиналы фотографий в браузер). Это съест кучу вашего времени, а при написании тысячи ссылок вы наверняка наделаете кучу опечаток, и эти ссылки просто не будут срабатывать. Используя только HTML-код, вы не сможете создать на своем сайте, например, форум, динамически меняющийся блок новостей или простой счетчик посещения страниц. Сайт будет выглядеть скучным и неинтересным. Любое добавление какого-либо контента на сайт превратится в нудную и рутинную работу по написанию дополнительных ссылок. Несмотря на это, прежде чем рассматривать программирование на PHP, мы вкратце рассмотрим HTML (см. гл. 3). Даже если весь ваш сайт будет состоять только из PHP-страниц, обойтись в сайтостроении без HTML-кода невозможно.

Используя язык PHP, вы можете создать сайт любой сложности! Ваш сайт может состоять хоть из нескольких тысяч страниц, причем создавать нужно будет всего одну страницу. Вы будете, например, только закачивать фотографии на сервер – и все. Программа, составленная на PHP, сама будет делать мини-копии этих фото, размещать ссылки на оригиналы этих фотографий на страницах, а сами новые страницы будут создаваться автоматически. Если, например, число фотографий на странице превысит 10, то одиннадцатая фотография разместится на новой странице, которую вам не нужно будет самому создавать. При помощи PHP вы сможете создавать на страницах своего сайта форумы, гостевые книги, блоки новостей, счетчики посещений, организовать опросы посетителей и многое другое. Ваш сайт станет как бы «живым» и динамическим, что, естественно, привлечет на него больше посетителей. Используя PHP, можно создать в Интернете целый портал с огромным числом страниц, управлять которым будет легко и интересно.

Для тестирования созданных PHP-страниц нужен сервер. Любой сайт в Интернете располагается на каком-либо сервере. Можно, конечно, создать сначала первую страницу своего сайта и разместить ее сразу на сервере и протестировать на работоспособность, затем создать вторую страницу и также сразу разместить ее на сервере и протестировать. Это не совсем удобно, поэтому вы создадите сначала все страницы сайта на своем компьютере, протестируйте их и только потом закачаете весь созданный вами сайт на выбранный вами сервер. Однако протестировать PHP-страницы на своем компьютере вы не сможете без нужного программного обеспечения (без сервера). Вы должны создать на своем компьютере свой сервер, чтобы тестировать созданные вами страницы с PHP-кодом. В гл. 2 рассказано, какие программы нам для этого понадобятся, как их установить и настроить для работы.

Глава 2. УСТАНОВКА И КОНФИГУРИРОВАНИЕ ПРОГРАММ ДЛЯ ПРОГРАММИРОВАНИЯ НА PHP

2.1. УСТАНОВКА ПРОГРАММ

Итак, для того чтобы вы могли изучать программирование на PHP, нужно установить на своем компьютере домашний сервер, точнее, серверные программы, с помощью которых можно создавать и тестировать PHP программы. Все эти программы вы можете найти в Интернете и скачать. Например, в поисковой системе yandex (*www.yandex.ru*) в строке поиска наберите название нужной вам программы. Вы найдете множество сайтов, где эти программы можно свободно скачать. Адреса этих сайтов я укажу ниже. В качестве операционной системы можно использовать Windows, желательно Windows XP или Windows 2000 SP4. Систему Windows 95 уже использовать нельзя, так как PHP5 в ней работать не будет!

Первая программа, которая нам будет нужна, **Apache 2.0.54 Win32**. Можно использовать ранние версии (v2.0.50) или, наоборот, более свежие. Используйте то, что сможете достать в Интернете или на программных дисках. Мы же здесь будем работать с версией 2.0.54 Win32. Официальная страница сайта для скачивания такого программного продукта *http://httpd.apache.org/download.cgi*.

Устанавливается программа следующим образом:

1. Запустите файл установки *apache_2.0.54-win32-x86-no_ssl* и следуйте инструкциям. При установке, параметры сервера можете вводить любые, например, как на рис. 2.1. Вместо *home.com* и *www.home.com* лучше просто написать *localhost*, поскольку сервер будет находиться на локальной машине, т. е. на нашем компьютере.
2. Далее, на следующем шаге выберите обычный тип установки (*Typical*). Затем вы должны выбрать директорию, где будут находиться файлы этой программы. Пусть все программы сервера у вас будут храниться на диске C в папке *server*, а данная программа – в папке *Apache*. Тогда путь к директории будет, например, как на рис. 2.2.
3. После выбора директории начнется установка программы, после которой нажмите кнопку *Finish* для завершения этой установки.
4. Далее произойдет автоматический запуск программы. В правом нижнем углу появится иконка в виде зеленого треугольника. Запустите Интернет-браузер на вашем компьютере (не подключая к Интернету) и в адресной строке браузера наберите строчку *http://localhost* и нажмите *Enter*. Если установка прошла успешно, то вы увидите страницу, показанную на рис. 2.3.

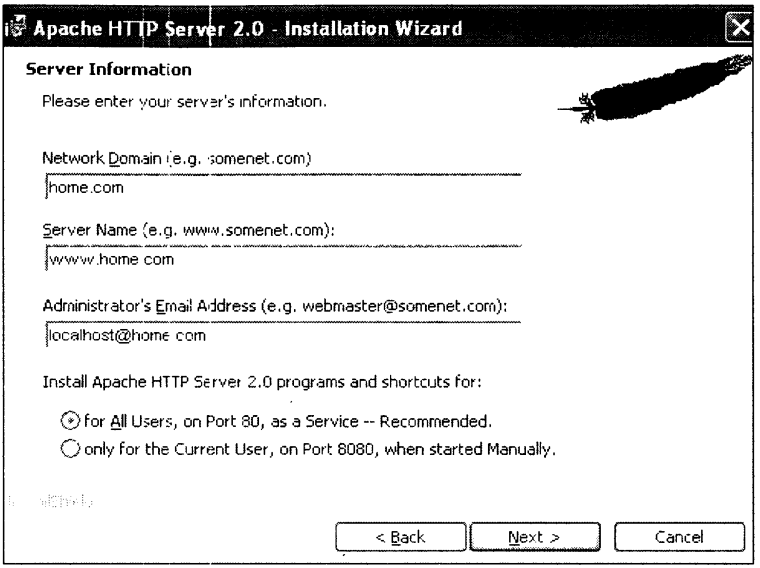

Apache HTTP Server 2.0 - Installation Wizard
Server Information
Please enter your server's information.
Network Domain (e.g. somenet.com)
home.com
Server Name (e.g. www.somenet.com):
www.home.com
Administrator's Email Address (e.g. webmaster@somenet.com):
localhost@home.com
Install Apache HTTP Server 2.0 programs and shortcuts for:
for All Users, on Port 80, as a Service -- Recommended.
only for the Current User, on Port 8080, when started Manually.
< Back
Next >
Cancel

Рис. 2.1


Apache HTTP Server 2.0 - Installation Wizard
Destination Folder
Click Change to install to a different folder
Install Apache HTTP Server 2.0 to the folder:
C:\server\Apache\
Change...
< Back
Next >
Cancel

Рис. 2.2

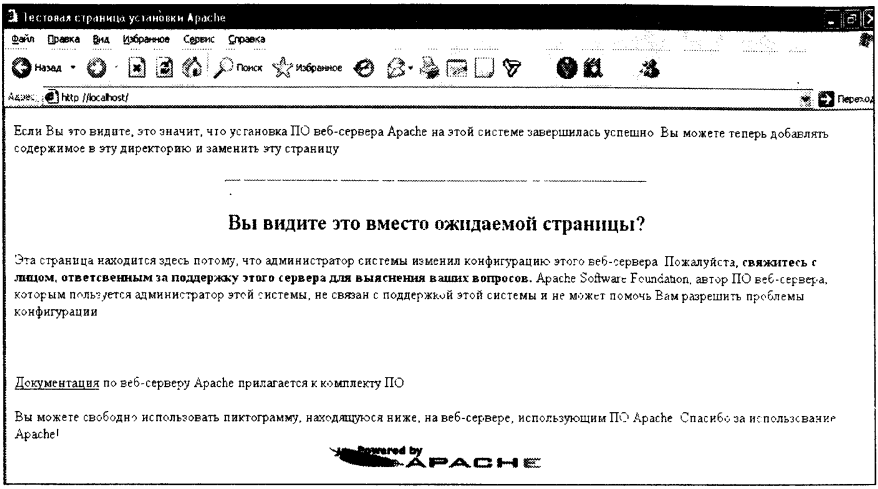


Рис. 2.3

Следующую программу, которую мы будем использовать, – это **php v5.0.5**. Она позволит нам запускать PHP-скрипты на нашем домашнем сервере. Можно использовать другую версию, например *php v5.0.0*. Официальная страница сайта для скачивания этой программы *http://www.php.net/downloads.php*.

Сделайте следующее:

1. Распакуйте файл php-5.0.5-Win32.zip в папку C:/server/PHP5/.
2. Найдите в этой папке файл php.ini-dist. Переименуйте его в php.ini и скопируйте в папку, где у вас установлена операционная система windows. В нашем случае, для Windows XP, это C:/WINDOWS.
3. Найдите файл libmysql.dll в папке C:/server/PHP5/ и скопируйте его в папку C:/WINDOWS/system32.

Пока все. Чуть позже мы сделаем некоторые настройки этой программы.

Следующую программу для установки я рекомендую **MySQL v4.0.26**. Эта программа для работы с базами данных. Есть другие версии этой программы, например *MySQL v4.0.20d*. Разницы между ними практически никакой. Официальная страница сайта для скачивания этой программы *http://www.mysql.com/downloads/index.html*. После того как скачаете, сделайте следующее:

1. Распакуйте файл *mysql-4.0.26-win32*. Запустите файл *setup.exe*.

2. При установке, в качестве директории, куда будет устанавливаться программа, выберите *C:/server/mysql* и нажмите кнопку *next*.
3. В следующем шаге тип установки выберите *Typical*. По окончании установки, нажмите кнопку *Finish*.

Вообще мы составим сайт без использования баз данных, но в учебных целях попробуйте поработать с базами данных самостоятельно, на будущее.

И, наконец, установим PHP-редактор, в котором было бы удобно составлять PHP-скрипты и просто web-страницы. Я рекомендую программу *PHP Edit v5.6 Rus*, хотя можно использовать и другие. Официальный сайт программы *http://phpedit.com.ua*.

Эту программу можно найти и на других сайтах. Она распространяется свободно.

Для установки программы запустите файл *phpedit.exe*. По умолчанию директория установки будет *C:\Program Files\Svoi.NET\PHP Edit* (рис. 2.4). Можете изменить директорию на свое усмотрение, например, на *C:/server/phpedit*.

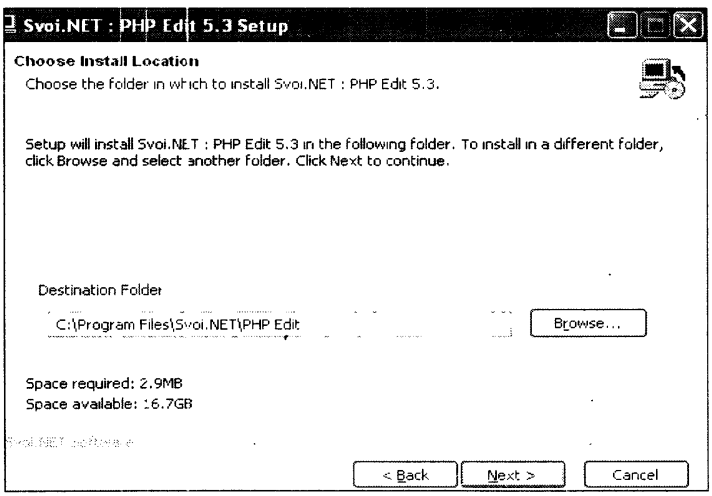


Рис. 2.4

После установки запустите программу. Если главное меню будет на английском языке, то выберите русский, выбрав предпоследний пункт главного меню (*Language*). Теперь настроим работу установленного PHP-редактора. Выберите в главном меню «*Пуск*», далее «*Настройки запуска скриптов*» (рис. 2.5).

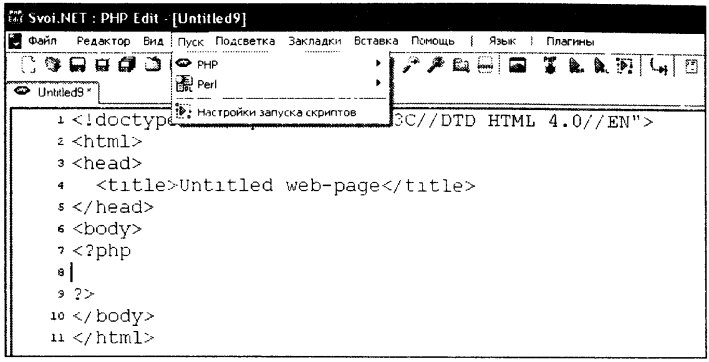


Рис. 2.5

В настройках вам надо указать путь к файлу *php.exe* (рис. 2.6). У нас он находится в папке *C:/server/PHP5/* (если вы, конечно, установили программу *php v5.0.5* в папку *PHP5*). Следовательно, в строке путь к PHP пишем *C:\server\PHP5\php.exe*. Обратите внимание, что слеш используется обратный «\». Далее в разделе «*Опции запуска*», на всех опциях поставьте флажки. Поставьте флажок на «*Использовать внешний веб-сервер (Apache, MS IIs, др.)*», поскольку мы и будем использовать «внешний» домашний сервер *Apache*.

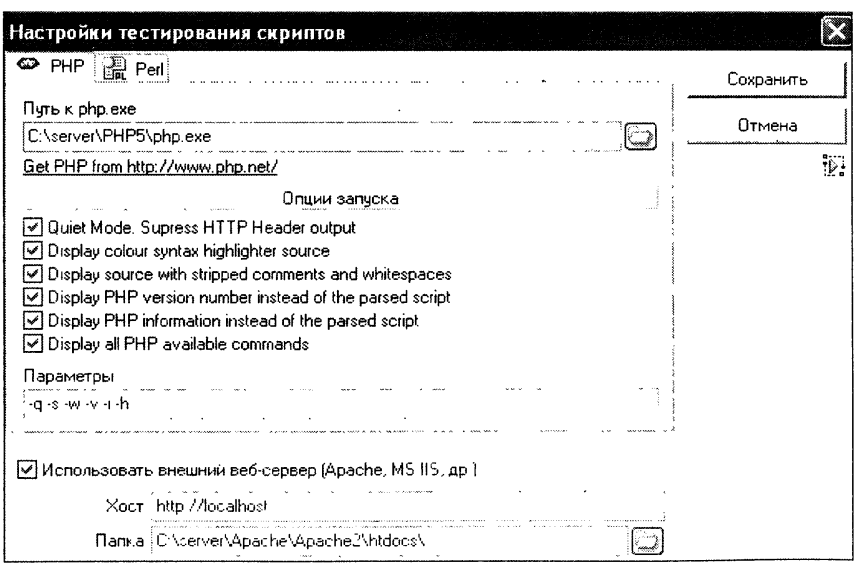


Рис. 2.6

В качестве хоста укажите *http://localhost*. Внизу, в строчке «*Папка*», нужно указать путь к папке *htdocs*. Эта папка будет у нас главной рабочей папкой нашего сервера. Именно в этой папке будут расположены все папки и файлы нашего сайта. На моем компьютере путь к этой папке выглядит так: *C:\server\Apache\Apache2\htdocs* (рис. 2.6).

Если путь к рабочей папке иной, например, *C:\server\Apache2\htdocs*, то так и укажите в строке «*Папка*».

Итак, мы установили все программы, которые нам понадобятся для работы. Однако PHP-скрипты запускаться у вас пока не будут, так как мы еще не настроили работу программ *Apache 2.0.54* и *php v5.0.5*. Этим мы сейчас и займемся.

2.2. НАСТРОЙКА ПРОГРАММ

2.2.1. Настройка Apache

1. В папке *Apache2* откройте вложенную там папку *conf*. Найдите там файл *httpd.conf* и откройте его.
2. Найдите там параметр *DocumentRoot*. Для быстрого поиска поставьте курсор в начало данного текстового документа. В пункте «*правка*» главного меню нажмите на пункт «*найти*». Откроется окно поиска (рис. 2.7), куда и надо записать слово, которое мы ищем (в строчку «*Что*») и нажать кнопку «*Найти далее*».

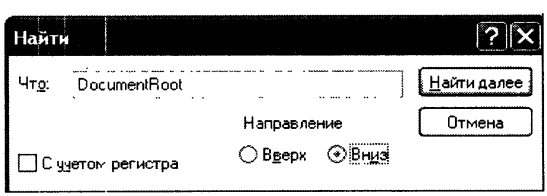


Рис. 2.7

В этом параметре, *DocumentRoot*, должен быть указан путь к рабочей папке *htdocs*, т. е. иметь вид: *DocumentRoot "C:/server/Apache/Apache2/htdocs"*. Если путь к рабочей папке у вас другой, то исправьте этот параметр.

3. Далее найдите строчки, содержащие слово *LoadModule*. Впишите в этом блоке последней строкой следующее:
LoadModule php5_module "c:/server/php5/php5apache2.dll".
У меня, как видите, файл *php5apache2.dll* расположен в папке *php5* (или *PHP5*). Если путь к файлу *php5apache2.dll* у вас другой, то измените эту строчку.

4. Далее найдите строки, содержащие *AddType*. Добавьте еще одну строку: *AddType application/x-httpd-php.php*. Мы указываем серверу расширение файлов, где находится PHP-код. Перед этой строкой не должно быть символа комментария #.
5. Сохраните изменения в файле *httpd.conf* и закройте его.

2.2.2. Настройка PHP

1. Раньше, если помните, мы скопировали файл *php.ini* из папки *C:/server/PHP5/* в папку *C:/WINDOWS/*. Именно этот файл нам и нужен. Зайдите в папку *C:/WINDOWS/* и откройте файл *php.ini*. В нем содержаться параметры для работы с PHP. Найдите в этом файле параметр *doc_root* и укажите рядом путь (через знак «=») к нашей рабочей папке *htdocs*, т. е. у вас должна получиться строчка:
   doc_root="C:\server\Apache\Apache2\htdocs"
   Обратите внимание, что здесь используется обратный слеш «\». Если путь к рабочей папке у вас другой, то измените этот параметр.
2. Найдите параметр *extension_dir* и измените его так:
   extension_dir="C:\server\PHP5\ext".
   В начале строки не должно быть точки с запятой. Этот символ, так же как и символ #, указывает на комментарий, т. е. то, что следует в строке за этими знаками, программа воспринимает как комментарий и игнорирует строку.
3. Найдите строчку *extension=php_mysql.dll* и уберите в ее начале точку с запятой.
4. Сохраните изменения в файле *php.ini* и закройте его.

После сохранения перезапустите сервер. В правом нижнем углу нажмите левой кнопкой мыши на иконке в виде зеленого треугольника и выберите из появившегося меню пункт «*Restart*». Вообще после любых внесенных вами изменений в файле *php.ini* нужно обязательно перезапускать сервер программы *Apache HTTP Server*.

Итак, мы настроили нужные нам программы. Давайте создадим какой-нибудь PHP-файл и протестируем его. Откройте редактор *PHP Edit* (рис. 2.8). В основном меню выберите пункт «*Создать*». Далее в подменю выберите пункт «*PHP*».

Откроется пустой PHP-документ, точнее, это HTML-документ, в который можно вставить PHP-код между тегами *<?php* и *?>* (рис. 2.9).

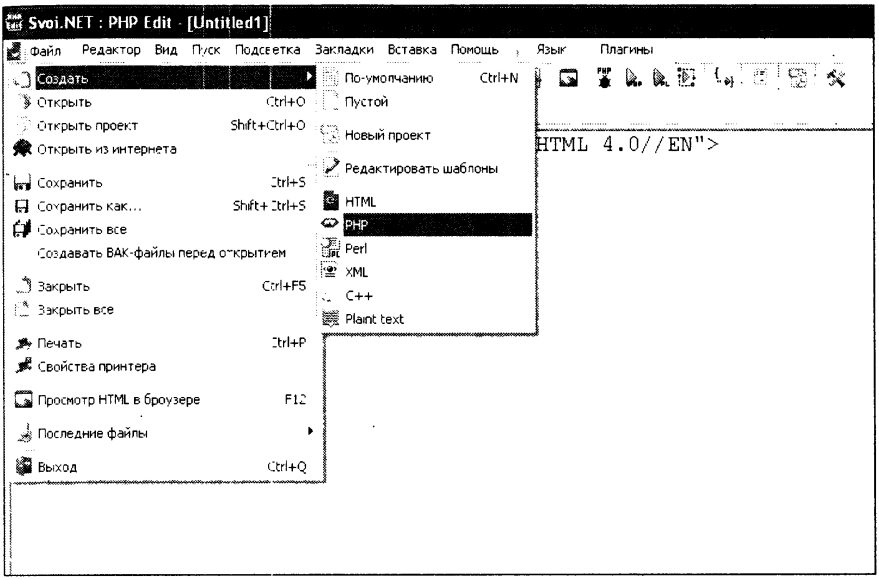


Рис. 2.8

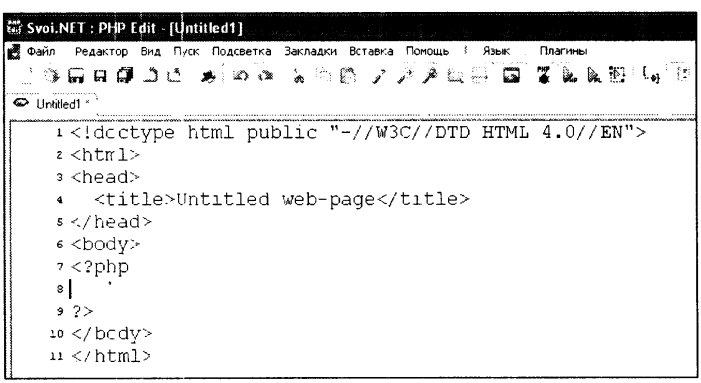


Рис. 2.9

После тега <*body*> напишите строчку: «*Это строка в HTML-коде*».

После тега <*?php* напишите: *echo "Это PHP-код";*.

В результате, когда вы это сделаете, код в PHP-редакторе должен выглядеть так, как на рис. 2.10.

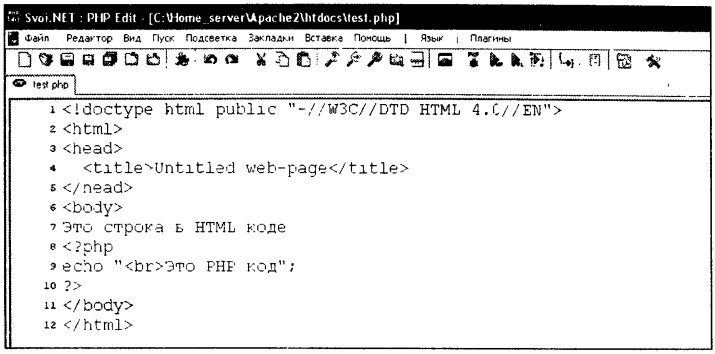


Рис. 2.10

Сохраните этот файл в рабочей папке *htdocs* нашего локального сервера под названием *test.php* (именно с расширением php!). Все созданные нами папки и файлы мы будем сохранять именно в основной рабочей папке *htdocs*, которая у меня, как я уже указал выше, находится по адресу: *C:/server/Apache/Apache2/htdocs*. Также файлы и папки сайта можно сохранять и в других папках, но которые вложены в папку *htdocs*. Файлы с расширением php, сохраненные «выше» папки *htdocs*, например в папке *server* или *Apache*, запускаться в PHP-редакторе не будут!

Запустите браузер, нажав на зеленый треугольник с надписью PHP. Он находится слева, под главным меню PHP-редактора. Если вы все сделали правильно, то в окне браузера вы увидите две написанные вами строчки. В адресной строке будет адрес: *http://localhost/test.php*.

Если у вас что-то не получилось, перечитайте внимательно эту главу. Может, вы что-то сделали не так, как написано. Правила установки данных программ можно найти и в Интернете.

Глава 3. ОСНОВЫ HTML, СОЗДАНИЕ ГЛАВНОЙ СТРАНИЦЫ САЙТА

3.1. НАЧИНАЕМ СОЗДАВАТЬ ГЛАВНУЮ СТРАНИЦУ САЙТА

Прежде чем приступить к строительству нашего сайта и изучению PHP, немного повторим основы языка HTML (повторим вкратце, поскольку, я надеюсь, вы язык HTML хоть немного знаете). Это будет несложно. В этой главе мы рассмотрим только основные понятия HTML и PHP. В последующих главах, по мере надобности, мы рассмотрим множество других возможностей данных языков в сайтостроении. Для начала в PHP-редакторе создайте новый PHP-документ, как написано в конце предыдущей главы. Самая верхняя строка (то, что заключено между знаками «<» и «>»):

<!doctype html public "-//W3C//DTD HTML 4.0//EN"> – объявляет формат и тип содержимого документа. Он указывает на тип документа, соответствующий стандарту HTML 4.0 и ориентированный на англоязычные документы. Вообще современные WEB-браузеры обходятся без этого тега, поэтому мы и не будем заострять на нем внимание.

Парный тег в начале *<html>* и *</html>* в конце документа указывает, собственно, на начало и конец данного HTML-документа. Объяснения здесь излишни.

Далее идет парный тег заголовка документа *<head>* и *</head>*. Между ними мы видим другой парный тег *<title>* и *</title>*. Текст, написанный между этой парой, будет отображаться в заголовке окна браузера, при загрузке данного документа. Это может быть, например, название вашей WEB-страницы.

В парном теге *<body>* и *</body>* отображается основная часть или тело документа. Это могут быть созданные вами текст, картинки, кнопки, баннеры и т. д.. Теги *<?php* и *?>*, между которыми заключается PHP-код, разберем позже. Уберем их пока из документа, чтобы они нам не мешали (рис. 3.1).

Сохраним документ под названием *index.php*. Обратите внимание, чтобы этот файл имел расширение именно *php*, поскольку потом мы будем вставлять сюда PHP-код (по умолчанию, редактором будет предложено сохранить файл с расширением *html*). PHP-код нам здесь потом понадобится для создания счетчика посещаемости на этой странице. Если вы сохраните этот файл с расширением *html*, то использовать в этом документе *PHP*-код будет уже нельзя!

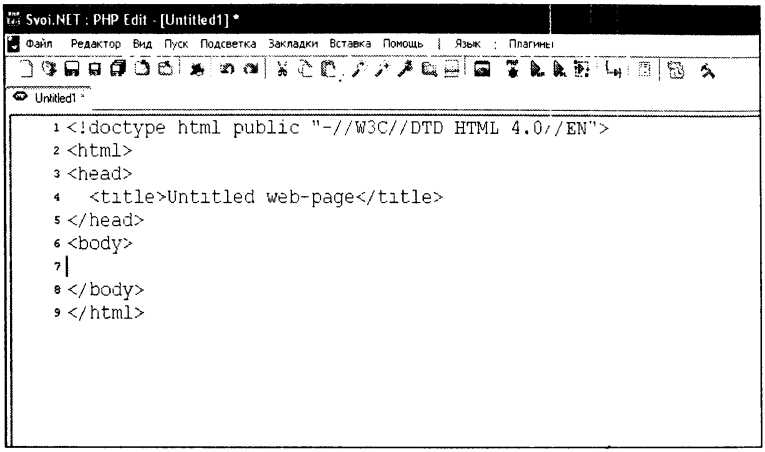


Рис. 3.1

Итак, перед нами «скелет» главной страницы нашего сайта. Теперь мы начнем шаг за шагом наращивать «мышечную массу» этой страницы.

Для начала напишите между тегами *<body>* и *</body>* фразу: *Добро пожаловать на мой сайт*. Запустите браузер (нажмите на изображение маленького зеленого треугольника с надписью *PHP* под основным меню редактора). В левом верхнем углу появится написанная вами фраза (рис. 3.2). Конечно, посетитель, зайдя на эту страницу, долго здесь не задержится, поскольку ничего привлекательного здесь еще нет. Поэтому в этой главе мы рассмотрим некоторые основные HTML-теги для украшения нашей главной страницы.

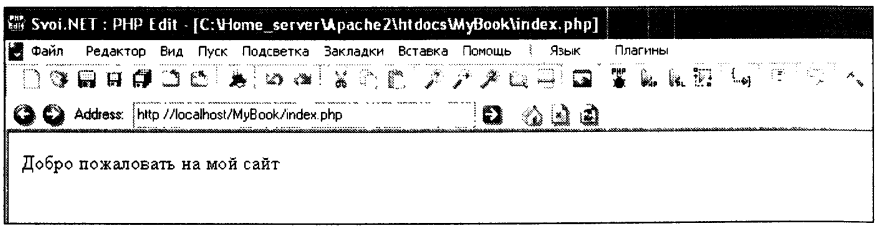


Рис. 3.2

Теги *<h1>...<h6>* и, соответственно, закрывающие их теги *</h1>...</h6>* указывают на размер шрифта. Чем больше цифра после буквы *h*, тем мельче шрифт. Теги могут содержать в себе дополнительные элементы или атрибуты с определенными значениями. Например, элемент *align* указывает на расположение объекта на странице по горизонтали. Он может принимать значения: *center* – по центру, *left* – слева, *right* – справа. В нашем случае объектом является написанный нами текст. Мы его можем заключить в теги, например,

расположить его между *<h2>* и *</h2>*, а чтобы этот текст располагался по центру страницы, в теге *<h2>* укажем атрибут *align* со значением *center*, т. е.:

<h2 align="center"> Добро пожаловать на мой сайт</h2>.

Закрывающийся тег *</h2>* обязателен, так как он указывает на то, что форматирование применено нами только к этому тексту. В теге *<body>* можно тоже указывать атрибуты, но они будут применены ко всему документу, т. е. к тому, что будет располагаться между <body> и </body>. Значение элемента *bgcolor* задает цвет фона. Например, запись *<body bgcolor="silver">* задает светло-серый фоновый цвет всего документа. Или можно записать так: *<body bgcolor="#c0c0c0">*. Здесь цвет указан в виде шестнадцатеричного кода.

В табл. 3.1 указаны основные цвета и их шестнадцатеричный код. Вы можете установить любой цвет фона на ваш вкус. Вместо однотонного цвета, на задний фон можно поместить картинку. Делается это при помощи атрибута *background*, значение которого – путь к данной картинке, например:

<body background="fon.gif">.

В этом случае на заднем фоне страницы будет изображено содержимое файла *fon.gif*, который должен быть расположен в той же папке нашего сайта, где находится главная страница *index.php* (в корневой папке нашего сервера *htdocs*). Если картинка расположена в другой папке, например в папке *image*, которая, естественно, должна быть вложена в главную папку сайта *htdocs*, то нужно записать так:

<body background="image/fon.gif">.

Вообще использовать картинку в качестве фона нецелесообразно, так как некоторые web-посетители отключают загрузку картинок в браузере для экономии времени, и поэтому вместо картинки они увидят белый фон. В качестве фона страницы лучше все-таки задавать цвет.

Таблица 3.1

Цвет	***Название***	***Код цвета***	***Цвет***	***Название***	***Код цвета***
Белый	white	ffffff	**Розовый**	fuchsia	ff00ff
Бирюзовый	agua	00ffff	**Светло-серый**	silver	c0c0c0
Голубой	teal	008080	**Синий**	blue	0000ff
Желтый	yellow	ffff00	**Сиреневый**	purple	800080
Зеленый	lime	00ff00	**Темно-зеленый**	green	008000
Красный	red	ff0000	**Темно-серый**	gray	808080
Малиновый	maroon	800000	**Темно-синий**	havy	000080
Оливковый	olive	808000	**Черный**	black	000000

Между тегами <*h2*> и </*h2*> можно вложить теги, задающие начертания шрифта:

- <*b*>...</*b*> – текст будет полужирным;
- <*I*>...</*I*> – текст будет задан курсивом;
- <*U*>...<*U*> – текст будет подчеркнут.

Я специально написал буквы в разных регистрах, так как не имеет значения, как вы напишите <*b*> или <*B*>. Результат будет один и тот же.

Итак, запишем в редакторе следующий код (рис. 3.3). Между тегами <*title*> и </*title*> пишем заголовок: «*Главная страница*». В теге <*body*> задаем цвет фона *silver*. Наш объект, т. е. текст «*Добро пожаловать на мой сайт*» заключаем в тег <*h2 align="center"*>, который задает размер шрифта текста и помещает его в центре страницы. Внутри тегов <*h2*> и </*h2*> вставляем теги начертания шрифта, а именно, задающие курсивный текст <*I*> и </*I*>.

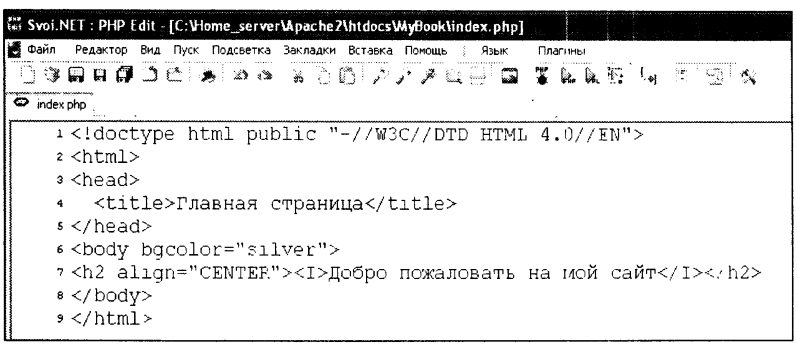


Рис. 3.3

Сохраните страницу нажатием на картинку, изображающую одиночную дискетку (которая побольше) под главным меню PHP-редактора. Запустите браузер (зеленый треугольник с надписью *PHP* под основным меню). Результат показан на рис. 3.4.

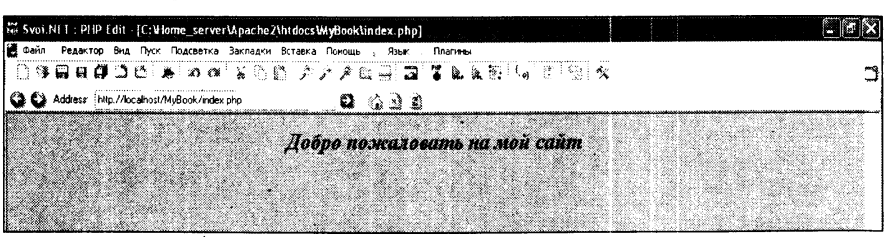


Рис. 3.4

Запомните правило размещения одних тегов внутри других: **Тег, который открывается раньше других, закрывается позже других.** В нашем случае сначала открывается тег *<h2>*, а затем *<I>*. После текста нужно закрыть сначала тег *</I>*, а потом *</h2>*. То есть тег *<I>* открылся внутри *<h2>*, следовательно, и закрываться *</I>* должен внутри *</h2>*.

3.2. КОНТЕЙНЕР МЕЖДУ <DIV> И </DIV>

Теги * * и * * используются для выделения какого-либо раздела, который можно разместить в любом месте страницы. Разместим, например, между ними текст (листинг 3.1):

«*Это учебный сайт. Он позволит быстро и наглядно изучить основные принципы программирования на PHP. Вы научитесь составлять скрипты для сайта любой сложности. Эти скрипты помогут вам автоматизировать добавление разнообразного материала на ваш сайт, вести подсчет посетителей вашего сайта, узнать мнение посетителей о вашем сайте при помощи гостевой книги или системы оценки материалов*».

Сохранив страницу и запустив браузер, вы увидите там набранный текст. Он будет располагаться ниже фразы «*Добро пожаловать на мой сайт*» и занимать всю ширину браузера.

Листинг 3.1 (файл index.php)

```
<html>
<head>
<title>Главная страница</title>
</head>
<body bgcolor="silver">
<h2 align="CENTER"><I>Добро пожаловать на мой сайт</I></h2>
<div>Это учебный сайт. Он позволит быстро и наглядно изучить
основные принципы программирования на PHP. Вы научитесь
составлять скрипты для сайта любой сложности. Эти скрипты
помогут вам автоматизировать добавление разнообразного материала на
ваш сайт, вести подсчет посетителей вашего сайта, узнать мнение посетителей
о вашем сайте при помощи гостевой книги или системы оценки материалов</div>
</body>
</html>
```

Давайте теперь сделаем так, чтобы контейнер * * занимал бы не более 25 % окна браузера, а текст был бы выровнен по всей ширине этого контейнера. Здесь нам помогут листы стилей.

3.3. ЛИСТЫ СТИЛЕЙ

С помощью листов стилей можно форматировать текст, определять его шрифт, цвет, размер и многое другое. Задается стиль с помощью ключевого слова *style*. Для какого-либо отдельного элемента стиль определяется так:

style="свойство:значение; свойство:значение; ...свойство:значение".

Style имеет свойства и значения этих свойств. Рассмотрим некоторые из этих свойств:

color определяет цвет текста, а его значения – названия этих цветов или их шестнадцатеричные коды. Свойство от значения отделяется двоеточием (*color:yellow* или *color:#ffff00*). Пары *свойства:значение* отделяются друг от друга точкой с запятой;

font-size задает размер шрифта, например *font-size:12pt* задает шрифт размером в 12 питов;

font-family задает название шрифта, например *font-family:serif*;

font-style задает вид шрифта, например *font-style:italic* задает наклонный шрифт, а *font-style:bold* – жирный шрифт;

background задает цвет фона, на котором будет находиться текст или другой объект. Например, *background: peachpuff* задает светло-оранжевый цвет фона элемента;

width задает ширину текста. Ее можно задать как в единицах длины, так и в процентах от общей ширины браузера, например *width:50%*;

text-align задает выравнивание текста, например *text-align:left* выравнивает текст по левому краю, *text-align:right* – по правому краю, *text-align:justify* – по ширине.

Другие свойства будут рассмотрены далее, в процессе строения сайта.

Зададим стиль для текста в контейнере * *, как показано в листинге 3.2.

Листинг 3.2 (файл index.php)

```
<html>
<head>
<title>Главная страница</title>
</head>
<body bgcolor="silver">
<h2 align="CENTER"><I>Добро пожаловать на мой сайт</I></h2>
<div style="color:red; font-family:Comic Sans MS; width:25%;
font-size:14pt ; text-align:justify ">
```

Это учебный сайт. Он позволит быстро и наглядно изучить основные принципы программирования на PHP. Вы научитесь составлять скрипты для сайта любой сложности. Эти скрипты помогут вам автоматизировать добавление разнообразного материала на ваш сайт, вести подсчет посетителей вашего сайта, узнать мнение посетителей о вашем сайте при помощи гостевой книги или системы оценки материалов
** **
</body>
</html>

Сохраните страницу и запустите браузер. Набранный текст отображен в виде столбика (контейнера) слева, занимающего 25 % от общей ширины браузера, так как мы задали свойству *width* значение *25 %*. Цвет текста (свойство *color*) задан красным, размер шрифта *14pt*, название шрифта задано *Comic Sans MS*. Если вы предпочитаете другой шрифт, например стандартный Times New Roman, то можете установить его. Свойству text-align (выравнивание текста) я присвоил одно из значений justify (выравнивание по ширине). Если вы поставите другое значение, например left, то текст будет выровнен по левому краю.

Теперь представьте, что у нас несколько текстовых разделов, каждый из которых заключен между парными тегами * * и * *. Причем формат, размер, цвет текста мы хотим оставить такими же. Нам нужно будет каждый раз расписывать стиль для каждого тега. Чтобы этого не делать, применим специальный контейнер или парный тег *<style>*. Он вставляется в заголовочную часть документа, например после закрывающего тега *</title>*, но до закрывающего тега *</head>*. Посмотрите на листинг 3.3.

Листинг 3.3 (файл index.php)

```
<html>
<head>
<title>Главная страница</title>
<style type="text/css">
    div {color:red; font-family:Comic Sans MS; font-size:14pt; text-align:justify}
    body {background:silver}
    </style>
</head>

<body>
<h2 align="CENTER"><I>Добро пожаловать на мой сайт</I></h2>

<div style="width:25%">
Это учебный сайт. Он позволит быстро и наглядно изучить
основные принципы программирования на PHP. Вы научитесь
составлять скрипты для сайта любой сложности. Эти скрипты
```

помогут вам автоматизировать добавление разнообразного материала на ваш сайт, вести подсчет посетителей вашего сайта, узнать мнение посетителей о вашем сайте при помощи гостевой книги или системы оценки материалов
** **

</body>
</html>

В теге *<style type="text/css">* значение *text/css* атрибута *type* сообщает браузеру, что применяется текст на языке *CSS*. *CSS* – набор правил, которые задают форматирование документа. Далее дается указание браузеру, что все элементы * * должны оформляться так, как указано в фигурных скобках. Как вы заметили далее, в основной части документа (после тега *<body>*), в теге * * в элементе *style* мы уже не указываем многих свойств, как в листинге 3.2, так как эти свойства уже указаны в головной части документа, в контейнере *<style> </style>*. Ширину же контейнера * * задаем индивидуально – *width:25%*, хотя, если вы хотите, чтобы все текстовые разделы в контейнерах * * имели ширину в 25 % от ширины браузера, то и это свойство элемента *style* можно перенести в фигурную скобку.

Также мы указали браузеру, что для элемента *<body>* фоновый цвет документа *silver*. Обратите внимание, что задний фон в этом случае определяет свойство *background*.

Наш сайт будет состоять из многих страниц. Если вы хотите, чтобы все страницы сайта были оформлены однотипно, то удобнее установить стиль сразу для всех страниц, т. е. создать один внешний файл листа стилей. В папке, где у вас находится наша главная страница *index.php*, создайте текстовый файл и вставьте в него строки:

div {color:red; font-family:Comic Sans MS; font-size:14pt; text-align:justify}
body {background:silver}

Мы записали то, что находится у нас между тегами *<style type="text/css">* и *</style>* (листинг 3.3). Сохраните этот файл как *stil.css*. Расширение *css* показывает, что созданный файл является именно файлом внешнего листа стилей. Теперь удалите из кода страницы *index.php* (листинг 3.3) контейнер *<style type="text/css">...</style>* и все, что в нем находится. Вместо этого запишите *<link type="text/css" rel="stylesheet" href="stil.css">*, и теперь у вас код будет выглядеть так, как в листинге 3.4.

Листинг 3.4 (файл index.php)

```
<html>
<head>
<title>Главная страница</title>

<link type="text/css" rel="stylesheet" href="stil.css">

</head>
```

```
<body>
<h2 align="CENTER"><I>Добро пожаловать на мой сайт</I></h2>

<div style="width:25%">
Это учебный сайт. Он позволит быстро и наглядно изучить
основные принципы программирования на PHP. Вы научитесь
составлять скрипты для сайта любой сложности. Эти скрипты
помогут вам автоматизировать добавление разнообразного материала
на ваш сайт, вести подсчет посетителей вашего сайта, узнать мнение
посетителей о вашем сайте при помощи гостевой книги или системы
оценки материалов
</div>

</body>
</html>
```

При помощи одиночного тега *<link>* внешний лист стилей (файл *stil.css*) подключается к нашему файлу *index.php*. Указанные атрибуты этого тега являются обязательными:

type="text/css" – указываем браузеру, что применяем текст формата *CSS*.

href="stil.css" – путь к файлу внешнего листа стилей. У нас этот файл находится в том же каталоге *hdocs*, что и *index.php*. Если каталог другой (например, вы решили сохранить лист стилей в подкаталоге *stili* каталога *hdocs*), то необходимо его указать (*href="stili/stil.css"*).

rel="stylesheet" – указываем браузеру, что элемент *<link>* устанавливает связь с внешним листом стилей.

Теперь мы можем применить внешний лист стилей *stil.css* к любому числу страниц сайта. Достаточно в головную часть каждой страницы (между тегами *<head>* и *</head>*) вставить указанную выше строку: *<link type="text/css" rel="stylesheet" href="stil.css">*.

Несмотря на то, что мы подключили внешний файл стилей, для каждого элемента (тега для оформления текста) на странице *index.php* (да и на других страницах сайта) можно установить свой индивидуальный стиль. Если мы в основной части документа (после тега *<body>*) в тег * * добавим, например, цвет *yellow* (* *), то текст станет желтым. Это несмотря на то, что во внешнем листе стилей *stil.css* цвет текста в теге * * указан красным. Мы как бы «перебили» красный цвет желтым. Для каждого элемента, если это необходимо, можно задать свой индивидуальный стиль.

Итак, слева мы разместили текст, занимающий 25 % ширины браузера. Далее на 50% ширины разместим картинку, а на остальных 25 % будут располагаться ссылки на другие страницы нашего сайта.

3.4. ВСТАВКА КАРТИНКИ

Вставка картинки в документ осуществляется при помощи одиночного тега *<img>*. Скопируйте в папку, с которой мы работаем (в папку *htdocs*, где находится главная страница сайта *index.php*), какую-нибудь картинку с вашего компьютера. Я, например, скопировал в рабочую папку файл *salut.gif*, где изображен салют. Далее в коде после закрывающего тега * *, но до закрывающегося тега *</body>* вставим строчку:

<img src="salut.gif" width="50%">.

В данном теге мы указываем при помощи обязательного атрибута *src* название файла вставляемой картинки, а при помощи необязательного атрибута *width* указываем ширину картинки. Она должна занимать 50 % ширины браузера. Сохраним страницу и запустим браузер. Наша картинка расположилась внизу, после колонки с текстом. Но нам нужно, чтобы изображение находилось на уровне текстового блока. Сам текстовый блок желательно было бы расположить справа от изображения, а не слева. Слева от картинки на оставшиеся 25 % окна браузера можно было бы расположить наши ссылки на другие страницы, которые мы создадим потом. Как это сделать? Здесь нам поможет таблица.

3.5. ВСТАВКА ТАБЛИЦЫ

Любая таблица в HTML начинается тегом *<table>* и заканчивается *</table>*. Внутри этих тегов располагаются парные теги, формирующие строки и столбцы. После откравающего тега *<table>* следует парный тег *<tr>...</tr>*, формирующий строку таблицы. Внутри этих тегов могут располагаться парные теги *<td>...</td>*, формирующие ячейки в данной строке. Вот, собственно, вся теория. Например, код для таблицы с двумя строками и двумя столбцами (всего 4 ячейки) будет такой:

<table> – начало таблицы;

<tr><td>Содержимое первой ячейки</td> – начинаем первую строку и вставляем туда первую ячейку;

<td>Содержимое второй ячейки</td></tr> – вставляем вторую ячейку и заканчиваем первую строку;

<tr><td>Содержимое третьей ячейки</td> – начинаем вторую строку и вставляем туда третью ячейку;

<td>Содержимое четвертой ячейки</td></tr> – вставляем четвертую ячейку и заканчиваем вторую строку;

</table> – окончание таблицы.

Если вы вставите данные строчки кода между тегами *<body>* и *</body>*, то увидите в браузере таблицу (рис. 3.5).

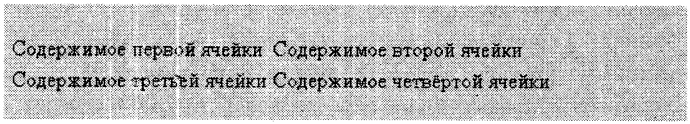


Рис. 3.5

Это та самая таблица, состоящая из двух строк и двух столбцов. Самой таблицы не видно, так как мы не задали толщину границ таблицы (по умолчанию она равна нулю).

В качестве содержимого ячеек таблицы может быть текст, картинка или несколько картинок и т.п.

В теге *<table>* можно задать атрибуты, например *width* – ширина таблицы, *align* – выравнивание таблицы относительно боковых границ окна браузера, *border* – задаст толщину границ таблицы в пикселях. Если параметр не указан, браузер создаст таблицу без рамок, что у нас и получилось на рис. 3.5, *bgcolor* – задает цвет фона таблицы. Если же воспользоваться таблицами стилей, то, например, красный цвет фона задается так: *style="background-color:red"*, *bordercolor* – задает цвет внешней рамки. Атрибуты можно задать не только для всей таблицы, но и для отдельной ячейки, в теге *<td>*, например, тег *<td valign="top">* выравнивает текст по высоте только в данной ячейке, а тег *<td align="left">* выравнивает текст по ширине (по левую сторону) в данной ячейке. Существует много других атрибутов, которые мы использовать не будем, но если вы хотите построить сайт высокого дизайна, то советую почитать дополнительную литературу по HTML.

Итак, поместим наш текст и рисунок в ячейки однострочной таблицы (листинг 3.5).

Листинг 3.5 (файл index.php)

```
<html>
<head>
<title>Главная страница</title>
<link type="text/css" rel="stylesheet" href="stil.css">
</head>
<body>
<h2 align="CENTER"><I>Добро пожаловать на мой сайт</I></h2>
<table width="100%"><tr>
<td width="25%"></td>
<td width="50%" valign="top">
<img src="salut.gif"></td>
```

```
<td width="25%">
<div>
Это учебный сайт. Он позволит быстро и наглядно изучить
основные принципы программирования на PHP. Вы научитесь
составлять скрипты для сайта любой сложности. Эти скрипты
помогут вам автоматизировать добавление разнообразного материала
на ваш сайт, вести подсчет посетителей вашего сайта, узнать мнение
посетителей о вашем сайте при помощи гостевой книги или системы
оценки материалов
</div></td>
</tr></table>
</body>
</html>
```

Разберем некоторые строчки кода:

<table width="100%"><tr> – мы задаем таблицу без рамок, общей шириной в 100 %, т. е. она будет занимать всю ширину браузера. Далее тег *<tr>* указывает на начало строки таблицы;

<td width="25%"></td> – создаем внутри строки первую левую ячейку таблицы шириной в 25 %. Эта ячейка пока пуста (между тегами *<td>* и *</td>* ничего нет). Здесь мы потом разместим ссылки в виде кнопок на другие страницы;

<td width="50%" valign="top">

<img src="salut.gif"></td> – создаем вторую ячейку *<td>*, шириной в 50 % от общей ширины браузера. Атрибут *valign* задает выравнивание картинки в ячейке по высоте. Этому атрибуту присвоено значение *top*, т. е. картинка выравнивается по верхней границе данной ячейки. Внутрь ячейки помещаем нашу картинку и закрываем тег этой ячейки *</td>*;

<td width="25%">

* *

Это учебный сайт. Он позволит быстро и наглядно изучить основные принципы программирования на PHP. Вы научитесь составлять скрипты для сайта любой сложности. Эти скрипты помогут вам автоматизировать добавление разнообразного материала на ваш сайт, вести подсчет посетителей вашего сайта, узнать мнение посетителей о вашем сайте при помощи гостевой книги или системы оценки материалов </td>

</tr></table> – задаем третью, правую ячейку, шириной в 25 % от ширины окна браузера. Помещаем туда контейнер * ... * вместе с текстом. Далее закрываем ячейку *</td>*, закрываем пока единственную строку этой таблицы *</tr>*, закрываем тег самой таблицы *</table>*.

Текст, который до таблицы был слева, в таблице помещен в правую колонку для удобства.

Если вы заметили, то мы убрали из тегов * * и *<img>* атрибут *width*, который при отсутствии таблицы указывал ширину объекта в % относительно ширины браузера. Сейчас в этом уже нет необходимости, так как эти объекты помещены в ячейки таблицы, а для каждой ячейки у нас определена своя ширина в процентах. Если бы мы оставили в теге * * атрибут *width="25%"*, то текст бы занял ширину в 25 % относительно данной ячейки таблицы, а не всей таблицы.

Результат листинга 3.5 представлен на рис. 3.6.

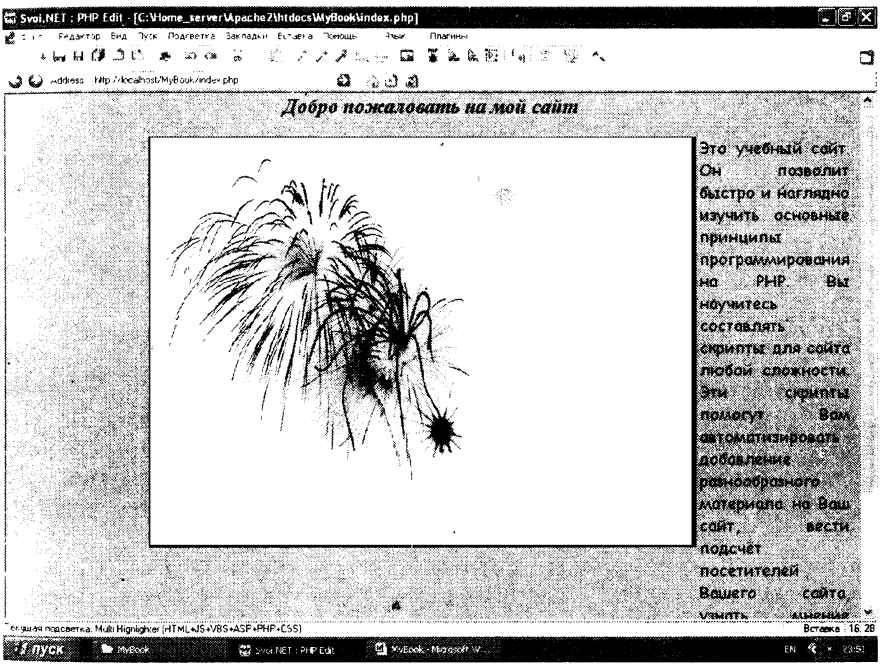


Рис. 3.6

3.6. СОЗДАНИЕ ССЫЛОК

В левую ячейку созданной нами таблицы поместим ссылки на другие страницы нашего сайта. Для начала сделаем простые ссылки, которые будут выглядеть как подчеркнутый текст, при нажатии на который будет происходить переход на заданную страницу.

Простая ссылка создается следующим образом:

<A href="адрес страницы, куда переходим">текст ссылки</A>

<A> – элемент привязки, который отображает текст ссылки на экране и указывает браузеру, на какую страницу необходимо перейти. При щелчке по тексту ссылки происходит переход на страницу, указанную в атрибуте *href*. В качестве текста ссылки может быть краткая информация о странице, на которую мы переходим.

Создадим 6 ссылок на шесть страниц нашего сайта. Соответственно, у нас в левой ячейке таблицы появятся названия этих шести ссылок. Назовем их следующим образом:

- новости сайта,
- гостевая книга,
- музыка, фото,
- статьи, магазин:

<A href="novosti/new1.php">Новости сайта</A>
<A href="gostevaja/gostev1.php">Гостевая книга</A>
<A href="muzic/muzic.php">Музыка</A>
<A href="risynki/kartinki.php">Фото </A>
<A href="statji.php">Статьи </A>
<A href="magazin/magazin.php">Магазин </A>

Пока наш сайт состоит только из одной страницы *index.php*, с которой мы сейчас и работаем. Других страниц пока нет, поэтому, естественно, ссылки срабатывать не будут. Шаг за шагом мы будем создавать эти шесть страниц. В конце концов, мы создадим полноценный сайт со множеством страниц, число которых может быть и тысячу, но они благодаря скриптам на PHP на тех шести страницах будут уже создаваться автоматически, без вашего участия. Но об этом позже. Итак, вставим приведенные выше ссылки в наш код. Вставить их надо в пустую первую ячейку таблицы после тега *<tr>* между тегами *<td width="25%">* и *</td>* (см. листинг 3.5 выше).

В листинге 3.6 приведен код с уже вставленными ссылками.

Листинг 3.6

```
<html>
<head>
<title>Главная страница</title>
<link type="text/css" rel="stylesheet" href="stil.css">
</head>
<body>
<h2 align="CENTER"><I>Добро пожаловать на мой сайт</I></h2>
<table width="100%"><tr>
```

```
<td width="25%" valign="top">
<A href="novosti/new1.php">Новости сайта</A><br><br>
<A href="gostevaja/gostev1.php">Гостевая книга</A><br><br>
<A href="muzic/muzic.php">Музыка</A><br><br>
<A href="risynki/kartinki.php">Фото </A><br><br>
<A href="statji/statji.php">Статьи </A><br><br>
<A href="magazin/magazin.php">Магазин </A>
</td>
<td width="50%" valign="top">
<img src="salut.gif"></td>
<td width="25%">
<div>
Это учебный сайт. Он позволит быстро и наглядно изучить
основные принципы программирования на PHP. Вы научитесь
составлять скрипты для сайта любой сложности. Эти скрипты
помогут вам автоматизировать добавление разнообразного материала
на ваш сайт, вести подсчет посетителей вашего сайта, узнать мнение
посетителей о вашем сайте при помощи гостевой книги или системы
оценки материалов
</div></td>
</tr></table>
</body>
</html>
```

Перед ссылками, в теге *<td width="25%" valign="top">* мы используем уже знакомый нам атрибут *valign* со значением top, т. е. выравниваем блок ссылок по верхнему краю таблицы. После каждой ссылки (после закрывающего тега *</A>*) ставим 2 одиночных тега *
*. Этот тег означает пропуск строки, т. е. мы делаем расстояние по вертикали между ссылками в две строки. Вообще это дело вкуса. Можно поставить после каждой ссылки и 5 тегов *
*, увеличив еще больше расстояние между ними.

Сохраните страницу и запустите браузер. В левой части окна браузера появятся ваши 6 ссылок. При нажатии на любую из них, появится сообщение, что невозможно найти страницу, поскольку эти страницы мы еще не создали.

Если вас не устраивает формат текста ссылок, то вы можете его изменить. Откройте созданный нами раньше внешний файл таблиц стилей *stil.css*. У нас там пока две строчки. Их трогать не нужно. Запишите внизу следующие строчки:

```
A {font-family:Comic Sans MS; font-size:14pt; color:black}
A:active{color:yellow}
A:hover{color:red}
```

Верхняя строчка дает указание браузеру, что все элементы <*A*> должны оформляться так, как указано в фигурных скобках. А в скобках мы указали название шрифта текста ссылок, размер шрифта и его цвет (изначально черный). Вторая строчка устанавливает цвет активной ссылки, на которую пользователь щелкнул мышью. Параметр *active* как раз и устанавливает стиль отображения активных ссылок, а именно, они отображены желтым цветом. Третья строчка указывает на отображение ссылки при наведении на нее указателя мыши. При наведении указателя мыши на какую-либо ссылку, она сменит черный цвет на красный. Если вы добавите в фигурные скобки еще и какой-нибудь другой тип шрифта или размер, то при наведении на ссылку указателя мыши будет изменен и сам шрифт. Вы, таким образом, как бы «оживите» ссылку. В общем, все это дело вашей фантазии. А пока сохраните файл стилей. Перезапустите браузер в редакторе и посмотрите, как изменился шрифт текста ссылок. Наведите указатель мыши на любую из ссылок и посмотрите, как меняется ее цвет.

Результат работы листинга 3.6 представлен на рис. 3.7.

Ну вот, собственно, и все о HTML. С его помощью можно составить любую HTML-страницу с любым дизайном. Существует много литературы по верстке страниц на языке HTML. Здесь рассмотрена только малая часть того, что может этот язык, ибо целью книги является научить вас писать скрипты на PHP и использовать их на практике для своего сайта.

Приступим теперь к изучению и практическому использованию языка PHP5 для нашего сайта. Создадим на главной странице index.php счетчик посещения.

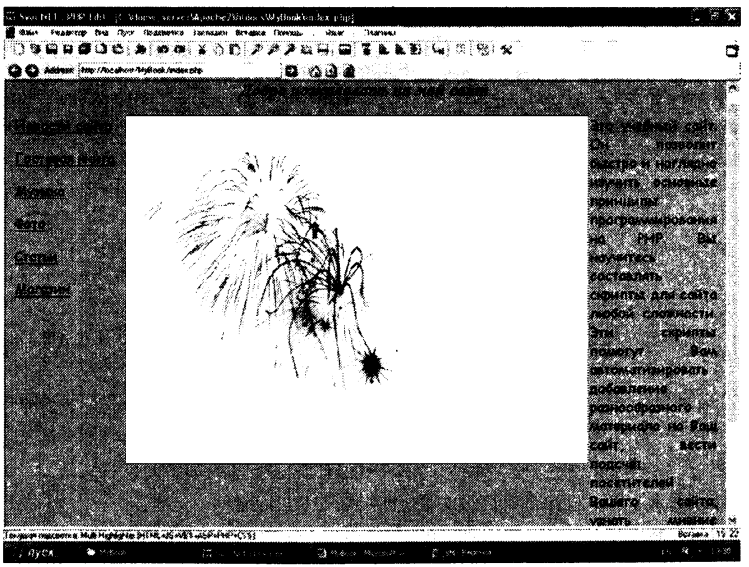


Рис. 3.7

Глава 4. ОСНОВЫ PHP5

4.1. ОСНОВНЫЕ ПОНЯТИЯ

Прежде чем приступить к составлению первого практического скрипта, рассмотрим основы синтаксиса языка PHP5.

Программы на PHP встраиваются в текст web-страницы при помощи окаймляющих угловых скобок с вопросительными знаками и указанием языка:

```
<?php
...
текст программы
...
?>
```

Команды PHP обязательно разделяются символом точки с запятой – «;» (символ конца абзаца или конца строки не учитывается никак), после последней в программе команды его можно не ставить. Также символ «;» не ставится после условных операторов (*if, switch*) и операторов цикла (*for, while* и других). Эти операторы мы рассмотрим чуть позже.

В программном коде листинга 3.6 после закрывающего тега таблицы *</table>* вставьте две строки:

```
<?php
?>
```

Теперь мы будем писать код только между этими двумя строчками.

Первый оператор, который мы рассмотрим, это оператор *echo*. С его помощью можно, например, вывести текст в окно браузера:

```
<?php
echo "Вывод текста на экран";
?>
```

Наберите эту строчку, сохраните страницу и запустите браузер. Слева внизу нашей страницы вы увидите набранный текст.

Текст можно также заключить в HTML-теги, например:

```
<?php
echo "<div style='color:yellow; font-family:Arial Black'>Вывод текста
  на экран</div>";
?>
```

Шрифт и цвет текста изменится, как указано в элементе *style*. Обратите внимание, что внутри двойных кавычек можно использовать только одинарные. Если бы мы заключили свойства элемента *style* в двойные кавычки *style="color:yellow"*, то программа выдала бы ошибку.

Как и во многих языках программирования, в PHP для хранения и изменения данных используются переменные. Любое имя переменной должно начинаться со знака *$*, не иметь пробелов. После знака *$* не должна первой стоять цифра. Переменные в PHP могут быть четырех типов – число (целое и дробное), строка текста, массив и объект. Интерпретатор PHP автоматически определяет тип переменной на основании анализа ее содержимого.

Примеры переменных:

$chislo=18 – переменной *chislo* присвоено значение 18;

$text="Простой текст"* – переменной *$text присвоена строка *"Простой текст"*. В данном случае мы имеем дело с переменной типа *String* (строковая переменная). Любой текст, присваиваемый строковой переменной, должен заключаться в двойные или одинарные кавычки. Однако двойные кавычки нельзя использовать внутри двойных (аналогично, внутри одинарных кавычек нельзя использовать одинарные).

Например, так писать нельзя:

$text="Фирма "Супер-Пупер" предлагает товары по низким ценам".

Эту фразу нужно писать так (внутри двойных кавычек использовать одинарные):

$text="Фирма 'Супер-Пупер' предлагает товары по низким ценам".

Если вы все-таки хотите использовать двойные кавычки внутри двойных, то можно прибегнуть к использованию символа экранирования «\», который, как бы, скрывает эти внутренние кавычки от php-интерпретатора. Они как бы есть, но программа их не замечает и ошибки не выдает. Этот символ ставится перед символом, который вы хотите экранировать (в нашем случае, перед кавычками). В этом случае приведенную выше фразу можно записать и так:

$text="Фирма \"Супер-Пупер\" предлагает товары по низким ценам";

Массив – это совокупность под одним именем пронумерованных переменных. Имя каждой переменной в массиве состоит из имени этого массива и индекса переменной – номера переменной в массиве. Индекс переменной может быть цифровым или символьным, т. е. представлять собой либо номер переменной в массиве, либо ее имя в нем.

Например, вот массив с числовыми индексами (нумерация индексов начинается с нуля, а не с единицы!):

$a[0]=5; $a[1]=8; $a[2]=12;

а вот с символьными:

$a['первый']=5; $a['второй']=8; $a['третий']=12;

Зачем нужны массивы? А нужны они для того, чтобы можно было к ним обращаться как к чему-то целому, тем самым получая возможность совершать автоматические действия со всеми элементами массива или с частью этих элементов, не указывая имени каждого их элемента. Иными словами, допустим, в какие-то переменные мы записали имена клиентов и теперь желаем вывести их. Как это сделать? Естественно, только перебрав все эти переменные, для чего нам понадобятся имена этих переменных, которые придется жестко задать в программе. А если заранее неизвестно, сколько будет клиентов, как тогда быть? Если же имена клиентов поместить в массив, то все их можно перебрать специальной командой (рассмотрим чуть позже), добавить же новое имя тоже нетрудно.

В PHP добавлять элементы в массив можно как явно указывая индекс элемента (например, *$a [50]="Москва"*), так и просто упоминая, в какой массив этот элемент добавляется – *$a[]="Москва"*. В последнем случае добавляемый элемент становится последним в массиве. Массив еще можно задать и так:

$m=array("Иванов", "Петров", "Сидоров");

В данном случае создается массив *$m*, состоящий из трех элементов, причем нулевым элементом этого массива будет *Иванов*, первым элементом – *Петров*, вторым – *Сидоров*.

Вывести элементы массива в окно браузера можно при помощи функции *print_r($m)*, где *$m* – массив элементов.

4.2. ОПЕРАТОРЫ И ДЕЙСТВИЯ НАД ПЕРЕМЕННЫМИ

Над переменными, как и над числами, можно проводить различные математические операции.

$chislo=$chislo+3 или *$chislo+=3*. В любом из этих случаев мы увеличиваем значение переменной *$chislo* на 3. Аналогично, чтобы уменьшить значение переменной на 3:

$chislo=$chislo-3 или $chislo-=3.

*$chislo=$chislo*3* или *$chislo*=3*. Увеличиваем значение переменной в 3 раза.

$chislo=$chislo/3 или *$chislo/=3*. Уменьшаем значение переменной в 3 раза.

$chislo++*. Увеличиваем значение переменной на 1. Аналогично, для уменьшения переменной на 1: *$chislo--.

$chislo.$text. Точка между двумя переменными означает операцию конкатенации или соединения. Рассмотрим пример:

echo $chislo.$text;

Если у нас в переменной *$chislo* число 18, а в переменной *$text* фраза «*Простой текст*», то приведенная выше строка выведет в браузер без кавычек: «*18Простой текст*».

Можно записать немного по-другому, без символа конкатенации, но в кавычках:

echo "$chislo$text";

Если вы просто напишете две переменные рядом без точки и кавычек:

echo $chislo $text; ,

то в этом случае браузер выведет ошибку.

В приведенных выше примерах были рассмотрены арифметические операторы, а также оператор конкатенации.

Нам понадобится еще множество других операторов, которые мы и рассмотрим.

Операторы отношения – это операторы, применяемые для сравнения значений переменных. Сравниваться могут не только числа, но и строки. Выражения, содержащие операторы сравнения, возвращают значение либо *True*, если выражение истинно, либо *False*, когда выражение ложно.

Применяются следующие операторы отношения:

$a == $b – проверка на равенство. Выражение истинно, когда переменные равны;

$a != $b – проверка на неравенство. Выражение истинно, когда переменные не равны;

$a > $b – проверка на больше. Выражение истинно, когда первая переменная больше второй;

$a < $b – проверка на меньше. Выражение истинно, когда первая переменная меньше второй;

$a >= $b – проверка на больше или равно. Выражение истинно, когда первая переменная больше или равна второй;

$a <= $b – проверка на меньше или равно. Выражение истинно, когда первая переменная меньше или равна второй;

$a === $b – проверка на идентичность (3 знака равенства). Здесь сравниваются не только значения, но и типы данных. Выражение истинно, когда первая переменная совпадает со второй переменной не только по значению, но и по типу. Например, если первая переменная имеет тип *integer* (целое число от -2 147 463 648 до 2 147 483 647), а вторая имеет тип *double* (вещественное число с плавающей точкой), то данные переменные будут не идентичны. Если, например, *$a=4*, а *$b=4.0*, то выражение *$a===$b* будет ложным.

Логические операторы:

$a && $b – возвращает *True*, если имеет место (истины выражения) и *$a* и *$b* (*&&* – логическое «и»);

$a || $b – возвращает *True*, если имеет место или *$a* или *$b* (|| – логическое «или»).

Условные операторы – операторы, проверяющие какое-либо выражение и в зависимости от значения этого выражения (истинно или ложно) выполняющие те или иные действия.

Оператор if. Его синтаксис *if(выражение) {последовательность действий}*. Если выражение в круглых скобках верно, то выполняется последовательность действий в фигурных скобках. Если выражение ложно, то все действия в фигурных скобках игнорируются. Есть другой синтаксис оператора:

if(выражение)
{последовательность действий1};
else {последовательность действий2};

Если выражение в круглых скобках верно, то выполняется последовательность действий1, иначе (если выражение ложно) выполняется последовательность действий2, после оператора *else*.

Немного попрактикуемся. В редакторе создайте новый PHP-файл. Уберите из него все теги html, так как они нам здесь не нужны. Оставьте только две строки *<?php* и *?>*. Между этими строками наберите следующий код (листинг 4.1).

Листинг 4.1

```
<?php
$a=10;
$b=20;
$c="Язык программирования PHP5";
$d="лучший язык для построения сайта!";
$a/=2; $b*=3; $f=$b/$a;
if(($f>=$a+2)||($a*$b==5*$f))
 {
  echo $c."--".$d;
  echo "<br>";
  echo "$f<br>$b";
 }
else
 {
  echo "PHP--это самый ".$d."<br>";
  echo $a.$b;
 }
?>
```

Сначала мы присваиваем переменным *$a* и *$b* числовые значения, а переменным *$c* и *$d* строковые выражения. Затем со значениями переменных *$a* и *$b* производим математические операции, *$a* делим на 2, а *$b* умножаем на 3. Переменной *$f* присваиваем значение, равное результату деления значения переменной *$b* на значение переменной *$a*. Все это можно разместить на одной строке, через точку с запятой. В результате числовые переменные получат следующие значения: *$a=5; $b=60; $f=12*.

Далее идет условный оператор *if(($f>=$a+2)||($a*$b==5*$f))*. Здесь проверяются сразу 2 условия. Между двумя условиями стоит логический оператор || (логическое «или»). Первое выражение в операторе *if* истинно (*12>=5+2*). Второе выражение ложно (*5*60==5*12*). Поскольку между двумя выражениями стоит логическое «или», то в целом общее выражение будет истинным, поскольку нам для истинности достаточно, чтобы хотя бы одно из выражений было истинным. В данном случае истинно первое выражение, а значит, все выражение оператора *if* истинно. Следовательно, будет выполняться код, который следует сразу за оператором *if* в фигурных скобках. А там у нас идет вывод в окно браузера строковых переменных *$c* и *$d*, разделенных тире «–». Поскольку тире является тоже текстом, его заключают в двойные кавычки. Оператор *$c*, тире и оператор *$d* соединены между собой точкой, т. е. оператором конкатенации. Далее идет тег html, означающий здесь переход на следующую строку (*
*). В PHP-языке любые HTML-теги должны обязательно находиться в кавычках. Оператор *echo* выводит этот тег, вернее результат его действия в браузер. Далее тот же оператор *echo* выводит в браузер значения числовых переменных *$f* и *$b*, разделенных строкой (тегом *
*). Здесь переменные и разделитель строки находятся в общих двойных кавычках (*echo "$f
$b"*). Можно записать и по-другому:

```
echo $f."<br>".$b;
```

Здесь мы используем оператор конкатенации Оба способа верны, но в любом случае любые теги html в программном коде PHP должны быть заключены в кавычки!

Если бы общее выражение в условном операторе *if* было бы ложным, то выполнился бы только тот код, который находится во вторых фигурных скобках, после оператора *else*.

Здесь также использованы кавычки и оператор конкатенации. Вместо строчки:

```
echo "PHP – это самый ".$d."<br>"
```

можно было бы написать:

```
echo "PHP – это самый $d<br>",
```

т. е. без использования оператора конкатенации.

Я специально использую разные записи, чтобы было понятно, что вывести значения переменных в окно браузера можно разными способами.

Сохраните созданный нами PHP-файл в рабочей папке нашего сервера *hdocs* под любым именем, но обязательно с расширением *php*! Запустите браузер. В результате вы увидите:

Язык программирования PHP5 – лучший язык для построения сайта!

12

60

Выполнился программный код в первых фигурных скобках, поскольку общее выражение в условном операторе *if* истинно.

А теперь поменяйте в условном операторе логический оператор || на *&&*. Получим следующее условное выражение:

```
if(($f>=$a+2)&&($a*$b==5*$f))
```

Оператор *&&* обозначает логическое «и», т. е. чтобы общее выражение условного оператора было истинным, нужно, чтобы были истинны все условные выражения (у нас их два) в этом условном операторе. В нашем случае выражение в первых скобках истинно (*$f>=$a+2*), но выражение во вторых скобках (*$a*$b==5*$f*) ложно, поскольку 300 не равно 60. Следовательно, общее выражение условного оператора *if* ложно, поэтому будет выполняться программный код во вторых фигурных скобках, после оператора *else*. Запустив браузер, мы увидим следующее:

PHP – это самый лучший язык для построения сайта!

560

Число *560* получилось благодаря оператору конкатенации значений переменных *$a* и *$b* (у нас *$a=5*, *$b=60*), этот оператор не осуществляет арифметического сложения, а просто соединяет значения, как строки.

Оператор switch. Он нужен, когда значение переменной нужно сравнить с определенной величиной и выполнить тот или иной фрагмент кода, в зависимости от результата сравнения. Синтаксис оператора выглядит следующим образом:

```
switch (выражение)
{
 case вариант1:
 код
 break;
 case вариант2:
  код
```

```
  break;
 ...
  default:
   код
}
```

Выражение в операторе *switch* сравнивается с вариантами (*вариант1*, *вариант2* и т. д.). Сработает только тот фрагмент кода, где вариант будет соответствовать значению выражения. Если выражение не будет соответствовать ни одному из вариантов, то выполнится код после оператора *default*. Оператор *break* приостанавливает действие оператора *switch*. Оператор *break* не является обязательным. Если его убрать, то код будет выполняться дальше и после того, как будет найден вариант, соответствующий значению выражения в операторе *switch*. В общем случае данный оператор *break* используется для мгновенного выхода из какого-либо цикла. Для примера наберите в PHP-редакторе следующие строки:

Листинг 4.2

```
<?php
$f=12; $a=5;
$c="Язык программирования PHP5";
$d="лучший язык для построения сайта!";
 switch ($f)
{
 case 10:
  echo "Выражение не равно 10";
  break;
 case 12:
  echo "Выражение равно 12";
  break;

  default:
   echo $c."--".$d;
}
?>
```

Допустим, выражение в операторе *switch* равно 12 (*$f=12*). В этом случае выполнится код после оператора *case 12*, т. е. в окно браузера будет выведена фраза: «*Выражение равно 12*», после чего работа оператора *switch* завершится, поскольку дальше стоит оператор *break*.

Теперь поставьте в операторе *switch* вместо переменной *$f* переменную *$a*. Сохраните код и запустите браузер. Вы увидите выражение: «*Язык программирования PHP5 – лучший язык для построения сайта!*», т. е. сработает только код после оператора *default*, поскольку значение выражения в операторе *switch ($a=5)* не соответствует ни одному из вариантов.

Операторы цикла – служат для многократного выполнения той или иной части кода до тех пор, пока выполняется некоторое условие. Самый распространенный из этих операторов – оператор *for*. Его синтаксис выглядит так:

for(начальное значение счетчика цикла; условное выражение;
шаг счетчика цикла)
{программный код};

Начальное значение счетчика цикла – числовая переменная с неким начальным значением, с которого начинается выполнение программного кода цикла. Условное выражение – условие выполнения цикла. Программный код цикла будет выполняться до тех пор, пока условное выражение истинно. Шаг счетчика цикла – показывает, насколько будет увеличено значение счетчика после каждого однократного выполнения кода цикла.

Наберите в PHP-редакторе следующие строки:

Листинг 4.3

```
<?php
$k=1;
for($i=1; $i<=10; $i++)
{
 $k=$k*$i;
}
echo $k;
?>
```

Здесь приведен простой PHP-сценарий для вычисления факториала числа. В операторе цикла *for* в качестве счетчика цикла выступает переменная *$i*. Выражение *$i<=10* говорит о том, что код цикла (то, что находится в фигурных скобках) будет выполняться, пока значение переменной *$i* не превосходит 10. Выражение *$i++* означает, что после каждого однократного выполнения кода в фигурных скобках значение переменной *$i* увеличивается на единицу. Код в фигурных скобках у нас будет выполняться 10 раз. Переменная *$k* у нас будет увеличиваться в *$i* раз после каждого выполнения кода. Когда значение переменной *$i* будет равно 11, условное выражение в операторе *for* станет ложным (*$i<=10*) и цикл прервется. Начнется выполнение кода, расположенного после закрывающейся фигурной скобки оператора цикла, т. е. в окно браузера выведется значение переменной *$k* (*3628800=1*2*3*4*5*6*7*8*9*10*).

Вкратце рассмотрим другие операторы цикла.

Оператор цикла *while* имеет синтаксис:
while (условие) { программный код }

Это цикл с условием. Команды в фигурных скобках выполняются до тех пор, пока выполняется условие в круглых скобках цикла *while*. Для того чтобы цикл прервался, нужно, чтобы условие выполняться перестало, т. е. стало ложным. Поэтому внутри цикла необходимо предусмотреть возможность влиять на это условие. Скажем, цикл *while ($i<=10) {...программный код... ; $i++; }* будет выполняться до тех пор, пока значение переменной *$i* не превысит 10. Если изначально оно было равно 1, то цикл выполнится 10 раз.

Цикл *do...while* имеет синтаксис:
do {программный код } while (условие)

Он работает так же, однако команды, указанные в фигурных скобках, будут выполнены по меньшей мере 1 раз – даже если условие выполняться не будет, так как условие для выполнения цикла проверяется уже после выполнения программного кода.

Рассмотрим теперь некоторые функции в PHP для работы с файлами, которые нам понадобятся для создания счетчиков посещения на страницах сайта.

4.3. ФУНКЦИИ PHP ДЛЯ РАБОТЫ С ФАЙЛАМИ

Данные счетчика мы будем хранить в текстовых файлах. Почти любая операция с файлами – создание нового файла, чтение или запись в существующий файл предваряется операцией открытия файла. Открытие файла на сервере осуществляется при помощи функции **fopen($filename, mode)**. В качестве первого аргумента функции выступает имя файла, а точнее, путь к нему. Второй аргумент *mode* задает режим работы с открываемым файлом и принимает следующие значения:

r – открыть файл только для чтения и приготовиться читать его с начала;

r+ – открыть файл для чтения и для записи и приготовиться работать с ним с его начала;

w – открыть файл только для записи, предварительно удалив из него все содержимое, причем если файл с указанным именем не существует, то создается новый файл с таким именем;

w+ – открыть файл как для записи, так и для возможного последующего чтения, предварительно удалив из него все содержимое, причем если файл с указанным именем не существует, то создается новый файл с таким именем;

a – открыть файл только для записи и приготовиться дописывать данные в его конец. Если файл с указанным именем не существует, то создается новый файл с таким именем;

a+ – открыть файл для записи и для чтения и приготовиться дописывать данные в его конец. Если файл с указанным именем не существует, то создается новый файл с таким именем.

Для чтения содержимого из файла применяется функция **fread($descr, length)**. Первый аргумент – дескриптор открываемого файла (то, что возвращает функция *fopen*), второй – количество считываемых байтов. Если мы хотим считать содержимое всего файла, то вместо аргумента *length* прибегают к функции **filesize($filename)** – возвращает число байтов, содержащихся в файле. Переменная *$filename* означает название файла или путь к нему. В итоге функция *fread* вместе с аргументами будет выглядеть так:

fread($descr, filesize($filename))

Запись в открытый файл осуществляется функцией **fwrite($descr, "выражение")**. Здесь аргумент *$descr* тоже означает дескриптор файла, который возвращается функцией *fopen*, т. е. *$descr=fopen($filename, mode)*. Именно дескриптор, а не путь к файлу используется в функциях *fread* и *fwrite*! Второй аргумент функции *fwrite* представляет собой выражение (число, текст и т. п.), которое мы и записываем в открытый файл.

После окончания работы с файлом, его следует закрыть с помощью функции: **fclose($descr)**. Единственный аргумент этой функции – дескриптор ранее открытого файла.

Удалить какой-либо файл можно при помощи функции **unlink("filename")**, где *filename* – название или путь к файлу (не дескриптор!), который нужно удалить.

Закрепим теоретические знания небольшим примером. Создайте в папке *hdocs*, где находятся наши файлы, простой текстовый файл. Пусть он будет называться *zapisi.txt*. Откройте его и запишите две строчки:

Первая запись

Вторая запись

Сохраните и закройте этот файл.

Создайте новый PHP-файл в PHP-редакторе и сохраните его в папке htdocs под любым названием с расширением php. Запишите в него следующие строки (листинг 4.4):

Листинг 4.4

```
<?php
$zapisi="zapisi.txt";
$p=fopen($zapisi, "r");
```

```
$n=fread($p, filesize($zapisi));
fclose($p);
echo $n;

$myfile="myfile.txt";
$m=fopen($myfile, "w+");
fwrite($m, "Моя запись в файл");
fclose($m);
?>
```

Данная программа состоит как бы из двух программ. В первой части программы мы считываем содержимое файла и выводим его в окно браузера. Во второй части мы создаем новый файл и делаем там запись. Я их объединил в 1 листинг, чтобы не писать 2.

Итак, в первой части программы мы присваиваем переменной *$zapisi* путь к нашему текстовому файлу, который мы создали. Поскольку этот файл находится в нашем рабочем каталоге, то в качестве пути к нему достаточно указать просто имя файла с расширением. Потом мы открываем этот файл, но только для чтения, поскольку вторым аргументом функции *fopen* мы указали параметр *r*. Функция возвращает дескриптор файла *$p*, который и используется в дальнейшем. Затем, с помощью функции *fread* мы считываем содержимое открытого файла и заносим это содержимое в переменную *$n*. Функция filesize, как уже говорилось, возвращает количество байтов в файле. Обратите внимание, что аргументом этой функции является путь к файлу (оно у нас в переменной *$zapisi*), а не его дескриптор. Далее функцией *fclose* закрываем файл (точнее, его дескриптор). Потом в окно браузера мы выводим содержимое переменной *$n*, в которую мы записали содержимое файла, считанного при помощи функции *fread*.

Во второй части кода переменной *$myfile* мы присваиваем путь к файлу *myfile.txt*. Этот файл мы еще не создали. Функцией *fopen*. со вторым аргументом «*w+*» создается новый пустой файл для записи. Эта функция возвращает дескриптор созданного файла *$m*. Далее функция *fwrite* записывает в пустой файл то, что стоит в качестве второго аргумента этой функции, а именно «*Моя запись в файл*». Последняя строчка закрывает созданный файл.

Сохраните код и запустите браузер. В окно браузера выведется содержимое файла *zapisi.txt*, который мы создали вручную.

Теперь зайдите в рабочий каталог. Вы увидите там новый файл *myfile.txt*, который и создала написанная вами программа. Открыв этот файл, вы увидите там записанную строчку «*Моя запись в файл*».

Запись программно (при помощи функции *fwrite*) что-либо в файл несет в себе одну неприятность. Рассмотрим, например, работу счетчика посещений вашего сайта (более подробно он будет рассмотрен ниже). При посещении посетителем страницы со счетчиком программа открывает текстовый

файл с числом посещений при помощи функции *fopen*, затем делает новую запись, т. е. записывает число, увеличенное на единицу. Затем при помощи функции *fclose* закрываем файл. Все пройдет нормально, если запись в файл происходит как бы единолично, т. е. зашел один посетитель на страницу, программа для записи в файл успешно сработала. Затем зашел другой посетитель, программа записи опять сработала. Но допустим, что на страницу одновременно зашло несколько посетителей. Программа не может выполняться несколько раз одновременно. Только после того как программа «обслужит» одного посетителя, она сможет корректно «обслужить» другого! При одновременном использовании программы для записи в файл несколькими посетителями могут возникнуть конфликтные ситуации, которые приведут к нарушению или полному разрушению информации в файле для записи. Разумно предположить, что пока программа работает для одного посетителя (даже если на это уходят микросекунды), допуск к ней других посетителей следует заблокировать. Для блокировки файла имеется специальная функция *flock()*. Она блокирует файл и не позволяет посетителям читать или записывать в него до тех пор, пока блокировка не будет снята. Функция блокировки имеет 3 аргумента, но мы рассмотрим только два из них, которые нам понадобятся: *flock($deckript, type)*. Первый аргумент представляет собой дескриптор блокируемого файла, который возвращает функция *fopen*. Второй аргумент *type* указывает на тип операции с файлом и нам понадобится 2 значения этого аргумента:

LOCK_EX – блокировка файла;

LOCK_UN – снятие блокировки, чтобы другие посетители могли получить доступ к файлу.

Функция *flock* вызывается сразу после открытия файла и перед закрытием файла для снятия с него блокировки. Перепишем листинг 4.4 следующим образом:

Листинг 4.4а (с использованием блокировки файла)

```
<?php
$zapisi="zapisi.txt";
$p=fopen($zapisi, "r");
flock($p, LOCK_EX);
$n=fread($p, filesize($zapisi));
flock($p, LOCK_UN);
fclose($p);
echo $n;

$myfile="myfile.txt";
$m=fopen($myfile, "w+");
flock($p, LOCK_EX);
```

```
fwrite($m, "Моя запись в файл");
flock($p, LOCK_UN);
fclose($m);
?>
```

Здесь мы применяем блокировку в двух случаях: при чтении файла и при записи в него. При открытии файла, когда посетитель посещает страницу с этим кодом и запускает его, файл открывается и блокируется от доступа к нему других посетителей. После чтения из файла или записи в него блокировка снимается и файл опять готов для чтения и записи другими посетителями. Это прием мы будем использовать для создания счетчика посещения страниц. Там я применял блокировку при записи в файл (файлы) новых данных о посещении страницы.

Для создания счетчика нам понадобится еще несколько функций для работы с файлами.

Функция **file(filename)** считывает файл и возвращает массив, каждый элемент которого обозначает строку в прочитанном файле.

Функция **file_exists("file")** проверяет, существует ли файл, что указан в круглых скобках или нет (*file* – путь к этому файлу). Если файл существует, функция возвращает *True*. В противном случае – *False*.

Функция **foreach (массив as переменная) {программный код;}** присваивает переменный элемент массива и выполняет программный код. Этот цикл повторяется до тех пор, пока не закончатся все элементы массива.

Функция **count(массив)** возвращает число элементов в массиве.

Функция **in_array($znach, $mass)** ищет значение *$znach* в массиве *$mass* и возвращает *true*, если это значение найдено, в противном случае возвращает *false*.

Для примера создайте в рабочем каталоге новый текстовый файл, например *massiv.txt*. Запишите туда 5 строк:

первый элемент
второй элемент
третий элемент
четвертый элемент
пятый элемент

Сохраните и закройте файл. Наберите в PHP-редакторе следующий код (листинг 4.5):

Листинг 4.5

```
<?php
$massiv="massiv.txt";
$q=file($massiv);
foreach($q as $stroka)
```

```
{
echo "$stroka<br>";
$r[]=$stroka;
}
echo count($r);
?>
```

Итак, функция *file* возвращает массив строк файла *massiv.txt*. Массив сохраняется в переменной *$q*. Далее функция *foreach* присваивает переменной *$stroka* сначала первый элемент массива *$q*. Далее выполняется код в фигурных скобках. А именно, мы выводим в окно браузера значение переменной *$stroka*, которой был присвоен первый элемент массива (первая строка файла *massiv.txt*). Затем мы значение переменной *$stroka* присваиваем первому элементу нового массива *$r*. Далее цикл повторяется. Переменной *$stroka* присваивается уже второй элемент массива *$q* (первый элемент массива в переменной *$stroka* стирается). Уже второму элементу массива *$r* присваивается новое значение переменной *$stroka*. Итак, цикл будет повторяться, пока не будут перебраны все элементы массива *$q*. Потом мы выводим число строк массива *$r*. Это число, как и в массиве *$q*, равно пяти, именно столько строк мы написали в текстовом файле *massiv.txt*. Все строки у нас уже будут в массиве *$r[]*.

4.4. РАБОТА СО СТРОКАМИ

Здесь мы рассмотрим некоторые функции для преобразования строк. При составлении php-программ эти функции будут важны для подсчета, например, числа символов в строке или для проверки вводимых данных пользователями на вашем форуме.

strlen("str") – возвращает длину строки *str*, т. е. количество символов. При этом учитываются служебные символы, например пробел или перевод строки (перевод строки состоит из двух символов!).

trim("str") – удаляет все пробелы в начале и в конце строки.

ltrim("str") – удаляет пробелы только в начале строки.

rtrim("str") – удаляет пробелы только в конце строки.

substr("str", start, length) – возвращает часть строки *str* с позиции (с символа) *start* и длиной символов *length*. Например, *substr("программирование", 4, 5)* возвратит *рамми*, т. е. возвращаемая часть строки начинается с четвертого символа (нумерация начинается с нуля) и имеет длину в 5 символов. Если третий аргумент функции *substr* отсутствует, то она возвратит часть строки с позиции, указанной во втором аргументе *start* до конца строки. Если третий аргумент функции отрицательный, то конец строки будет обрезан

на указанное число символов в этом аргументе, т. е. *substr("программирование", 4, -2)* возвратит *раммирован*.

stristr("str", "simvol") – возвращает часть строки, начиная с символа *simvol* и до конца строки. Если символ не найден, то возвращается *false*.

strchr("str", "simvol") – возвращает часть строки, начиная с символа *simvol* и до конца строки. Если символ не найден, то возвращается *false*. В отличие от предыдущей функции, поиск символа начинается с конца строки.

str_replace("str1", "str2", "str") – заменяет в строке *str* все вхождения *str1* на *str2*.

str_repeat("str", n) – повторяет строку *n* раз и возвращает ее.

strrev("str") – возвращает строку, где символы следуют в обратном порядке.

strpos("str", "str1") – возвращает позицию первого вхождения подстроки *str1* в строке *str*.

strrpos("str", "str1") – возвращает позицию последнего вхождения подстроки *str1* в строке *str*.

strtolower("str") – преобразует символы строки *str* в нижний регистр и возвращает ее.

strtoupper("str") – преобразует символы строки *str* в верхний регистр и возвращает ее.

ucfirst ("str") – преобразует первый символ строки *str* в верхний регистр и возвращает ее.

ucwords ("str") – преобразует первый символ каждого слова строки *str* в верхний регистр и возвращает ее.

str_word_count("str", nomer) – возвращает массив слов в строке. Параметр *nomer* определяет вид массива. Если в качестве этого параметра задать значение 1, то числовые индексы элементов возвращаемого массива будут соответствовать порядковому номеру слова в строке *str*. Если мы параметру *nomer* присвоим значение 2, то числовые индексы элементов массива будут соответствовать номеру позиции слова в строке. Следует помнить, что нумерация элементов в массиве начинается с нуля. Если параметр *nomer* не указан, то возвращается число слов в строке.

strtok("str", "str1") – возвращает первую подстроку строки *str*, отделенной от остальной части строки символом, указанным в строке *str1*. Эта функция бывает удобна, когда нужно, например, отделить название файла от его расширения.

parthinfo("myfile") – функция для работы с путями к файлам и каталогам. Здесь *myfile* – путь к файлу. Функция возвращает массив (присвоим ему имя *$mass*), хранящий в своих элементах: путь к директории, по которому распо-

ложен файл (*$mass["dirname"]*), имя файла (*$mass["basename"]*), расширение файла (*$mass["extension"]*).

Далее я приведу пример, который наглядно покажет действия приведенных выше функций для работы со строками.

Создайте в PHP-редакторе новый PHP-файл и сохраните его в рабочей папке нашего сервера под названием *stroki.php*. Наберите следующий код:

Листинг 4.6 (файл stroki.php)

```
<?php
$str="  Эта строка, которую Вы видите, будет преобразована ";
$eng="Welcome in the world of programming!";

$dlina=strlen($str);
echo "$dlina<br>";

$itog=trim($str);
$dlina=strlen($itog);
echo "$itog $dlina<br>";

$itog=ltrim($str);
$dlina=strlen($itog);
echo "$itog $dlina<br>";

$itog=rtrim($str);
$dlina=strlen($itog);
echo "$itog $dlina<br>";

$itog=substr($str, 6, 25);
echo "$itog<br>";

$itog=stristr($str, "в");
echo "$itog<br>";

$itog=strchr($str, "п");
echo "$itog<br>";

$itog=str_replace("будет", "уже", $str);
echo "$itog<br>";

$itog=str_repeat($str, 2);
echo "$itog<br>";

$itog=strrev($str);
echo "$itog<br>";
```

```
$itog=strpos($str, "видите");
echo "$itog<br>";

$itog=strrpos($str, "р");
echo "$itog<br>";

$itog=strtolower($str);
echo "$itog<br>";

$itog=strtoupper($str);
echo "$itog<br>";

$itog=ucfirst ($str);
echo "$itog<br>";

$itog=ucwords ($str);
echo "$itog<br>";

$itog=str_word_count($eng, 1);
print_r($itog);
echo "<br>";

$itog=str_word_count($eng);
print_r($itog);
echo "<br>";

$itog=strtok($str, ",");
echo "$itog<br>";

$path=pathinfo("C:\server\Apache2\htdocs\stroki.php");
echo $path["dirname"]."<br>";
echo $path["basename"]."<br>";
echo $path["extension"];
?>
```

В начале листинга 4.6, мы присваиваем переменной *$str* строку на русском языке, которую обязательно заключаем в кавычки. При этом мы ставим перед строкой 2 пробела, а в конце строки еще один пробел. Переменной *$eng* присваиваем фразу на английском языке.

Запустите программу в браузере. Вы должны увидеть результат работы данной программы примерно как на рис. 4.1. Постарайтесь разобраться самостоятельно.

```
53
Эта строка, которую Вы видите  будет преобразована 50
Эта строка, которую Вы видите, будет преобразована 51
Эта строка, которую Вы видите, будет преобразована 52
строка, которую Вы видите
Вы видите, будет преобразована
преобразована
Эта строка, которую Вы видите, уже преобразована
Эта строка, которую Вы видите, будет преобразована Эта строка, которую Вы видите, будет преобразована
анавозарбоерп тедуб ,етидив ыВ юуроток ,акорте атЭ
25
44
эта строка, которую вы видите, будет преобразована
ЭТА СТРОКА, КОТОРУЮ ВЫ ВИДИТЕ, БУДЕТ ПРЕОБРАЗОВАНА
Эта строка, которую Вы видите, будет преобразована
Эта Строка, Которую Вы Видите, Будет Преобразована
Array ( [0] => Welcome [1] => in [2] => the [3] => world [4] => of [5] => programming )
6
Эта строка
C:\server\Apache2\htdocs
stroki.php
php
```

Рис. 4.1

Некоторые строчки кода я все-таки разберу:

$dlina=strlen($str); – переменной *$dlina* присваиваем количество символов в строке *$str*, причем включая все пробелы слева и справа;
*echo "$dlina
";* – выводим в браузер это количество символов. У нас их 53, что и отражено в первой строке браузера (см. рис. 4.1);

$itog=trim($str); – при помощи функции *trim* удаляем пробелы слева и справа у строки *$str* и присваиваем теперь эту строку (уже без пробелов слева и справа) переменной *$itog*;
$dlina=strlen($itog); – присваиваем количество символов в строке *$itog* опять переменной *$dlina* (старое значение сотрется);
*echo "$itog $dlina
";* – выводим в браузер нашу строку на русском языке, но уже без пробелов слева и справа. Рядом со строкой выводим количество символов в этой строке. В итоге вторая строка в браузере будет выглядеть так: *Эта строка, которую вы видите, будет преобразована 50*. После того как мы удалили пробелы (два в начале, один в конце строки), количество символов стало равно 50. В конце строки ставим тег *
*, означающий переход на новую строку;

$itog=ltrim($str); – при помощи функции *ltrim* удаляем пробелы только слева у строки *$str* и присваиваем теперь эту строку (уже без пробелов слева) переменной *$itog* (старое значение сотрется);

$dlina=strlen($itog); – присваиваем количество символов в строке *$itog* переменной *$dlina* (старое значение сотрется);

*echo "$itog $dlina
";* – выводим в браузер нашу строку на русском языке, но уже без пробелов слева. Рядом со строкой выводим количество символов в этой строке (их 51);

$itog=rtrim($str); – анологично при помощи функции *rtrim* удаляем пробелы только справа у строки *$str* и присваиваем теперь эту строку (уже без пробелов справа) снова переменной *$itog* (старое значение сотрется);
$dlina=strlen($itog); – присваиваем количество символов в строке *$itog* переменной *$dlina* (старое значение сотрется);
*echo "$itog $dlina
";* – выводим в браузер нашу строку на русском языке, но уже без пробелов справа. Рядом со строкой выводим количество символов в этой строке (их 52);

$itog=substr($str, 6, 25); – с помощью функции *substr* от строки *$str* оставляем только 25 символов, начиная с шестого символа (два пробела в начале строки тоже учитываются!);
*echo "$itog
";* – выводим в браузер урезанную строку (она будет пятой по счету);

$itog=stristr($str, "в"); – с помощью функции *stristr* создаем новую строку из строки *$str*, но уже с буквы «*в*». С этой буквы у нас начинается местоимение «*вы*». С этого местоимения и начнется теперь строка, которая будет выведена шестой в браузере;
*echo "$itog
";*

$itog=strchr($str, "n"); – с помощью функции *strchr* создаем новую строку из строки *$str*, но уже с буквы «*n*» (с первой буквы «*n*», но с конца строки!). В итоге седьмая строка будет состоять только из одного слова «*преобразована*»;
*echo "$itog
";*

$itog=str_replace("будет", "уже", $str); – с помощью функции *str_replace*, в строке *$str* заменим слово «*будет*» на «*уже*». Итог на восьмой строке;
*echo "$itog
";*

$itog=str_repeat($str, 2); – с помощью функции *str_repeat* создаем двойную строку *$str*. Итог на девятой строке;
*echo "$itog
";*

$itog=strrev($str); – с помощью функции *strrev* создаем строку *$itog*, которая будет записана в обратном порядке, по отношению к исходной строке *$str*. Итог на десятой строке;
*echo "$itog
";*

$itog=strpos($str, "видите"); – с помощью функции *strpos* находим позицию в исходной строке *$str*, с которого начинается слово «*видите*». Это двадцать пятая позиция (два пробела в начале исходной строки тоже учитываются!). Итог на одиннадцатой строке;
*echo "$itog
";*

$itog=strrpos($str, "p"); – с помощью функции *strrpos* находим позицию в исходной строке *$str*, которой соответствует последняя буква «*p*». Это сорок четвертая позиция (два пробела в начале исходной строки тоже учитываются!). Итог на двенадцатой строке;
*echo "$itog
";*

$itog=strtolower($str); – с помощью функции *strtolower* преобразуем все символы исходной строки *$str* в нижний регистр (все буквы строчные). Итог на тринадцатой строке;
*echo "$itog
";*

$itog=strtoupper($str); – с помощью функции *strtoupper* преобразуем все символы исходной строки *$str* в верхний регистр (все буквы заглавные). Итог на четырнадцатой строке;
*echo "$itog
";*

$itog=ucfirst ($str); – функция *ucfirst* преобразует первый символ исходной строки *$str* в верхний регистр. Но поскольку у нас первый символ – пробел, строка не изменится и выведется в браузер в исходном виде пятнадцатой строкой;
*echo "$itog
";*

$itog=ucwords ($str); – функция *ucwords* преобразует первые буквы всех слов в исходной строке в заглавные. Результат на шестнадцатой строке;
*echo "$itog
";*

$itog=str_word_count($eng, 1); – функция *str_word_count* возвращает массив отдельных слов в строке *$eng*. В качестве второго параметра этой функции задано значение 1, т. е. числовые индексы элементов возвращаемого массива будут соответствовать порядковому номеру слова в строке *$eng*. Увы, данная функция некорректно работает с русским языком, поэтому в качестве исходной строки мы использовали строку *$eng* с фразой на английском языке;
print_r($itog); – с помощью функции *print_r* выводим массив слов в браузер. Оператор *echo* для вывода структуры массива здесь не подойдет. Результат на семнадцатой строке;
*echo "
";*

$itog=str_word_count($eng); – функция *str_word_count* в данном случае подсчитает число слов в строке *$eng* (второй параметр этой функции не указан);
print_r($itog);* – выводим число слов в строке *$eng в браузер. Результат на восемнадцатой строке. Здесь можно было бы использовать и оператор *echo* и записать вместо данной строки следующее:
*echo "$itog
";*

$itog=strtok($str, ","); – функция *strtok* возвращает первую подстроку исходной строки *$str* (фраза на русском языке), отделенной от остальной части

строки запятой. Исходная строка укоротится до первой запятой. Результат на девятнадцатой строке;
*echo "$itog
";*

$path=pathinfo("C:\server\Apache2\htdocs\stroki.php");* – функция *pathinfo* создает массив (назовем его *$path) из строки (пути к некоему файлу *stroki.php*), указанной в круглых скобках;
*echo $path["dirname"]."
";* – последние 3 строки выводят в браузер название директории, где находится файл *stroki.php*, само название файла и его расширение.
*echo $path["basename"]."
";*
echo $path["extension"];

4.5. СУПЕРГЛОБАЛЬНЫЕ МАССИВЫ

Есть еще такое понятие в PHP, как суперглобальные массивы. Эти массивы особенны тем, что их переменные доступны во всех функциях, расположенных в программе PHP, т. е. они являются глобальными. Один из суперглобальных массивов **$_SERVER**. Он содержит переменные, непосредственно связанные с окружением выполнения текущего PHP-сценария. В этой главе нам понадобится только два элемента суперглобального массива $_SERVER, а именно элемент **$_SERVER["REMOTE_ADDR"]**. Он определяет IP-адрес посетителя, пришедшего на страницу.

$_SERVER["PHP_SELF"] – этот элемент массива возвращает путь к странице, на которой находится посетитель. На практике (для построения того же счетчика) полный путь к странице нам не нужен. Нам понадобится только название страницы, причем без расширения.

4.6. ДРУГИЕ ФУНКЦИИ PHP, КОТОРЫЕ ПОНАДОБЯТСЯ ДЛЯ СОЗДАНИЯ СЧЕТЧИКА

Функция **basename($filename, ".ras")** выделяет из полного пути к файлу (*$filename*) название этого файла с расширением. Второй необязательный параметр указывает на расширение файла, и если он правильно указан, данная функция возвратит название файла без расширения.

Для формирования страниц нам понадобятся функции для работы со временем. Функция **time()** выводит многозначное число, обозначающее число секунд, прошедшее с начала эпохи Unix, т. е. с нуля часов первого января 1970 г. до настоящего времени.

И, наконец, нам понадобится функция **date("time")** – выводит дату и время. Параметр *time* определяет строку форматирования, т. е. в каком виде выводить дату и время. Некоторые значения параметра *time* указаны в табл. 4.1.

Таблица 4.1. Значения параметра time

Символ	*Описание*	*Возвращаемое значение*
d	День месяца, 2 цифры	от 01 до 31
D	Название дня недели (сокращенное, 3 символа)	От Mon до Sun
F	Название месяца	От January от December
g	Часы в 12-часовом формате	От 1 до 12
G	Часы в 24-часовом формате	От 0 до 23
h	Часы в 12-часовом формате с ведущими нулями	От 01 до 12
H	Часы в 24-часовом формате с ведущими нулями	От 00 до 23
i	Минуты с ведущими нулями	От 00 до 59
j	День месяца	От 1 до 31
l	Полное название дня недели	От Sunday до Saturday
m	Порядковый номер месяца с ведущими нулями	От 01 до 12
M	Название месяца (сокращенное, 3 символа)	От Jan до Dec
n	Порядковый номер месяца без ведущих нулей	От 1 до 12
s	Секунды с ведущими нулями	От 00 до 59
w	Порядковый номер дня недели	От 0 (воскресенье) до 6 (суббота)
Y	Порядковый номер года (4 цифры)	Например: 2008
y	Порядковый номер года (2 цифры)	Например: 08

Например, следующий сценарий:

```
<?php
echo date("d.m.Y H.i");
?>
```

выведет в окно браузера текущий день (d), месяц (m), год (Y), разделенные точкой. Затем текущий час (H) и минуту(i).

4.7. СЧЕТЧИК ПОСЕТИТЕЛЕЙ САЙТА

Итак, приступим, наконец, к практическому использованию языка PHP для создания сценариев. Создадим счетчик посещений для страниц нашего сайта, который будет выводить в браузер общее число посещений какой-либо страницы сайта, число посещений за текущие сутки и число посетителей с разными IP-адресами (уникальных посетителей). Разберите все варианты

кода счетчика, так как это поможет вам лучше усвоить язык PHP. Сначала разберитесь во всех строчках кода (разбор кода сразу после листинга), а потом попробуйте самостоятельно набрать код в PHP-редакторе.

Сначала в корневой папке *hdocs* создайте подпапку под названием *chetchik*, где будут храниться текстовые файлы с данными для счетчика. Теперь создайте в PHP-редакторе новый файл и сохраните его в корневой папке *hdocs* под названием *chetchik.php*.

Первый вариант кода

Листинг 4.6 (файл chetchik.php)

```
<?php
$data=date("d.m");

$stranica=substr(basename($_SERVER["PHP_SELF"]), 0, -4);//эта строчка здесь
не нужна! Она должна быть на той странице, где будет установлен счетчик!

$ip=$_SERVER["REMOTE_ADDR"];
$file="chetchik/chetchik".$stranica.".txt";
$file2="chetchik/today".$stranica.".txt";
$file3="chetchik/data".$stranica.".txt";
$file4="chetchik/ip".$stranica.".txt";

if (!file_exists($file))
{
 $chetcik=fopen($file, "w+");
 fwrite($chetcik, "1");
 fclose($chetcik);
 $read=1;

 $chetcik=fopen($file2, "w+");
 fwrite($chetcik, "1");
 fclose($chetcik);
 $read2=1;

 $chetcik=fopen($file3, "w+");
 fwrite($chetcik, $data);
 fclose($chetcik);

 $chetcik=fopen($file4, "w+");
 fwrite($chetcik, $ip."\n");
 fclose($chetcik);
 $ipkolich=1;
}
```

```
else
{
 $chetcik=fopen($file, "r");
 $read=fread($chetcik, 100);
 fclose($chetcik);
 $read++;
 $chetcik=fopen($file, "w+");
 flock($chetcik, LOCK_EX);
 fwrite($chetcik, $read);
 flock($chetcik, LOCK_UN);
 fclose($chetcik);
 $data2=fopen($file3, "r");
 $read3=fread($data2, 100);
 fclose($data2);

 if($read3!=$data)
 {
  $data2=fopen($file3, "w+");
  flock($data2, LOCK_EX);
  fwrite($data2, $data);
  flock($data2, LOCK_UN);
  fclose($data2);
  $today=fopen($file2, "w+");
  flock($today, LOCK_EX);
  fwrite($today, "1");
  flock($today, LOCK_UN);
  $read2=1;
  fclose($today);
 }
 else
 {
  $today=fopen($file2, "r");
  $read2=fread($today, 100);
  fclose($today);
  $read2++;
  $today=fopen($file2, "w+");
  flock($today, LOCK_EX);
  fwrite($today, $read2);
  flock($today, LOCK_UN);
  fclose($today);
 }
```

```
$ip2=file($file4);
$ipkolich=count($ip2);
if(in_array($ip."\n",$ip2)==false)
{
 $ipopen=fopen($file4, "a");
 flock($ipopen, LOCK_EX);
 fwrite($ipopen, $ip."\n");
 flock($ipopen, LOCK_UN);
 $ipkolich++;
 fclose($ipopen);
 }
}
echo "<table border=2 id=lolo4 bordercolor=orange bgcolor=lime><tr>
<td colspan=2 align=center>Посещаемость</td></tr>
<tr><td align=center>Всего</td><td align=center>Сегодня</td></tr>
<tr><td align=center>$read</td><td align=center>$read2</td></tr>
<tr><td colspan=2>Уник.ip: $ipkolich</td></tr></table>";
?>
```

Разберем каждую строчку этого кода:

$data=date("d.m") – в переменную *$data* записываем текущую дату и месяц;

$ip=$_SERVER["REMOTE_ADDR"] – в переменную *$ip* записываем ip пользователя, просматривающего данную страницу;

$file="chetchik/chetchik".$stranica.".txt" – в переменную *$file* записываем строку, содержащую путь к текстовому файлу, где у нас будет записано общее число посещений. Все текстовые файлы, где будут храниться данные счетчика, мы, как я уже отметил, будем помещать в созданную вами папку *chetchik*. Сам текстовый файл для хранения данных об общем числе посещений будет называться *chetchik$stranica.txt*. В переменной *$stranica* будет находиться название той страницы, куда будет помещен счетчик. Это делается, чтобы различать данные счетчиков с разных страниц. Например, если web-страница, куда будет помещен счетчик, имеет название *index.php*, то название текстового файла, где будет фиксироваться общее число посещений данной страницы, будет называться *chetchikindex.txt*. А для страницы с названием *video.php*, текстовый файл будет уже иметь название *chetchikvideo.txt*. То есть по названию текстовых файлов мы сможем определить, данные с каких страниц там содержатся. Наш счетчик получился универсальным. Достаточно в начале кода каждой страницы, где вы хотите разместить счетчик, указать переменную $stranica, а точнее записать:

$stranica=substr(basename($_SERVER["PHP_SELF"]), 0, -4);

Здесь элемент глобальной переменной *$_SERVER["PHP_SELF"]* содержит полный путь на сервере к текущей странице, где размещена данная строчка. Далее при помощи функции *basename* от полного пути остается только название данной страницы с расширением, например *index.php*. И, наконец, функция *substr* удаляет из названия четыре последних символа (точку и расширение *php*). Все данные функции для работы со строками были рассмотрены выше;

$file2="chetchik/today".$stranica.".txt" – аналогично в переменную *$file2* записываем строку, содержащую путь к текстовому файлу, где будут храниться данные о том, сколько было посетителей на сегодняшний день;

$file3="chetchik/data".$stranica.".txt" – в переменную *$file3* записываем строку, содержащую путь к текстовому файлу, где будет храниться текущий день и месяц. Зачем это надо, я объясню позже;

$file4="chetchik/ip".$stranica.".txt" – наконец, в переменную *$file4* записываем строку, содержащую путь к текстовому файлу, где будут храниться IP-адреса посетителей страницы.

Итак, каждый счетчик на каждой странице будет иметь по 4 текстовых файла с данными. Эти данные будут потом считываться и выводиться в окно браузера.

Далее идет условный оператор *if*, который проверяет условие, записанное в круглых скобках, и если оно истинно, то выполняются команды в фигурных скобках. Функция *file_exists*, которую мы рассматривали выше, проверяет существование файла, который указан в качестве аргумента этой функции. А в качестве аргумента у нас переменная *$file*, куда записан путь к текстовому файлу, где хранится число, сообщающее об общей посещаемости страницы. Получается, что выражение *file_exists($file)* проверяет, существует ли файл в папке *chetchik* под названием *chetchik$stranica.txt*. Первоначально такого файла не существует. Если вашу страницу еще никто не посещал, то счетчик пуст и никаких текстовых файлов с данными о посещениях нет. Перед функцией *file_exists($file)* стоит оператор «!», который означает отрицание. В итоге получаем следующее условие:

if (!file_exists($file)) – если файла, который указан в переменной *$file*, еще не существует (данную страницу еще никто не посещал), то выполнится последовательность действий между фигурными скобками под номером 1 (от открывающейся скобки под номером 1 до закрывающейся скобки под номером 1). Скобки я пронумеровал для удобства:

1{

$chetcik=fopen($file, "w+") – мы создаем и открываем при помощи функции *fopen* новый пустой файл, который будет располагаться и иметь название, как указано в переменной *$file*. Весь этот процесс создания нового файла

с именем, указанным в переменной *$file*, мы присвоим переменной *$chetcik*. То есть в переменной *$chetcik* будет содержаться идентификатор нового созданного нами файла и открытого для записи. Почти во всех функциях для работы с файлами, в качестве аргумента используется именно идентификатор файла, а не название самого файла;

fwrite($chetcik, "1") – в этот пустой текстовый файл, идентификатор которого содержится в переменной *$chetcik*, при помощи функции *fwrite* записывается число 1. Если первый посетитель уже зашел на страницу, то число посещений уже не нулевое и в нашем новом текстовом файле, где указывается общее число посещений, будет записано число 1. Итак, если страница, где расположен счетчик, будет называться *index.php* и ее посетил первый посетитель, то после выполнения двух предыдущих строчек кода в папке *chetchik* будет образован новый текстовый файл под названием *chetchikindex.txt*. В этот файл будет записано число 1;

fclose($chetcik) – закрываем созданный нами файл, точнее, его идентификатор при помощи функции *fclose*. После этой команды переменная *$chetcik* будет пуста;

$read=1 – вводим переменную *$read* и присваиваем ей значение 1. При первом посещении значение этой переменной, обозначающее количество посещений, будет выведено в браузер.

Далее аналогично мы создаем и открываем опять при помощи функции *fopen* новый пустой файл, который будет располагаться и иметь название, как указано в переменной *$file2*. Весь этот процесс создания нового файла с именем, указанным в переменной *$file2*, мы снова можем присвоить переменной *$chetcik*. Можно, конечно, выбрать другую переменную, но я не стал этого делать. Одной и той же переменной можно присвоить другое новое значение, при этом старое значение аннулируется. То есть переменная *$chetcik* будет уже идентификатором для нового созданного нами файла *todayindex.txt*, и этот файл будет открыт для записи. В этом файле, как было указано выше, будет содержаться количество посещений текущего дня:

$chetcik=fopen($file2, "w+") – создаем и открываем новый текстовый файл для учета количества посещений текущих суток;

fwrite($chetcik, "1") – записываем в этот файл значение 1;

fclose($chetcik) – закрываем файл (его идентификатор);

$read2=1 – вводим переменную *$read2* и присваиваем ей значение 1. При первом посещении, значение этой переменной, обозначающее количество посещений за текущие сутки, будет выведено в браузер;

$chetcik=fopen($file3, "w+") – аналогично создаем и открываем новый текстовый файл для учета текущей даты посещения;

fwrite($chetcik, $data) – записываем в этот файл текущую дату, которая указана в переменной *$data*;

fclose($chetcik) – закрываем файл (его идентификатор);

$chetcik=fopen($file4, "w+") – аналогично создаем и открываем новый текстовый файл для учета IP-адресов посетителей;

fwrite($chetcik, $ip."\n")* – записываем (используя функцию *fwrite*) в этот файл IP-адрес первого посетителя, который хранится в переменной *$ip. Символ «*\n*» обозначает, если вы помните, перевод строки. После выполнения этой команды в созданном нами текстовом файле *ipindex.txt* (счетчик, по предположению, мы разместили на странице index.php) будет записан IP-адрес первого посетителя и осуществлен перевод строки. Строку мы переводим для того, чтобы следующий IP-адрес следующего посетителя начинался с новой строки. Это необходимо для удобного подсчета общего количества адресов, т. е. оно будет равно числу строк в этом текстовом файле. Это число и будет выводиться в браузер;

fclose($chetcik) – закрываем файл (его идентификатор);

$ipkolich=1 – вводим переменную *$ipkolich* и присваиваем ей значение 1. При первом посещении, значение этой переменной, обозначающее количество посетителей с различными IP-адресами, будет выведено в браузер;

1} – мы закрыли фигурную скобку.

Итак, весь этот приведенный выше код в фигурных скобках под номером 1 будет выполнен только при первом посещении страницы, т. е. всего один – единственный раз! Больше он выполняться не будет, поскольку при повторном посещении страницы условие в круглых скобках после условного оператора *if (!file_exists($file))* будет ложным, так как файл, указанный в переменной *$file*, уже будет существовать. Он был создан при первом посещении страницы. А если условие ложно, то будет выполнен только код, который расположен ниже в фигурных скобках под номером 2, после оператора *else* (иначе):

else – если условие ложно и текстовый файл *chetchikindex.txt* уже существует (посетитель уже не первый), то выполнится код ниже за фигурной скобкой;

2{

$chetcik=fopen($file, "r") – при помощи функции *fopen* открываем текстовый файл, название которого (точнее, путь к нему) находится в переменной *$file*, а именно, если счетчик расположен на странице *index.php*, *chetchik / chetchikindex.txt*. Этот файл уже существует, так как был создан программно при первом посещении страницы. Файл открываем только для чтения, поскольку второй аргумент функции *fopen* имеет значение «*r*». Переменная *$chetcik* будет являться идентификатором этого файла;

$read=fread($chetcik, 100) – при помощи функции *fread* мы считываем данные из открытого выше файла, идентификатор которого *$chetcik*. Второй аргумент функции указывает количество считываемых байтов. Для считывания одного числа, обозначающего общее число посещений страницы, достаточно и 100 байтов. В нашем случае после первого посещения страницы в этом текстовом файле будет находиться число 1. После выполнения данной команды число это будет считано и присвоено переменной *$read*, т. е. *$read=1*;

fclose($chetcik) – закрываем файл;

$read++* – увеличиваем считанное из файла значение на единицу. При посещении второго посетителя *$read=2. Теперь это новое значение нужно сохранить;

$chetcik=fopen($file, "w+") – вновь открываем файл *chetchik / chetchikindex.txt*, но уже для записи, так как второй аргумент функции *fopen* имеет значение «*w+*». При этом старая информация в файле стирается;

flock($chetcik, LOCK_EX) – блокируем файл перед записью. Для чего это нужно, сказано выше, в разд. 4.3;

fwrite($chetcik, $read) – с помощью функции *fwrite* записываем в файл, идентификатор которого *$chetcik*, значение, находящееся в переменной *$read*. Например, при втором посещении страницы в этот файл будет записана цифра 2;

flock($chetcik, LOCK_UN) – после записи отключаем блокировку файла;

fclose($chetcik) – закрываем файл, в котором уже будет новое значение.

Итак, при каждом посещении страницы значение переменной *$read* будет увеличиваться на 1 и перезаписываться в текстовый файл, указанный в переменной *$file*.

В текстовом файле, указанном в переменной *$file2* мы, как сказано выше, будем сохранять число посещений за текущие сутки. Это число должно так же увеличиваться на единицу при каждом посещении страницы. Однако при наступлении новых суток когда изменится дата, число при очередном посещении страницы должно снова стать равным 1.

Чтобы контролировать дату посещения, мы и создали текстовый файл, который указан в переменной *$file3*. Если код этого счетчика будет расположен на странице *index.php*, то текстовый файл, где записана дата посещения страницы, будет расположен в каталоге *chetchik* и будет называться *dataindex.txt*:

$data2=fopen($file3, "r") – открываем текстовый файл с датой;

$read3=fread($data2, 100) – считываем эту дату и записываем ее в персменную *$read3*;

fclose($data2) – закрываем файл с датой.

Далее идет проверка условия. Мы сравниваем дату, которая была записана в текстовом файле *dataindex.txt* с текущей датой. А текущая дата у нас находится в переменной *$data*. Смысл сравнения понятен. Пока посетители посещают вашу страницу в течение одних суток, дата в текстовом файле и текущая дата совпадают. Как только наступят новые сутки, текущая дата станет другой. Число посещений за сутки снова должно «сброситься». А вместо старой даты в текстовый файл *dataindex.txt* нужно будет записать новую дату:

if($read3!=$data) – если дата, записанная в файле *dataindex.txt*, и текущая дата не совпадают (наступили новые сутки), то выполнится блок команд в фигурных скобках под номером 3;

3{

$data2=fopen($file3, "w+") – открываем текстовый файл с датой, стирая содержимое, так как второй аргумент функции *fopen* у нас «w+». Прошедшая дата нам уже не нужна. Идентификатором файла будет являться переменная *$data2*;

flock($data2, LOCK_EX) – блокируем этот файл перед записыванием;

fwrite($data2, $data) – записываем в файл новую текущую дату;

flock($data2, LOCK_UN) – разблокируем файл;

fclose($data2) – закрываем файл с записанной новой датой;

$today=fopen($file2, "w+") – открываем текстовый файл, где записано число посещений за текущие сутки, стирая содержимое, ибо число посещений за прошедшие сутки нам уже не нужно;

flock($today, LOCK_EX) – блокируем файл перед записыванием;

fwrite($today, "1") – записываем в файл единицу;

flock($today, LOCK_UN) – разблокируем файл;

$read2=1 – переменной *$read2* присваиваем значение 1. При первом посещении значение этой переменной, обозначающее количество посещений за текущие сутки, будет выведено в браузер;

fclose($today) – закрываем файл;

3} – закрываем скобку.

Если условие *if($read3!=$data)* ложно, т. е. дата в текстовом файле *dataindex.txt* и текущая дата совпадают, значит, сутки не поменялись. Мы просто увеличим число посещений за текущие сутки на 1:

else – если текущая дата прежняя, то выполнится блок программ между фигурными скобками под номером 4, а тот блок программ, который был выше (между фигурными скобками под номером 3), будет проигнорирован;

4{

$today=fopen($file2, "r") – открываем текстовый файл для чтения, где хранится число посещений страницы за текущие сутки. Путь к этому файлу указан

в переменной *$file2*. В переменной *$today* будет содержаться идентификатор этого файла;

$read2=fread($today, 100) – считываем содержимое открытого нами файла и записываем его в переменную *$read2*;

fclose($today) – закрываем файл;

$read2++* – увеличиваем переменную *$read2 на единицу;

$today=fopen($file2, "w+") – снова открываем текстовый файл, где хранится число посещений страницы за текущие сутки, но уже для записи. При этом старая информация в файле уничтожается;

flock($today, LOCK_EX) – блокируем файл перед записью;

fwrite($today, $read2) – записываем в файл уже новое число, которое у нас в переменной $read2;

flock($today, LOCK_UN) – разблокируем файл;

fclose($today) – закрываем файл;

4} – закрываем скобкой блок программ.

Теперь займемся текстовым файлом, где у нас хранятся IP-адреса. Путь к этому файлу указан в переменной *$file4*. Если страница, где будет расположен этот код, называется *index.php*, то сам текстовый файл будет называться *ipindex.txt*. Каждый IP-адрес будет располагаться на новой строке, т. е. после записи очередного адреса должен следовать символ перевода строки «*\n*»:

$ip2=file($file4) – при помощи функции *file* превращаем содержимое файла *ipindex.txt* в массив строк. Каждый элемент массива является строкой, а каждая строка содержит какой-нибудь один IP-адрес. Массив записываем в переменную *$ip2*;

$ipkolich=count($ip2) – функция *count* определяет число элементов в массиве *$ip2*. В нашем случае это будет число строк в файле, т. е. число IP-адресов. Именно это число, которое сохраним в переменной *$ipkolich*, мы выведем потом в окно браузера.

Посетитель с одного IP-адреса может посетить вашу страницу неоднократно. Нам нужно сделать так, чтобы IP-адрес записывался в текстовый файл только при первом посещении определенного посетителя. При дальнейшем посещении страницы данным посетителем его IP-адрес записываться не должен, так как он у нас уже есть. Таким образом, в окно браузера будет выведено число уникальных, неповторяющихся посетителей вашей web-страницы. Если конкретный посетитель попал на вашу страницу впервые, то его адреса не будет в текстовом файле *ipindex.txt* и его надо записать:

if(in_array($ip."\n",$ip2)==false) – здесь проверяется условие. Функция *in_array* осуществляет поиск элемента (первый аргумент функции) в массиве

$ip2 (второй аргумент функции). Первый аргумент функции *in_array* у нас *$ip."\n"*. В переменной *$ip* содержится элемент суперглобального массива *$_SERVER* (см. начало кода), определяющий IP-адрес посетителя страницы. Далее при помощи символа конкатенации (точки) ставим символ перевода строки «*\n*». Таким образом, функция *in_array* будет искать в массиве строк файла *ipindex.txt* строку с записью IP-адреса посетителя и перевода строки. Если данный IP-адрес будет найден, то функция возвратит *True*, в противном случае возвратит *False*. В общем, данное условие звучит так: если данного IP-адреса нет в массиве строк *$ip2* (функция *in_array* возвращает *False*), то выполнится блок команд в фигурных скобках под номером 5;

5{

$ipopen=fopen($file4, "a") – открываем текстовый файл с IP-адресами для записи, причем содержимое его не стираем, а будем записывать данные после последнего символа (вторым аргументом функции *fopen* является «*a*»). А последним символом у нас является символ перевода строки после прошлой записи, когда страницу посетил первый посетитель. Короче, запись нового IP-адреса начнется с новой строки;

flock($ipopen, LOCK_EX) – блокируем текстовый файл перед записью;

fwrite($ipopen, $ip."\n") – записываем в этот файл очередной IP-адрес и снова ставим курсор на новую строку, используя символ «*\n*»;

flock($ipopen, LOCK_UN) – разблокируем файл;

$ipkolich++ – увеличиваем число ip посетителей на единицу. Поскольку в текстовый файл *ipindex.txt* мы записали новую строку, то число элементов в массиве строк увеличится на единицу;

fclose($ipopen) – закрываем файл;

5} – закрываем фигурную скобку. Весь этот блок команд, приведенный выше между фигурными скобками под номером 5, будет выполняться тогда и только тогда, когда на страницу зашел новый посетитель, ранее не посещавший вашу страницу. Если же посетитель заходит не впервые (условие *if(in_array($ip."\n",$ip2)==false)* ложно), весь этот блок команд будет проигнорирован;

2} – наконец, закрываем большой блок команд в фигурных скобках под номером 2 (он начинался после оператора *else*) нашего первого условия *if(!file_exists($file))*, которое проверяло, посещал ли вообще кто-нибудь вашу страницу.

Код счетчика практически написан и разобран. Осталось теперь вывести данные о посещении страницы в окно браузера. Создадим маленькую табличку при помощи HTML-тега *<table>*. Толщину линий таблицы сделаем в 2px (*border=2*), цвет линий *bordercolor* установим равным *orange*, цвет фо-

на *bgcolor* установим равным *lime*. Вообще, как создается таблица, описано выше в гл. 3. Вывод таблицы в браузер осуществляется при помощи известного вам оператора *echo*. Вся таблица от тега *<table>* до тега *</table>* обязательно заключается в кавычки, ибо любые HTML-теги со всем их содержимым в PHP-коде перед оператором *echo* должны быть в кавычках:

echo "<table border=2 bordercolor=orange bgcolor=lime><tr> – тег *<tr>* создает первую верхнюю строчку таблицы. Это будет шапка нашей таблицы со словом «*Посещаемость*»;

<td colspan=2 align=center>Посещаемость</td></tr> – эта верхняя строка будет занимать сразу 2 столбца (2 ячейки), так как в теге, создающей ячейку в строке *<td>*, мы указали *colspan=2*. Текст в ячейке устанавливаем по центру, так как есть параметр *align=center*. Далее закрываем ячейку и строку при помощи операторов *</td>* и *</tr>*;

<tr><td align=center>Всего</td><td align=center>Сегодня</td></tr> – вторая строка состоит из двух ячеек, со словами «Всего» и «Сегодня»;

<tr><td align=center>$read</td><td align=center>$read2</td></tr> – третья строка тоже состоит из двух ячеек. В первую ячейку под ячейкой со словом «*Всего*» записываем переменную *$read*. В ней, если вы внимательно разобрались с кодом, находится общее число посещений страницы. Во вторую ячейку под ячейкой со словом «*Сегодня*» записываем переменную *$read2*. В этой переменной находится число посещений страницы за текущие сутки;

<tr><td colspan=2>Уник.ip: $ipkolich</td></tr></table>"* – ну и, наконец, последняя строка, как и первая, состоит из одной большой двойной ячейки, в которой после слов «*Уник.ip*» записывается переменная *$ipkolich, где хранится число уникальных посетителей, т. е. посетителей с разными IP-адресами.

Вот и все! Я старался как можно понятнее «разжевать» вам каждую строчку кода. Если не совсем поняли, прочтите главу «Основы PHP5» и разбор этого кода еще раз. Код счетчика затрагивает такие важные темы в php, как работа со строками, создание и открытие файла, чтение из файла и запись в файл. Знание и понимание этих тем – очень важно для изучения языка php. Вы не сможете создать ни одной путной программы для своего сайта, не разобравшись в коде и принципе работы счетчика. Поэтому разберитесь с этим внимательно. Код набирайте сами. Постарайтесь набирать его как можно реже, смотря в книгу на листинг этого кода. Постарайтесь понять смысл каждой строчки кода. Только так можно научиться.

Итак, скрипт счетчика готов. Вставим его в нашу главную страницу *index.php* при помощи оператора *include*. Запишите между тегами *<?php* и *?>* следующую строку:

include "chetchik.php";

В файле *chetchik.php* находится, как вы помните, код нашего счетчика (листинг 4.6), который мы подробно разобрали. После выполнения этой команды код счетчика вставится в нашу страницу *index.php*. Но это еще не все. Нам нужно указать в коде страницы *index.php* переменную *$stranica*. Перед тем как вставить счетчик (до команды *include* !), определим переменную $stranica:

```
$stranica=substr(basename($_SERVER["PHP_SELF"]), 0, -4);
```

Эту строчку мы уже разбирали выше. Здесь указывается элемент суперглобального массива *$_SERVER["PHP_SELF"]*, где хранится путь к текущей странице. На странице *index.php* элемент *$_SERVER["PHP_SELF"]* у меня, например, будет иметь такое значение: «*/MyBook/index.php*», так как все PHP-файлы для данной книги я сохраняю в папке *MyBook*, которая, в свою очередь, находится в рабочей папке сервера *htdocs*. Функция *basename* урезает путь к файлу и оставляет только его название с расширением, т. е. у нас останется *index.php*. Далее функция *substr* урезает из названия четыре последних символа, оставляя только *index*. В итоге переменной *$stranica* будет присвоена строка «*index*». Теперь переменная *$stranica* у нас определена и ее можно использовать в коде счетчика.

Итак, вернемся к листингу 3.6, где находится программный код страницы index.php. Внизу, после строчки *</tr></table>*, вставьте следующие строки:

```
<?php
$stranica=substr(basename($_SERVER["PHP_SELF"]), 0, -4);
include "chetchik.php";
?>
```

Мы создали универсальный счетчик. Достаточно на каждую страницу вашего сайта вставить приведенные выше две строчки, чтобы на всех страницах было изображение таблички с данными о посещении конкретной страницы сайта. В созданной вами папке *chetchik* будут храниться текстовые файлы с данными счетчиков для всех страниц. Для каждой страницы в папке chetchik будут созданы 4 текстовых файла, с которых и будут считываться эти данные для счетчика.

Запустите браузер. Внизу слева на странице вы увидите маленькую табличку с данными о посещении. При первом запуске данные в таблице будут такими, как на рис. 4.1.

Зайдите в папку *chetchik*. Вы увидите там 4 текстовых файла, которые программа создала сама (рис. 4.2).

В файле *chetchikindex* записано общее число посещений страницы *index.php*. В файле *todayindex* – число посещений за текущие сутки. В файле *ipindex* будут хранится IP-адреса. Откройте этот файл. Там вы можете увидеть пока единственный IP-адрес, после которого стоит маленький прямоугольник. Это указывает на перевод строки. И, наконец, в файле *dataindex* вы увидите текущее число и месяц.

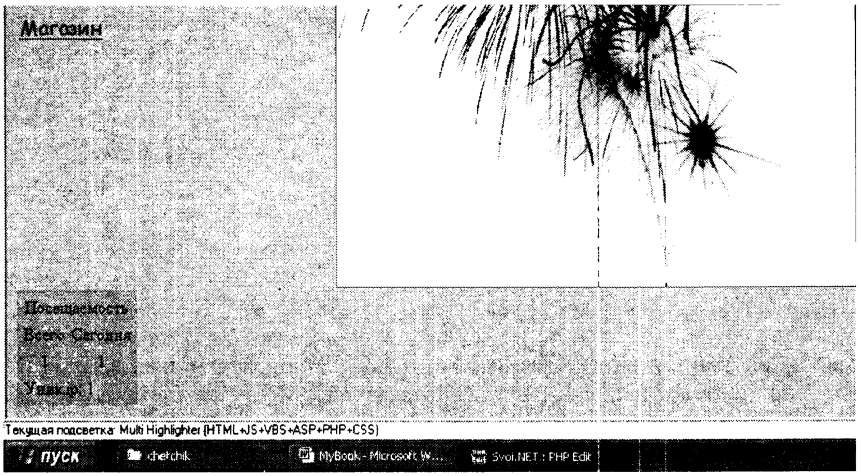


Рис. 4.1

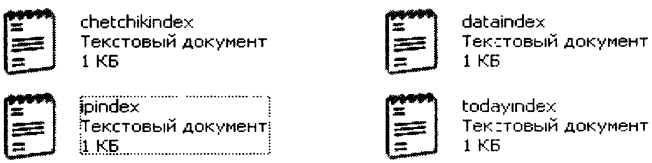


Рис. 4.2

Запустите заново файл *index.php* в браузере или нажмите на панели инструментов значок «обновить». Вы увидите, что общее число посещений и число посещений за сегодняшний день увеличилось на единицу. Число уникальных ip осталось без изменения, и это понятно. Вы пока единственный уникальный посетитель своей странички. Нажмите значок обновления страницы еще несколько раз. При каждом обновлении числа под словами «*Всего*» и «*Сегодня*» будут увеличиваться на единицу.

Теперь измените дату на своем компьютере. Поставьте, например, завтрашнее число. Убедитесь потом, что дата на вашем компьютере действительно поменялась. Опять обновите страницу. Вы увидите, что общее число посещений продолжит увеличиваться, а число посещений за текущие сутки опять сбросится до единицы. Все, не забудьте снова установить у себя сегодняшнее число.

Второй вариант кода

Первый вариант кода не очень рационален, и я привел его для того, чтобы вы могли лучше понять, как создавать текстовые файлы, считывать из них данные и выводить эти данные в браузер. Наш счетчик использовал 4 тексто-

вых файла. Напрашивается вопрос, а нельзя ли все данные для счетчика поместить в два текстовых файла? Не только можно, но и нужно. Программный код при этом сократится, так как нам придется открывать и закрывать всего два файла вместо четырех. В одном текстовом файле у нас будет массив строк, состоящий из трех элементов: общее число посетителей, текущая дата, число посетителей за текущие сутки. Другой файл будет содержать массив строк, состоящий из IP-адресов посетителей. В листинге 4.7 приведен код второго варианта счетчика. Постарайтесь разобраться в этом коде самостоятельно. Если что-то непонятно, после листинга я вкратце разобрал некоторые строчки кода. В PHP-редакторе создайте новый файл, где будет располагаться второй вариант кода и сохраните его в рабочей папке hdocs под названием *chetchik2.php*.

Листинг 4.7 (файл chetchik2.php)

```
<?php
$data=date("d.m");
$ip=$_SERVER["REMOTE_ADDR"];
$file1="chetchik/ip".$stranica.".txt";
$file2="chetchik/count".$stranica.".txt";
if (!file_exists($file2))
{
 $vsego=1;
 $segodna=1;
 $ipkolich=1;
 $count=$vsego."\n".$data."\n".$segodna;
 $chetcik=fopen($file2, "w+");
 fwrite($chetcik, $count);
 fclose($chetcik);
 $ip2=fopen($file1, "w+");
 fwrite($ip2, $ip."\n");
 fclose($ip2);
}
else
{
 $file=file($file2);
 foreach($file as $stroka)
 {
  $massive[]=$stroka;
 }
```

```
$vsego=(int)$massive[0];
$data2=(float)$massive[1];
$segodna=(int)$massive[2];
$vsego++;
if($data2!=$data)
{
$segodna=1;
}
else
{
 $segodna++;
}
$count2=$vsego."\n".$data."\n".$segodna;
$chetcik=fopen($file2, "w+");
flock($chetcik, LOCK_EX);
fwrite($chetcik, $count2);
flock($chetcik, LOCK_UN);
fclose($chetcik);
$ip2=file($file1);
$ipkolich=count($ip2);
if(in_array($ip."\n",$ip2)==false)
{
$ipopen=fopen($file1, "a");
flock($ipopen, LOCK_EX);
fwrite($ipopen, $ip."\n");
flock($ipopen, LOCK_UN);
$ipkolich++;
fclose($ipopen);
}
}
echo "<table border=2 bordercolor=orange bgcolor=lime><tr>
<td colspan=2 align=center>Посещаемость</td></tr>
<tr><td align=center>Всего</td><td align=center>Сегодня</td></tr>
<tr><td align=center>$vsego</td><td align=center>$segodna</td></tr>
<tr><td colspan=2>Уник.ip: $ipkolich</td></tr></table>";
?>
```

Многое из этого кода вам должно быть понятно, поэтому больших комментариев ко многим строкам делать не буду:

```
<?php
$data=date("d.m");
$ip=$_SERVER["REMOTE_ADDR"];
```

$file1="chetchik/ip".$stranica.".txt" – в переменной *$file1* будет находиться путь к текстовому файлу, в который записываются IP-адреса;

$file2="chetchik/count".$stranica.".txt" – в переменной *$file2* будет находиться путь к текстовому файлу, в который запишется общее число посетителей, текущая дата, число посетителей за текущие сутки;

if (!file_exists($file2)) – если текстовых файлов (хотя бы одного из них) еще не существует, т. е. ваша страница посещается впервые, то создаем их в блоке команд между фигурными скобками под номером 1;

1{

$vsego=1 – создаем переменную *$vsego*, где будет храниться общее число посетителей, и присваиваем ей начальное значение единица;

$segodna=1 – создаем переменную *$segodna*, где будет храниться число посетителей за текущие сутки, и присваиваем ей начальное значение единица;

$ipkolich=1 – создаем переменную *$ipkolich*, где будет храниться число уникальных IP-адресов, и присваиваем ей начальное значение единица;

$count=$vsego."\n".$data."\n".$segodna – переменные *$vsego, $data и $segodna* записываем в одну общую переменную *$count*, разделяя их символом перевода строки;

$chetcik=fopen($file2, "w+") – создаем и открываем новый текстовый файл для хранения данных;

fwrite($chetcik, $count) – записываем туда значение переменной *$count*, которая содержит 3 строки данных – число посещений, общее и за текущие сутки, и текущую дату в средней строке;

```
fclose($chetcik);
```

$ip2=fopen($file1, "w+") – создаем текстовый файл для хранения адресов;

fwrite($ip2, $ip."\n") – записываем туда IP-адрес первого посетителя и переводим строку;

```
fclose($ip2);
```

1}

else – если визит на страницу не первый и текстовые файлы уже существуют, то выполняется другой блок команд ниже, между фигурными скобками под номером 2;

2{

$file=file($file2) – превращаем данные в текстовом файле в массив строк;

foreach($file as $stroka) – функция *foreach* пробегает массив строк *$file*, присваивая значение элемента этого массива (содержимое строки) переменной *$stroka*. Сначала переменной *$stroka* присваивается содержимое первой строки (у нас это общее число посещений), далее выполняются команды ниже в фигурных скобках (у нас там всего одна команда). Потом переменная *$stroka* опустошается и ей уже присваивается содержимое второй строки (там у нас текущая дата), далее опять выполняется команда за фигурными скобками. Весь цикл будет выполняться между фигурными скобками под номером 3 столько раз, сколько всего элементов в массиве. В нашем случае 3 раза;

3{

$massive[]=$stroka – при каждом выполнении этого цикла. мы будем сохранять содержимое переменной *$stroka* в массив, который назовем *$massive*. Это нужно делать, так как переменная *$stroka* обновляется при каждом цикле, и ее старое значение затирается. В нулевом элементе массива *$massive[0]* сохранится содержимое первой строки. В первом элементе *massive[1]* сохранится содержимое второй строки и т. д.;

3}

$vsego=(int)$massive[0] – переменной *$vsego* присваивается нулевой элемент массива *$massive*, т. е. общее число посещений. Это целое число, поэтому перед этим элементом массива ставим оператор *int*. Это нужно, чтобы программа «понимала» какой тип данных хранится в данном элементе массива;

$data2=(float)$massive[1] – переменной *$data2* присваивается первый элемент массива (там у нас текущая дата), т. е. текущее число и месяц, разделенные точкой. Все это можно представить как десятичное число, поэтому перед этим элементом массива ставим оператор *float*;

$segodna=(int)$massive[2] – ну и, аналогично, присваиваем переменной *$segodna* второй элемент массива, т. е. число посетителей за текущие сутки. Это тоже целое число;

$vsego++* – увеличиваем переменную *$vsego на единицу;

if($data2!=$data) – если наступили другие сутки:

4{

$segodna=1 – сбрасываем значение переменной *$segodna* до единицы;

4}

else

5{

$segodna++ – в противном случае, если дата еще не изменилась, просто увеличиваем эту переменную на единицу;

5}

$count2=$vsego."\n".$data."\n".$segodna – переменные *$vsego,$data и $segodna* записываем в одну общую переменную *$count2*, разделяя их символом перевода строки;

$chetcik=fopen($file2, "w+") – открываем файл для записи новых данных, удалив старые;

flock($chetcik, LOCK_EX);

fwrite($chetcik, $count2) – записываем в этот файл переменную *$count2*;

flock($chetcik, LOCK_UN);

fclose($chetcik);

$ip2=file($file1) – данные текстового файла, где хранятся IP-адреса превращаем в массив строк *$ip2*;

$ipkolich=count($ip2) – в переменную *$ipkolich* сохраняем число элементов массива (число строк или число IP-адресов);

if(in_array($ip."\n",$ip2)==false) – если в текстовом файле (или в массиве строк) такого IP-адреса еще нет, т. е. элемент массива, содержащий данный IP-адрес, не найден, то выполнится блок команд ниже в фигурных скобках под номером 6.

Далее все аналогично, как и в первом варианте кода, разве только название переменных, содержащих данные, имеют другие названия.

```
6{
 $ipopen=fopen($file1, "a");
 flock($ipopen, LOCK_EX);
 fwrite($ipopen, $ip."\n");
 flock($ipopen, LOCK_UN);
 $ipkolich++;
 fclose($ipopen);
6}
2}
echo "<table border=2 bordercolor=orange bgcolor=lime><tr>
<td colspan=2 align=center>Посещаемость</td></tr>
<tr><td align=center>Всего</td><td align=center>Сегодня</td></tr>
<tr><td align=center>$vsego</td><td align=center>$segodna</td></tr>
<tr><td colspan=2>Уник.ip: $ipkolich</td></tr></table>";
?>
```

В теге таблицы *<table>* вместо параметра цвета фона таблицы *bgcolor* можно задать фоновый рисунок при помощи параметра *background*, например, можно записать *background=pic1.gif*. Фоновым рисунком для нашей таблички с данными о посещении будет в этом случае рисунок *pic1.gif*. Естественно, этот рисунок должен находиться в нашей рабочей папке сервера

htdocs. Если же рисунок у вас будет храниться в отдельной папке, например *risunki* (которая, в свою очередь, должна находиться в папке *htdocs*), то в параметре background нужно указать путь к данному графическому файлу: *background=risunki/pic1.gif*. Цвет текста в таблице можно задать при помощи листа стилей. Естественно, на странице, куда вы поставите счетчик при помощи команды *include*, должен быть указан файл стилей. На нашей главной странице *index.php* в начале кода лист стилей указан:

<link type="text/css" rel="stylesheet" href="stil.css"> – см. листинг 3.4.

Лист стилей у нас в текстовом файле *stil.css*, который мы создали ранее (см. гл 3, п. 3.3). Откройте этот файл и вставьте в него новую строчку:

#lolo{font-size:12pt; color:red; font-family:Comic Sans MS}. Здесь создается идентификатор *lolo* (можете назвать любыми латинскими буквами), в котором задается размер шрифта (*font-size*), цвет текста (*color*), название шрифта (*font-family*). Можете установить эти параметры по своему усмотрению. Сохраните и закройте файл *stil.css*. Теперь в коде счетчика, в теге *<table>* допишите *id=lolo*, т. е. получится строчка:

<table border=2 bordercolor=orange bgcolor=lime id=lolo>

Запустите файл *index.php* в браузере PHP-редактора. Вы увидите, что шрифт и цвет текста в таблице счетчика изменился.

4.8. СЧЕТЧИК ПОСЕТИТЕЛЕЙ С ИСПОЛЬЗОВАНИЕМ БАЗЫ ДАННЫХ MYSQL

Все скрипты в этой книге написаны без использования баз данных. Однако, для построения своего сайта, вы можете использовать для удобства реляционные базы данных, например MySQL. Литературы по базам данных предостаточно, и вы, при желании, можете ее изучить. В этой же книге я лишь кратко ознакомлю с базами данных и приведу пример их использования для нашего счетчика посещений.

Сначала рассмотрим основные понятия:

1. Реляционная база данных – совокупность связанных таблиц, где хранятся данные.
2. Таблица состоит из полей (*field*), которые имеют имя и тип. Тип поля зависит от того, какие данные в этом поле записаны. Существуют следующие типы полей: целые, вещественные, строковые, бинарные, дата и время, перечисления и множества.
3. Запись (*record*) – это набор полей в таблице, содержащих связанную между собой информацию. Например, если в таблице имеется 3 поля строкового типа, содержащие соответственно имя, фамилию и отчество, то совокупность содержимого этих трех полей и будет называться записью.

4. Запрос (*query*) – оператор, который выбирает записи и поля из одной или нескольких таблиц согласно заданному условию.

Для написания запросов к базам данных разработан язык SQL (структурированный язык запросов). Рассмотрим несколько операторов этого языка. Создать таблицу в базе данных можно при помощи оператора CREATE. Создадим, например, таблицу, с названием Schetchik для нашего счетчика посещений. Пусть таблица будет содержать 3 поля с именами vcego (поле будет содержать общее число посещений страницы), segodna (будет содержать число посещений за текущие сутки) и chislo (будет содержать текущую дату). Тогда создание нашей таблицы на языке SQL будет выглядеть так:

```
CREATE TABLE Schetchik
(
vcego int NOT NULL,
segodna int NOT NULL,
chislo data NOT NULL,
)
```

После имени каждого поля стоит тип поля. Первые два поля (общее число посещений и число посещений за текущие сутки) будут содержать целочисленный тип данных, а именно, тип int, который может содержать числа в диапазоне от -2 147 483 648 до 2 147 483 647. Думаю, что нам этого достаточно. Третье поле (текущая дата) имеет тип data и содержит дату в формате ГГГГ-ММ-ДД (год, месяц, число). Модификатор NOT NULL указывает, что поле должно быть определено, т. е. оно не может быть пустым.

Добавить данные в созданную таблицу можно при помощи оператора INSERT. Заполним первые два поля единицей, а в третье поле впишем текущую дату:

```
INSERT INTO Shetchik
VALUES (1,1,TO_DATE(15/09/08,'YY/MM/DD'));
```

В списке VALUES перечисляются значения полей таблицы. Дату в третье поле записываем в формате год/месяц/число.

Если мы хотим изменить запись в таблице, то нужно использовать оператор UPDATE. Синтаксис этого оператора выглядит так:

```
UPDATE Имя_таблицы
SET Поле1=значение1, Поле2=Значение2...
WHERE условие;
```

После SET пишется имя поля и присваиваемое ему значение (через знак равенства). Условие WHERE – какое значение того или иного поля изменить. В нашем простом случае в каждом поле по одному значению, поэтому условие WHERE здесь не нужно. Вообще приведенные выше строчки можно записать и в одну строку:

UPDATE Имя_таблицы SET Поле1=значение1, Поле2=Значение2... WHERE условие;

Перейдем, наконец, к практике.

Практически на любом уважающем себя хостинге клиенту предоставляется возможность создания и управления базами данных. Для создания базы данных предоставляется достаточно удобный интерфейс. Например, на хостинге, где расположен мой сайт (majordomo.ru), можно легко создать базу данных (кнопка «Создать»). На рис. 4.3 видно, как я создал на хостинге базу данных с именем b19147_baza.

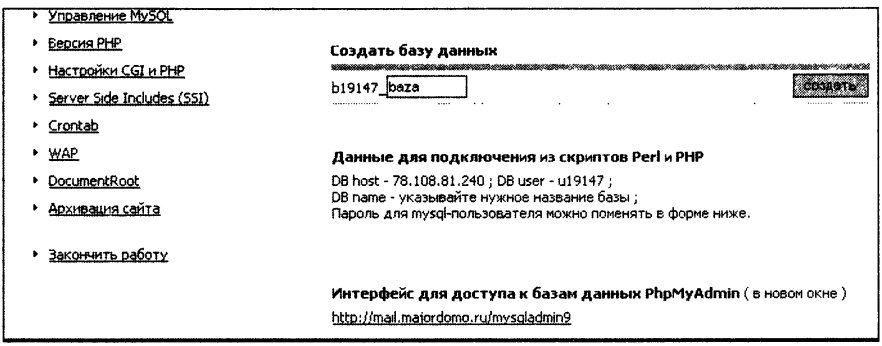


Рис. 4.3

Запомните или запишите потом данные (DB host, DB user) для подключения базы данных из скрипта PHP, который мы потом составим. Пароль создаете сами.

Вообще создать базу данных можно и используя интерфейс *PhpMyAdmin*, нажав на соответствующую ссылку (рис. 4.3). Вас попросят ввести имя (DB user) пользователя (в нашем случае это u19147) и пароль, который вы должны были создать. Только после этого вы попадете на *PhpMyAdmin* (рис. 4.4).

После создания базы данных нужно потом ее выбрать из ниспадающего списка слева. Затем надо создать таблицу в этой базе данных. Создадим таблицу с именем *Schetchik*. Количество полей сделаем, как договорились, 3 (рис. 4.5).

После создания таблицы нужно указать тип и имена полей (рис. 4.6). Пусть в первом поле у нас будет находиться число, определяющее общее число посетителей страницы сайта. Это поле с именем *vcego*. Тип поля задаем INT (целые числа). Далее задается длина числа (максимальная – 11), значение (можно ничего не ставить), значение по умолчанию поставьте 1. Аналогично для второго поля с именем *segodna* (там, если помните, будет храниться число посещений за текущие сутки) задайте те же значения. Третье поле с именем *chislo* будет содержать дату. Тип поля задаем DATE, а значением по умолчанию зададим любую дату в формате ГГГГ-ММ-ДД, например 2008-12-23.

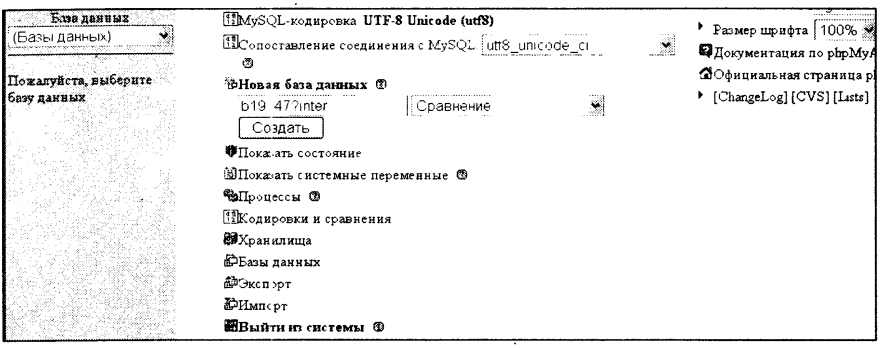


Рис. 4.4

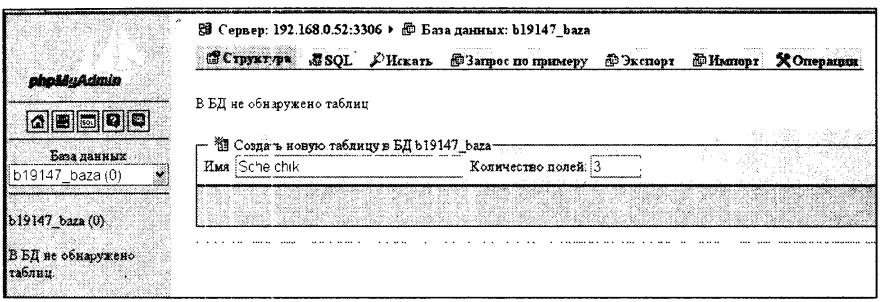


Рис. 4.5

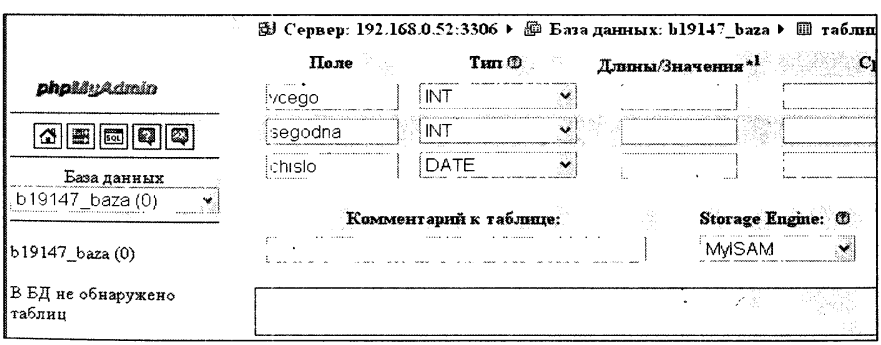


Рис. 4.6

Далее нажимаете на кнопку «Создать» (текст на кнопке может быть различным, например «Пошел» или «Вперед»). Все, поля созданы (рис. 4.7).

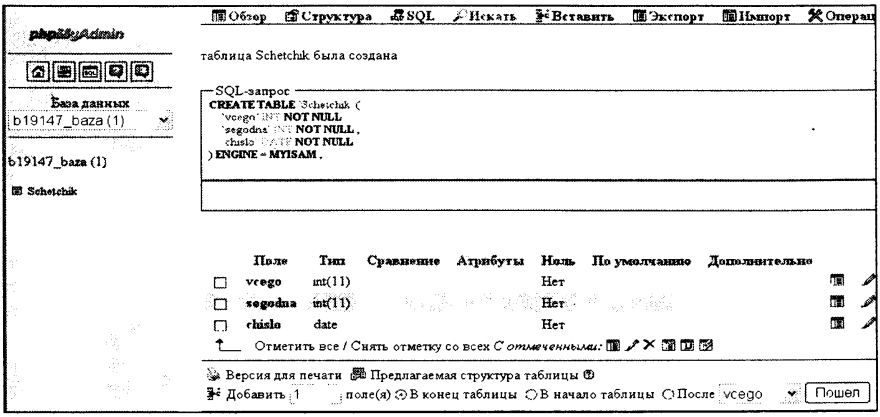


Рис. 4.7

После того, как поля созданы, можно выходить из интерфейса *PhpMyAdmin*. На других хостингах интерфейс для создания и управления баз данных может быть другим. Если вы не сможете самостоятельно создать на хостинге (где будет находиться создаваемый вами сайт) базу данных и таблицу, то попросите помощи по E-mail у хостера. На любом хостинге есть возможность написать письмо с вопросом в службу технической поддержки.

Итак, база и таблица в базе созданы. Напишем теперь скрипт счетчика, используя созданную нами базу данных.

Подключиться к серверу баз данных в PHP-сценарии можно при помощи функции *mysql_connect($Server,$User,$Password)*, где *$Server* – имя сервера базы данных, *$User* – имя пользователя базы данных, *$Password* – пароль. Имя сервера, имя пользователя (*DB host* и *DB user*) вы получите у своего хостера при создании базы данных (см. рис. 4.3). Пароль указываете тот, который вы создали при создании самой базы данных.

После подключения к серверу базы данных, нужно выбрать нужную нам базу данных. Делается это при помощи функции *mysql_select_db($name)*, где *$name* – имя базы данных, созданной вами на сервере.

После того как мы подключились к серверу и выбрали базу данных, нужно сделать запрос к этой базе данных. Делает это функция *mysql_query("Текст запроса")*. Эта функция возвращает идентификатор выполнения запроса. Например:

*$m=mysql_query("select * from Schetchik");*

В данном случае функция *mysql_query()* возвратит набор, состоящий из записей в таблице *Schetchik*. Идентификатор этого набора будет находиться в переменной *$m*. Используя данный идентификатор, можно получить нужный элемент из набора записей при помощи функции *mysql_result($m,*

$str, $stl), где *$str* – номер строки или записи в таблице (в нашем случае у нас всего одна запись), *$stl* – номер столбца или поля таблицы (у нас есть 3 поля). Например, код:

```
$w=mysql_result($m, 0, 2);
echo $w;
```

выведет в окно браузера содержимое третьего поля таблицы *Schetchik* (нумерация полей и записей начинается с нуля). У нас там, если помните, записана дата.

Получить нужный элемент из набора *$m* позволяет и функция *mysql_fetch_array($m)*. Данная функция возвращает строку в виде массива, ключами которого будут имена полей. Например, следующий код:

```
$k=mysql_fetch_array($m);
echo $k[segodna];
```

выведет в браузер содержимое поля с именем *segodna*. У нас там записана единица.

Давайте теперь изменим данные в нашей таблице. Сделать это достаточно просто:

$day=date("Y-m-d"); – в переменной $day у нас будет текущая дата.

mysql_query("UPDATE Schetchik SET vcego='2', segodna='2', chislo='$day' ");

Мы делаем запрос к текущей базе данных при помощи функции *mysql_query()* и вносим изменения в таблицу при помощи оператора UPDATE.

Составим теперь код счетчика с использованием базы данных (листинг 4.8). Счетчик будет показывать общее число посетителей и число посетителей за текущие сутки. Создайте PHP-страницу под названием, например *chetchikMySql.php*.

Листинг 4.8 (файл chetchikMySql.php).

```
<?php
if ($rc=mysql_connect("78.108.81.240","u19147","parol"))
{
 $day=date("Y-m-d");
 mysql_select_db("b19147_baza");
$tabl="Schetchik";
 $mar=mysql_query("SELECT * FROM $tabl");
 $k=mysql_fetch_array($mar);
 $k[vcego]=$k[vcego]+1;
 if($day!==$k[chislo])
```

```
{
 $k[chislo]=$day;
 $k[segodna]=1;
}
else
{
 $k[segodna]=$k[segodna]+1;
}
mysql_query("UPDATE Schetchik SET vcego='$k[vcego]', segodna='$k[segodna]',
chislo='$k[chislo]'");
echo "<table border=2 bordercolor=orange><tr>
<td colspan=2 align=center>Посещаемость</td></tr>
<tr><td align=center>Всего</td><td align=center>Сегодня</td></tr>
<tr><td align=center>$k[vcego]</td>
<td align=center>$k[segodna]</td></tr></table>";
}
else
{
echo "Error!";
}
?>
```

Разберем некоторые строчки кода:

<?php – начало PHP-кода;

if ($rc=mysql_connect("78.108.81.240","u19147","parol")) – при помощи оператора *mysql_connect* соединяемся с сервером, указывая DB host, DB user и пароль. Этому соединению присваиваем идентификатор *$rc*. При помощи условия *if* проверяем, произошло ли соединение, т. е. определена ли переменная *$rc*. Если соединение произошло успешно, выполнится весь код между фигурными скобками под номером 1;

1{

$day=date("Y-m-d"); – переменой $day присваиваем текущую дату в формате Y-m-d (четыре цифры года, две цифры месяца и две цифры числа месяца);

mysql_select_db("b19147_baza"); – выбираем созданную нами базу данных;

$tabl="Schetchik";* – переменной *$tabl присваиваем название созданной нами таблицы;

*$mar=mysql_query("SELECT * FROM $tabl");* – делаем запрос к выбранной базе данных. Получаем набор, состоящий из записей в таблице *Schetchik*. Идентификатор этого набора будет находиться в переменной *$mar*;

$k=mysql_fetch_array($mar); – функция *mysql_fetch_array*, как указывалось выше, возвращает строку в виде массива *$k*, ключами которого будут имена полей;

$k[vcego]=$k[vcego]+1; – в элементе массива *$k[vcego]* содержится общее число посетителей. Увеличиваем его на единицу;

if($day!==$k[chislo]) – если текущая дата не совпадает с той датой, которая указана в поле *chislo* базы данных (дата уже изменилась), то выполнится код между фигурными скобками под номером 2. Дату в поле *chislo*, если помните, мы записали в том же формате, что и потом в переменной *$day* (четыре цифры года, две цифры месяца и две цифры числа месяца);

2{

$k[chislo]=$day; – присваиваем элементу массива *$k[chislo]* текущую дату, чтобы потом обновить содержимое поле *chislo* в базе данных;

$k[segodna]=1;* – элементу массива *$k[segodna] присваиваем единицу (если наступили новые сутки, значение числа посетителей текущих суток сбрасывается опять на единицу);

2}

else – если текущая дата совпадает с той датой, которая указана в поле *chislo* базы данных (дата еще не изменилась), то выполнится строка между фигурными скобками под номером 3.

3{

$k[segodna]=$k[segodna]+1; – элемент массива *$k[segodna]* просто увеличиваем на единицу;

3}

mysql_query("UPDATE Schetchik SET vcego='$k[vcego]', segodna='$k[segodna]', chislo='$k[chislo]'"); – мы опять делаем запрос к текущей базе данных при помощи функции *mysql_query()* и вносим изменения в таблицу при помощи оператора UPDATE. Обновляем запись в полях таблицы. В поле *vcego* вносится обновленное значение общего числа посетителей, в поле *segodna* – значение числа посетителей за текущие сутки, в поле *chislo* обновляем дату;

echo "<table border=2 bordercolor=orange><tr>

<td colspan=2 align=center>Посещаемость</td></tr>

<tr><td align=center>Всего</td><td align=center>Сегодня</td></tr>

<tr><td align=center>$k[vcego]</td>

<td align=center>$k[segodna]</td></tr></table>";* – заносим значения переменных *$k[vcego] и *$k[segodna]* в небольшую таблицу и выводим эту таблицу при помощи оператора *echo* в окно браузера;

1} – окончание кода, выполняющегося при удачном соединении с сервером;

else – если соединения с сервером базы данных не произошло, то выводится сообщение об ошибке;

```
4{
echo "Error!";
4}
```

?> – конец PHP-кода.

Ну, вот и все. Как видите, использование базы данных позволяет значительно сократить код счетчика. Здесь нам не понадобились текстовые файлы. Однако скрипт работает быстрее без использования базы данных (для соединения с сервером требуется некоторое время). В общем, выбирать вам. Конечно, для счетчика посещений базу данных можно и не использовать (просто использовать текстовые файлы), но для более сложных проектов базы данных могут быть просто необходимы.

Вы, конечно, заметили что я не использовал подсчет уникальных посетителей (посетителей с различным IP). Попробуйте сделать это самостоятельно. Для этого создайте в базе данных вторую таблицу с единственным полем строкового типа, где будут сохраняться IP посетителей. Для поиска IP посетителя (посещал ли страницу данный посетитель с данным IP) можно применить функцию *mysql_fetch_array()*. При помощи данной функции можно перебрать все записи в таблице. Если таблица *Schetchik* содержит 3 поля и только одну запись, то таблица с IP-адресами будет иметь только одно поле и много записей (одна запись – один IP-адрес). При каждом вызове в скрипте функции *mysql_fetch_ array()* она выбирает следующую запись. Используя операторы цикла, можно, таким образом, перебрать все записи в таблице. Общее число записей в таблице можно определить при помощи функции *mysql_num_rows()*. Допустим, вы назовете таблицу с IP-адресами *adress*, а единственное поле этой таблицы будет называться *iprow*. Приведу кусок кода, который ищет IP-адрес в базе данных:

```
$tabl="adress";
$mar2=mysql_query("SELECT * FROM $tabl");
$ipnum=mysql_num_rows($mar2);
$ip=$_SERVER["REMOTE_ADDR"];
$s=0;
for ($i=0; $i<$ipnum; $i++)
{
$ipadr=mysql_fetch_ array($mar2);
if ($ipadr[iprow]==$ip) $s=$s+1;
}
if ($s=0) @mysql_query("INSERT INTO $tabl VALUES('$ip')");
```

В этом коде в переменной *$ipnum* находится общее число записей, т. е. общее число IP-адресов. В переменной *$ipadr[iprow]* находится выбранный из таблицы IP-адрес и сравнивается с IP-адресом пришедшего посетителя. Если IP-адрес не найден, т. е. посетитель посещает вашу страницу впервые, переменная $s останется равной нулю. В этом случае при помощи оператора *INSERT* мы вносим новую запись с новым IP в таблицу. Значок @ ставится для подавления ошибок при записи в таблицу.

Вот и все, что я хотел вам сказать о базах данных. Сделал я это очень кратко. Если хотите знать о базах данных больше, обратитесь к соответствующей литературе.

Глава 5. НОВОСТИ САЙТА

5.1. ИСПОЛЬЗОВАНИЕ ФОРМ И ОТПРАВКА ДАННЫХ НА СЕРВЕР

Для любого хорошего проекта, страница новостей сайта просто необходима. Она показывает, чем «живет» и «дышит» ваш сайт, что нового в нем появилось. Вместо новостей можно просто писать свои какие-нибудь рассуждения или просто мысли вслух, ваши впечатления о чем-нибудь. Или просто – услышали свежий анекдот, да и выложили его на сайт. Наверняка это будет интересно для посетителей сайта, а сам сайт станет «живым», если, конечно, хотя бы раз в неделю вы будете что-то записывать. Если посетителям будет интересно читать, что вы написали, то наверняка они захотят посещать ваш сайт вновь и вновь. Поддерживать страничку новостей очень просто. У вас будет текстовый файл, куда вы и будете записывать новости сайта или что-либо другое. Ваши записи с помощью программы на php будут выведены в браузер в удобно читаемом виде. Программа автоматически отсортирует новости (более свежие будут наверху) и поставит дату написания той или иной записи. Для заполнения текстового файла с новостями сделаем удобный для этого дела интерфейс в виде формы, но это чуть позже, а пока продолжим изучение языка php, а именно того, что нам понадобится для создания странички с новостями.

Очень часто на web-страницах сайтов используют элементы формы: раскрывающиеся списки, кнопки, поля ввода данных и т.п. Это позволяет посетителям вводить, например, свои данные, посылать сообщения и т. д. Все это должно быть принято и обработано какой-нибудь php-программой. Например, на форуме вы в текстовом поле создаете сообщение и отсылаете его при помощи нажатия кнопки формы. Спустя короткое время ваше сообщение появится на форуме сайта. Но для того чтобы это сообщение появилось, нужна php-программа, которая запишет ваше сообщение в какой-нибудь текстовый файл на сервере, сохранит его и выведет в окно браузера. Как работать с текстовыми файлами, мы уже разобрали в прошлой главе. В этой главе мы научимся создавать формы, отправлять данные на сервер и обрабатывать полученную информацию.

Форма на web-странице задается при помощи парного HTML-тега *<form>...</form>*. Все, что находится между этими тегами, и составляет форму. Тег *<form>* имеет свои атрибуты:

- **action** – указывает на URL-адрес программы, которая и обрабатывает переданные данные из формы. В нашем случае это будет PHP-файл, где со-

держится код для обработки переданных данных. Вместо URL можно указать электронный адрес, например:
action="mailto:name@name.ru" – в этом случае данные будут направлены по электронной почте по указанному адресу;

- **method** – метод передачи данных на сервер. Имеет 2 значения *get* и *post*:
  get – устанавливается по умолчанию и означает, что передаваемые из формы данные присоединяются к URL-адресу через разделитель в виде символа знака вопроса «*?*». Чтобы было понятнее, чуть позже я приведу пример. Этот метод используют для передачи небольших объемов данных;
  post – этим методом можно передавать большие объемы информации, а также сообщения для электронной почты. Данные будут находиться в теле запроса и не выводиться в адресную строку с URL-адресом. Это повышает безопасность передачи данных;
- **enctype** – метод кодирования передаваемых на сервер данных:
  text/plain – используется для передачи данных по электронной почте;
  multipart/form-data – используется для передачи файлов на сервер со своего компьютера. Посетители могут, например, передавать на ваш сайт картинки или фотографии, если, конечно, вы им это позволите;
  application/x-www-form-urlencoded – используется в остальных случаях. Это значение используется по умолчанию и его можно не указывать;
- **name** – имя формы.

Общий вид тега формы может быть таким:

<form action=novosti.php method=get> – при отправке данных запустится программа обработки этих данных в файле *novosti.php*. Данные передадутся методом *get*.

После тега *<form>* с включенными в него атрибутами, могут следовать элементы пользовательского интерфейса. Такие элементы, как кнопки, поля ввода данных, переключатели, создаются при помощи тега *<input>*. Атрибутом *type* задается тип элемента. Тип элемента может быть *text* (текстовое поле), *checkbox* (флажок), *radio* (радиокнопка), *password* (текстовое поле для ввода пароля), *file* (поле для выбора файла на вашем компьютере), *submit* (кнопка для отправки данных), reset (кнопка для отмены введенных данных), *image* (тоже кнопка для отправки данных, но в графическом виде).

Давайте рассмотрим и разберем эти и другие элементы пользовательского интерфейса.

В PHP-редакторе создайте новый файл. Сохраните его в нашей рабочей папке *htdocs* под названием *forma.php*. Для начала разберитесь в листинге 5.1. После листинга я подробно разъяснил почти каждую строчку. Только после этого наберите код листинга 5.1 самостоятельно.

Листинг 5.1 (файл forma.php)

```
<html>
<head>
 <title>Untitled web-page</title>
</head>
<body bgcolor="#C0C0C0"> //задаем цвет фона данной web-страницы.
<form action=novosti.php method=get>
<input type=text  value='Мой электронный адрес'  size=25 name=adres><br><br>
<input type=checkbox  value='Выбрать' checked name=fleg><br><br>
<input type=radio  value='Выбрать' checked name=radioknopka><br><br>
<input type=password name=parol><br><br>
<textarea name=novosti cols=50 rows=15 wrap=virtual></textarea><br><br>
<input type=file name=filename><br><br>
<b>Выберите нужное количество экземпляров</b><br>
<select name=knigi>
<option value=1>один экземпляр</option>
<option value=2>два экземпляра</option>
<option value=3>три экземпляра</option>
<option value=4>четыре экземпляра</option>
<option value=5>пять экземпляров</option>
<option value=0 selected>не заказывать</option>
</select><br><br>

<fieldset>
<b>Выберите нужную книгу</b><br>
<input type=radio value='Сборник стихов А. Ахматовой' name=books>Сборник стихов
А. Ахматовой<br>
<input type=radio value='Сборник стихов С. Есенина' checked name=books>Сборник
стихов С. Есенина<br>
<input type=radio value='Собачье сердце М. Булгакова' name=books>"Собачье
сердце" М. Булгакова<br>
<input type=radio value='Аэлита А. Толстого' name=books>"Аэлита" А. Толстого
</fieldset><br><br>

<input type=submit value='Отправить'>
<input type=reset value='Отменить'>
<input type=image src=pic2.bmp>
</form>
<?php

?>
</body>
</html>
```

Запустив файл с этой программой в браузере, вы увидите множество элементов формы (рис. 5.1).

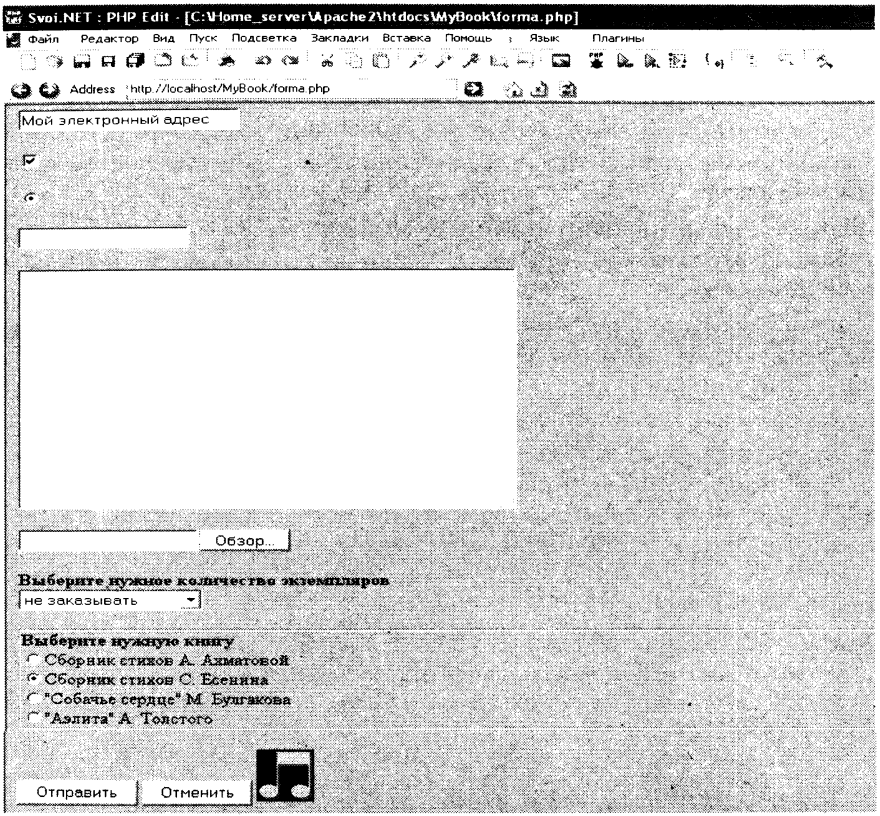


Рис. 5.1

Разберем теперь этот код, посматривая на рис. 5.1:

<form action=novosti.php method=get> – при отправке данных запустится программа обработки этих данных в файле *novosti.php* (этот файл мы заполним кодом потом). Данные передадутся методом *get*. Пока просто создайте в PHP-редакторе новый пустой файл и сохраните его в рабочей папке *hdocs* под названием *novosti.php*;

*<input type=text value='Мой электронный адрес' size=25 name=adres>

* – создает при помощи атрибута *type* текстовое поле для ввода данных, длиной в 25 символов. Это первый элемент, который вы увидите в браузере слева вверху. В поле уже содержится надпись, заданная атрибутом *value*.

Этому элементу присвоено имя, заданное атрибутом *name*. Значения атрибутов *type*, *name* указаны без кавычек, поскольку используется только одно английское слово. В атрибуте *value* используется русский текст, поэтому его обязательно заключаем в кавычки, даже если там было бы всего одно слово. После каждого элемента интерфейса, чтобы они не сливались и были хорошо видны, я пропускаю одну или две строки, при помощи оператора *
*;

*<input type=checkbox checked name=fleg>

* – создает флажок в виде маленького квадратика. Атрибут *checked* указывает, что данный флажок помечен галочкой;

*<input type=radio checked name=radioknopka>

* – создает радиокнопку, в виде меленького кружочка. Атрибут *checked* указывает, что данная кнопка отмечена;

*<input type=password name=parol>

* – создает текстовое поле для пароля. Буквы, вводимые в этом поле, будут отображаться в виде точек;

*<textarea name=novosti cols=50 rows=15 wrap=virtual></textarea>

* – создает большое текстовое поле или текстовую область для ввода посетителем какой-либо текстовой информации. Атрибут *cols* задает ширину этого поля, а именно, количество символов в каждой строке. Атрибут *rows*, соответственно, задает высоту поля или количество строк. Если количество строк превышает число, заданное параметром *rows*, то справа от текстового поля появится полоса прокрутки. Атрибут *wrap* включает автоматический перенос слов на другую строку;

*<input type=file name=filename>

* – создает элемент в виде небольшого текстового поля и кнопки, позволяющий вам выбрать какой-либо файл на вашем компьютере для отправки и отобразить его название (вернее, путь к нему) в этом текстовом поле.

Далее для примера мы создаем раскрывающийся список. Этот список создается тегом *<select>*. Элементы списка создаются тегами *<option>*. Пусть это будет список количества экземпляров выбранных книг. Сами книги будут выбираться чуть ниже (в блоке *<fieldset>*):

*Выберите нужное количество экземпляров
* – задаем текст над раскрывающимся списком, хотя в нашем случае это необязательно;

<select name=knigi> – создаем раскрывающийся список с именем *knigi*. Для этого тега существуют дополнительные атрибуты. Например, атрибут *align* задает выравнивание данного элемента по горизонтали (*align=center* – поместит раскрывающийся список по середине). Атрибут *multiple* дает возможность выбрать из списка несколько элементов. Атрибут *size* задает количество одновременно видимых элементов списка. По умолчанию, *size=1*.

Далее следуют 6 элементов списка, которые, как я уже отметил, задаются тегом *<option>*. Атрибут тега *value* задает ассоциированное значение для каждого элемента списка. Зачем это нужно, поймете чуть позже. Между тегами *<option>* и *</option>* надпись, которая будет в списке для каждого элемента:

<option value=1>один экземпляр</option>

<option value=2>два экземпляра</option>

<option value=3>три экземпляра</option>

<option value=4>четыре экземпляра</option>

<option value=5>пять экземпляров</option>

<option value=0 selected>не заказывать</option> – атрибут selected означает, что по умолчанию будет выбран именно этот элемент списка;

*</select>

 – завершаем список и ставим два тега перевода строки.*

Далее следует блок элементов из радиокнопок с одинаковыми именами *books*. Эти кнопки с одинаковыми именами образуют группу или массив. Особенностью этой группы является то, что выбрать можно только одну кнопку (в нашем случае только одну книгу). Атрибут тега *value* задает ассоциированное значение для каждой радиокнопки. Поскольку значению *value* мы присваиваем целое русское предложение, мы обязательно должны заключить его в кавычки. Тег *<fieldset>* создает выделенную рамку вокруг радиокнопок для оформления:

<fieldset>

*Выберите нужную книгу
*

*<input type=radio value=' Сборник стихов А. Ахматовой' name=books> Сборник стихов А. Ахматовой
* – первый элемент блока радиокнопок;

*<input type=radio value=' Сборник стихов С. Есенина' checked name=books> Сборник стихов С. Есенина
* – второй элемент блока радиокнопок. Атрибут checked указывает на то, что выбран именно этот элемент;

*<input type=radio value='Собачье сердце М. Булгакова' name=books>"Собачье сердце" М. Булгакова
* – третий элемент блока радиокнопок;

<input type=radio value='Аэлита А. Толстого' name=books>"Аэлита" А. Толстого> – и, наконец, четвертый элемент блока радиокнопок;

*</fieldset>

* – закрывающийся тег *</fieldset>* указывает нижнюю границу рамки. Если вы забудете поставить этот тег, то в выделенную рамку войдут все остальные элементы, расположенные ниже;

<input type=submit value='Отправить'> – создает один из главных элементов формы, а именно кнопку для отправки данных на сервер, т. е. тех данных, которыми вы заполнили текстовые поля формы. Надпись на кнопке, как вы уже знаете, задается атрибутом *value*;

<input type=reset value='Отменить'> – создает кнопку для отмены или удаления того, что вы написали в текстовых полях;

<input type=image src=pic2.bmp> – тоже создает кнопку для отправки данных, но в виде рисунка. Путь к рисунку указан в атрибуте *src*. Если графический файл расположен в отдельной папке, например *image*, то в этом случае нужно написать так: *src=image/pic2.bmp*. При щелчке мыши на этом рисунке осуществляется отправка данных;

</form> – тег «закрытия» формы.

```
<?php
```

После формы здесь может быть расположен PHP-код. Пока у нас тут ничего нет.

```
?>
```

Запустите вновь в браузере эту программу и заполните текстовые поля, например так, как указано внизу на рис. 5.2. В текстовом окне для ввода пароля (изображение в виде точек) я ввел 4 буквы на английском *lolo*. При помощи кнопки «*Обзор*» выбрал файл для отправки.

Нажмите на кнопку «*Отправить*». Мы перейдем с вами на страницу *novosti.php*, которая пока пуста. Если вы еще не создали эту страницу в нашей рабочей папке *htdocs*, то создайте ее и сохраните. Именно этот файл указан в теге *<form>*, куда мы должны перейти после нажатия кнопки «*Отправить*». После перехода на страницу *novosti.php*, обратите внимание на адресную строку браузера вверху. У меня, например, она имеет вид:

http://localhost/MyBook/novosti.php?adres=stroganov921@yandex.ru&fleg=on&radioknopka=on&parol=lolo&novosti=%C4%C5%CB%C0+%C8%C4%D3%D2+%CD%CE%D0%CC%C0%CB%DC%CD%CE.+%C8%C7%D3%D7%C0%DE+PHP.&filename=C%3A%5CHome_server%5CApache2%5Chtdocs%5CMyBook%5Cforma.php&knigi=2&books=Bulgakov.

Как вы видите, после названия страницы, куда мы переходим, следуют переданные на эту страницу данные. Сначала передается содержание первого текстового поля, куда я ввел электронный адрес. Далее у нас следует элемент флажковый переключатель *checkbox* (см. листинг 5.1), которому мы присвоили имя *fleg*. Это имя и будет ассоциироваться с данным элементом. Поскольку этот флажок у нас отмечен, то *fleg* будет иметь значение *on*. Далее идет элемент с именем *radioknopka*. Этот элемент у нас тоже отмечен, поэтому *radioknopka=on*. Затем, если вы помните, следует текстовое поле для ввода пароля с именем *parol*. В адресной строке браузера пароль становится видимым. Это нежелательно, поскольку ваш пароль или другие конфиденциальные данные могут быть перехвачены недоброжелателем. Чтобы этого не произошло, при использовании в форме паролей и другой секретной информации метод передачи данных на сервер (он указывается в теге *<form>*) должен быть не *GET*, а *POST*. В данном же случае, мы просто обучаемся, поэтому можно использовать метод *GET*.

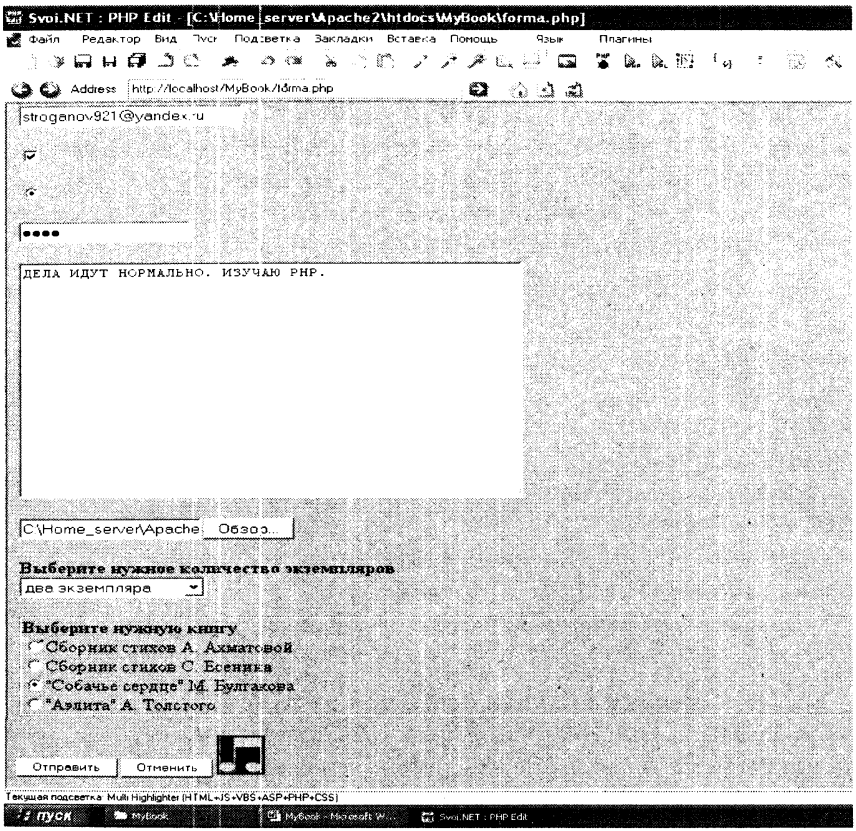


Рис. 5.2

После передается содержимое большого текстового поля с именем *novosti*. Русские буквы передаются в зашифрованном виде. В текстовом поле с именем *filename* у нас находится путь к файлу, который вы выбрали. Я выбрал файл с нашей формой *forma.php*. Далее идет раскрывающийся список именем *knigi*. Передаваемое значение *knigi=2* говорит о том, что выбран элемент списка со значением *value=2*. И, наконец, после следует массив радиокнопок с общим именем *books*. Если вы посмотрите в листинг 5.1, то увидите, что название книг Булгакова «Собачье сердце», которую я выбрал в форме, стоит рядом с радиокнопкой со значением *value=Bulgakov*. Именно это значение и передается на страницу *novosti.php*. Если вы выбрали другую книгу, то, соответственно, передастся другое значение. Все, адресную строчку в браузере мы разобрали.

Все данные, которые мы ввели в форме и передали методом *GET*, будут храниться в суперглобальном массиве *$_GET* (если данные передавались бы методом *POST*, то они были бы сохранены в суперглобальном массиве *$_POST*). Например, данные элемента формы с именем *books* (этим именем мы назвали массив радиокнопок для выбора книги) будут храняться в элементе суперглобального массива *$_GET['books']*, т. е. у нас *$_GET['books']=Bulgakov*.

Давайте, наконец, в файле *novosti.php* введем код с использованием элементов суперглобального массива (листинг 5.2).

Листинг 5.2 (файл novosti.php)

```
<?php
$adres=$_GET['adres'];
$knigi=$_GET['knigi'];
$books=$_GET['books'];
$novosti=$_GET['novosti'];
echo "<b>Уважаемый клиент! Вы нам прислали письмо с
текстом:<br>$novosti<br>
Вы выбрали $knigi экземпляра книги:<br>$books<br>
Условия оплаты данного заказа мы вышлем вам
по указанному вами электронному адресу:<br>$adres</b>";
?>
```

Вначале мы присваиваем неким переменным значения элементов суперглобального массива: *$имя переменной=$_GET['имя элемента формы']*. Имя переменной может и не совпадать с именем элемента формы. Здесь я это сделал для удобства. После присваивания в переменной *$adres*, например, будет находиться содержимое текстового поля с именем *adres*, которое мы заполняли в форме, а именно: *$adres=stroganov921@yandex.ru*. В переменной *$knigi* будет находиться значение *value* выбранного элемента раскрывающегося списка, которому было присвоено имя *knigi*. В нашем случае это количество заказанных экземпляров. Переменной *$books* будет присвоено значение *value* выбранной радиокнопки (выбранной книги). Ну и в переменной *$novosti* будет находиться содержимое большого текстового поля с именем *novosti*.

После оператора *echo* в двойных кавычках мы выводим текст, для удобства чтения разбивая его построчно при помощи оператора *
*. В текст вставлены наши переменные, которым выше были присвоены значения.

Снова запустите наш файл с формой и заполните эту форму, хотя бы как на рис. 5.2. В большом текстовом поле можете написать *«Желаю заказать у вас книгу»*.

Когда заполните, нажмите кнопку «*Отправить*» или графическую кнопку с рис. 5.2. Запустится созданный нами код в файле *novosti.php*. В браузер выведется результат работы кода листинга 5.2 (рис. 5.3).

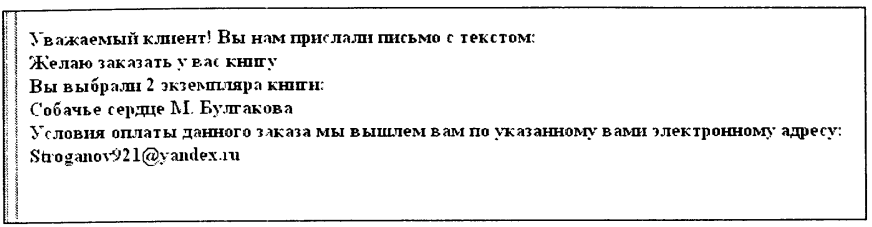


Рис. 5.3

Иногда возникает ситуация, когда на другую страницу (на ту, куда мы перейдем после нажатия кнопки формы) нужно передать какую-нибудь переменную. Это можно сделать при помощи скрытого элемента формы, который задается параметром *type* и имеет значение *hidden*. Например:

<input type=hidden name=parol value=$parol>* – в данном случае, на страницу, которую мы указали в теге *<form>*, передастся переменная *$parol. Она будет находиться в элементе суперглобального массива *$_GET['parol']*. Если вы назовете это скрытое поле по-другому, например *name=mytext*, то переданная переменная *$parol* будет находиться в *$_GET['mytext']*.

Передавать какие-либо переменные на какую-либо страницу можно и без использования формы. Для этого достаточно добавить к URL-адресу страницы имя переменной и его значение. Чтобы, например, переменная *$knigi* в листинге 5.2 файла *novosti.php* была определена, достаточно в адресной строке браузера после названия файла, добавить: *?knigi=2*, т. е. мы получим: *novosti.php?knigi=2*. Перед именем переменной символ *$* не указывается. Если передаваемых переменных несколько, то они отделяются друг от друга символом *&*. По умолчанию информация, добавляемая к URL-адресу, передается методом *GET*, т. е. значения передаваемых переменных будут храняться в элементах суперглобального массива *$_GET*. Данный способ используется также для передачи на одну и ту же страницу разной информации, в зависимости от передаваемого параметра.

Наберите в PHP-редакторе следующий код:

```
<?
$i=$_GET['i'];
if ($i==2)
{
echo "Переменная равна двум";
}
```

```
else if ($i==3)
{
echo "Переменная равна трем";
}
else
{
echo "Переменная неопределена или не равна двум или трем";
}
?>
```

Сохраните этот файл где-нибудь в папке *htdocs* под названием, например *peremen.php*. Запустите программу. Вы увидите в браузере надпись «*Переменная неопределена или не равна двум или трем*». Теперь в адресной строке браузера добавьте к строке параметр: *?i=2* и нажмите рядом на стрелочку, указывающую направо «перейти» (или вручную наберите в браузере: *http://localhost/ peremen.php?i=2*). Вы попадете на эту же страницу *peremen.php*, но с переданным параметром: *?i=2*. В результате в браузере вы увидите надпись «*Переменная равна двум*». Аналогично, если вы к строке допишете: *?i=3*, то в браузере увидите надпись «*Переменная равна трем*». Как видите, у нас 3 PHP-страницы объединены как бы в одну. Меняется только параметр *i*. Этот способ мы будем с вами использовать при выводе на страницу новостей сайта в этой главе, а также большого числа графических объектов в гл. 8.

Надеюсь, вы поняли этот раздел. Если нет, прочтите его еще раз. В этой главе нам понадобится форма для удобного ввода новостей или другой информации на сайт. Более сложная практика по использованию формы будет рассмотрена позже, при создании простого электронного магазина.

5.2. СОЗДАНИЕ СТРАНИЦЫ С НОВОСТЯМИ САЙТА

Для начала мы создадим программу для вывода новостей сайта или другой информации без использования формы. Новости или другая информация будут писаться вручную в текстовый файл, и эта информация будет отображена в окне браузера. Программа не вызовет у вас затруднений, если вы разобрались с материалами предыдущей главы при создании счетчика посещений.

В рабочей папке *htdocs* нашего домашнего сервера создайте папку *novosti*. В этой папке мы будем хранить страницы для вывода новостей. Создайте там простой текстовый файл, т. е. откройте блокнот, сохраните его в нашей созданной папке *novosti* под названием *novosti.txt*. Запишите в этом файле несколько строчек, примерно так:

первая новость
вторая новость
третья новость
четвертая новость
пятая новость
шестая новость
седьмая новость

Сохраните записанное и закройте файл. Теперь в PHP-редакторе создайте новый PHP-файл. Между тегами *<?php* и *?>* наберите следующий код:

Листинг 5.3 (файл nov1.php)

```
<?php
$filen="novosti.txt";
$chislo=5;
$lolo=file($filen);
krsort($lolo);
foreach($lolo as $line2)
{
$j++;
if($j<=$chislo) echo $line2."<br><br>";
}
?>
```

Сохраните этот файл в папке *novosti* под названием *nov1.php*. Обратите внимание, файл должен сохраниться именно с расширением *php*. Попробуйте разобраться в этом коде самостоятельно, не подсматривая на разбор кода ниже. Если это у вас вызывает затруднения, то еще раз прочтите предыдущую главу, где мы разбирали функции работы со строками и массивами. Разберем теперь строчки кода листинга 5.3:

<?php

$filen="novosti.txt";* – записываем в переменную *$filen путь к текстовому файлу с новостями. Поскольку этот файл находится в той же папке *novosti*, что и данная программа, мы указываем просто название файла;

$chislo=5; – вводим переменную $chislo и присваиваем ей число 5. Это будет количество выводимых новостей. Эта переменная будет подробно рассмотрена в конце кода;

$lolo=file($filen); – функция *file* создает массив строк из содержимого файла, указанного в переменной *$filen* и заносит весь этот массив в переменную *$lolo*. Таким образом, в переменной *$lolo* будет массив строк, которые мы за-

писали в файле *novosti.txt*. Причем первая строчка будет иметь индекс 0, вторая – индекс 1 и т. д.;

krsort($lolo); – функция *krsort* сортирует массив по убыванию значений индексов. Это делается для того, чтобы свежие новости были вверху, т. е. строки из файла *novosti.txt* выводились бы снизу вверх;

foreach($lolo as $line2) – функция *foreach* обходит массив, указанный в переменной *$lolo*, построчно, т. е. выполняется цикл, причем столько раз, сколько строчек в массиве *$lolo* (у нас их семь). При каждом выполнении цикла переменной *$line2* присваивается содержимое очередной строки, которое потом будет выводиться в окно браузера;

1{

$j++; – вводим счетчик цикла для подсчета количества новостей или других записей. При каждом исполнении цикла, он будет увеличиваться на единицу;

*if($j<=$chislo) echo $line2."

";* – проверяется условие. Если счетчик цикла *$j* меньше или равен числу в переменной *$chislo*, то в окно браузера выводится очередная строка из текстового файла с новостями. В переменной же *$chislo*, как было указано выше, мы будем указывать количество новостей, которое нужно будет вывести в браузер. Например, если у нас 1000 новостей, но нам нужно показать на странице только 5 последних, то в начале этого кода или же в коде той страницы, куда мы вставим данный код (при помощи оператора *include*) нужно указать, что *$chislo=5* (именно так у нас и сделано). Все, код разобран. Между новостями я поставил два html-оператора перевода строки *
*. Естественно, они должны быть заключены в кавычки и присоединены к строке с помощью точки (оператора конкатенации).

1}

?>

Итак, мы создали программу для вывода на страницу новостей сайта. Запустите в PHP-редакторе созданную программу. Слева вы увидите 5 последних строчек. Первой строкой у нас будет самая последняя строка в текстовом файле *novosti.txt*, а именно строка с надписью: *седьмая новость*. Далее идет шестая строка, пятая, четвертая, третья. Вторая и первая строки уже не выведутся в браузер, поскольку мы указали в коде, что нужно вывести только 5 новостей (строка с условием *if($j<=$chislo)* или *if($j<=5)*). Если надо вывести все новости, которые есть и будут, то установите, например, значение переменной *$chislo=1000*. Все новости, пока их не больше тысячи, будут выведены в браузер.

Давайте создадим теперь простую страницу новостей, куда мы и вставим файл с кодом *nov1.php*. В PHP-редакторе создайте новый файл и сохраните его под названием *nov2.php*. При помощи html-тегов создайте дизайн для этой страницы на ваше усмотрение. Я не буду создавать здесь красивый ди-

зайн, поскольку это выходит за рамки данной книги. Я набрал примерно такой код:

Листинг 5.4 (файл nov2.php)

```
<html>
<head>
<title>Новости сайта </title>
<style type="text/css">
#lolo{font-size:12pt; color: brown; font-family:Comic Sans MS}
</style
</head>
<body bgcolor="cyan">
<center><b  id="lolo" style="font-size:20pt">Новости сайта</b></center><br>
<table width="50%" border="2" bordercolor="green" bgcolor="silver"
id="lolo"><tr><td>
<?php
include ("nov1.php");
?>
</td></tr></table><br>
</body>
</html>
```

Разберем этот код, хотя все должно быть вам понятно и так, но, как говорят, повторение – мать учения:

<html> – начало HTML-кода;

<head>

<title>Новости сайта</title> – вводим заголовок страницы;

<style type="text/css"> – открываем таблицу стилей;

#lolo{font-size:12pt; color:red; font-family:Comic Sans MS} – вводим произвольный идентификатор стиля *lolo*, где указываем размер, цвет и тип шрифта;

</style> – закрываем таблицу стилей;

</head>

<body bgcolor="cyan"> – открываем тег основной части кода, где устанавливаем цвет фона cyan для всей страницы;

* Новости сайта *
*
* – выводим надпись вначале страницы, помещая ее в центр при помощи тегов * * и * *. В теге **, указывающем на жирный шрифт, мы вводим идентификатор стиля *lolo* (он был указан выше, в таблице стилей *<style>*), т. е. мы указываем браузеру, что между тегами ** и ** будет располагаться текст с типом шрифта *Comic Sans MS* коричневого цвета. Раз-

мер шрифта решил сделать побольше, дополнительно указав: *style="font-size:20pt"*;

<table width="50%" border="2" bordercolor="green" bgcolor="silver" id="lolo"><tr><td> – создаем простую таблицу, шириной в половину от общей ширины окна браузера. Далее указываем на ширину линий таблицы (*border="2"*), цвет линий (*bordercolor="green"*), цвет фона (*bgcolor="silver"*). Затем снова указываем идентификатор *lolo*, т. е. устанавливаем размер, цвет и тип шрифта, как было указано в начале кода, в таблице стилей. После открываем единственную строку таблицы при помощи тега *<tr>* и первую (и единственную) ячейку в этой строке при помощи оператора *<td>*;

<?php – начало PHP-кода, т. е. вставляем PHP-код внутри HTML-кода;

include ("nov1.php"); – вставляем код для вывода новостей в браузер, который у нас на странице *nov1.php*. Новости будут выведены в нашу одноколонную таблицу;

?> – конец PHP-кода;

*</td></tr></table>
* – закрываем ячейку, строку и саму таблицу;

</body> – закрываем основную часть HTML-кода;

</html> – конец HTML-кода.

Запустите в браузере созданную страницу *nov2.php*. Вы увидите примерно такую картину, как показано на рис. 5.4.

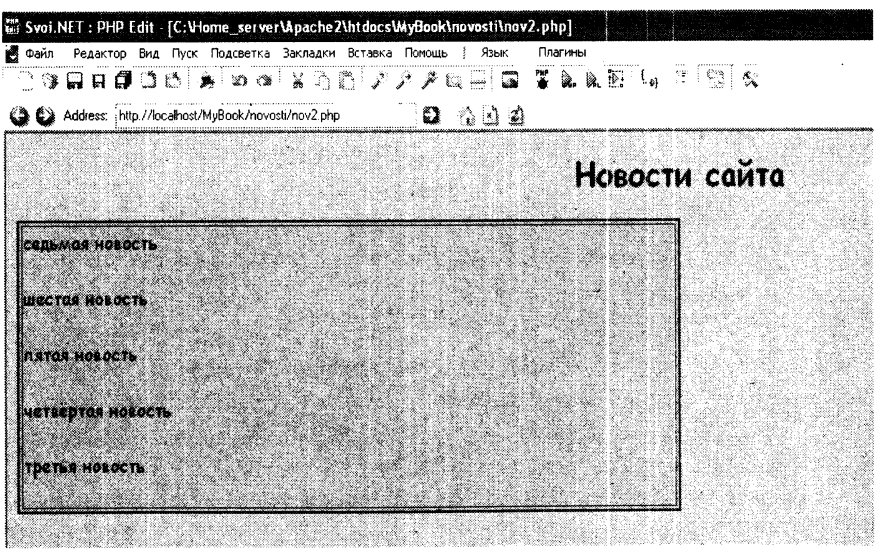


Рис. 5.4

Поэкспериментируйте. Допишите в текстовый файл *novosti.txt* внизу еще строчку: *восьмая новость* и сохраните файл. Вновь запустите в браузере файл *nov2.php*. Последняя написанная нами строчка появится первой, а строка: *седьмая новость* сместится на вторую позицию.

Итак, простейшая страница новостей создана. Можно, конечно, на этом и закончить. Вы будете заходить на хостинг, где будет расположен ваш сайт и вручную редактировать файл *novosti.txt*. Однако на то и существует php-программирование, чтобы облегчить задачу по созданию и наполнению страниц сайта. Используя формы, мы создадим удобный интерфейс написания новостей или другой информации для вашего сайта. Вам не нужно будет постоянно заходить на хостинг. У вас будет в Интернете специальная страничка администратора, откуда вы и будете заполнять страницу новостей. Заходить на страницу администратора сможете по паролю только вы. Пароль будет храниться в зашифрованном виде в каком-нибудь текстовом файле. В конце каждой новости будет автоматически добавляться дата написания данной новости. На страницу будет выводиться по 5 новостей. На каждой странице с новостями, естественно, будут указаны ссылки на другие страницы с новостями. Страница с новостями будет, правда, всего одна. Мы просто введем некий параметр, в зависимости от значения которого будет выводиться та или иная пятерка новостей.

При входе на страницу редактирования новостей появится текстовое поле для ввода пароля, который будете знать только вы. Вы вводите пароль и нажимаете на кнопку «*ввести пароль*», которую мы создадим. Программа сравнит содержимое заполненного вами текстового поля с содержимым текстового файла, где у нас будет храниться пароль. Если пароль будет введен правильно, то откроется текстовое окно для редактирования файла с новостями. В противном случае будет выведена надпись, что пароль введен неверно. Создадим сначала текстовый файл с паролем. Можно, конечно, это сделать вручную, т. е. создать простой текстовый файл и записать туда свой пароль. Этого делать категорически нельзя! Если какому-нибудь недоброжелателю удастся открыть файл с паролем, то он без особого труда может зайти на вашу страничку для редактирования новостей и записать на ваш сайт все, что вздумается, например вредоносный код. Пароль нужно обязательно зашифровать. Сейчас мы создадим небольшую программу, которая сама создаст текстовый файл с зашифрованным паролем.

В PHP-редакторе создайте новый файл. Сохраните его в папке *novosti* под названием *newparol.php*. Наберите следующий код между тегами *<?php* и *?>*.

Листинг 5.5 (файл newparol.php)

```
<?php
if(!$_GET['parol'])
{
 echo "<form action=newparol.php method=GET>";
 echo "<input type=password name=parol>";
```

```
 echo "<input type=submit value='Ввести пароль'>";
 echo "</form><br><br>";
}
else
{
 echo "ПАРОЛЬ ВВЕДЕН!!!";
 $parol=$_GET['parol'];
 $file="parol.txt";
 $newfile=fopen($file, "w+");
 $write=fwrite($newfile, md5($parol));
 fclose($newfile);
}
?>
```

Здесь мы создаем форму, на которой будут находиться всего 2 элемента: поле для ввода пароля (задается атрибутом *type=password*) и кнопка (задается атрибутом *type=submit*). Сохраните написанный код и запустите браузер. Вы увидите примерно такое, как на рис. 5.5.

Введите какой-нибудь пароль и нажмите на кнопку. После нажатия элементы формы исчезнут, а в браузере появиться надпись «*ПАРОЛЬ ВВЕДЕН!!!*». Зайдите в папку *novosti*. У вас там появится текстовый файл *parol.txt*. Откройте его. Вы увидите там свой введенный пароль, но в зашифрованном виде (хеш-код). Догадаться, что же было введено, практически невозможно.

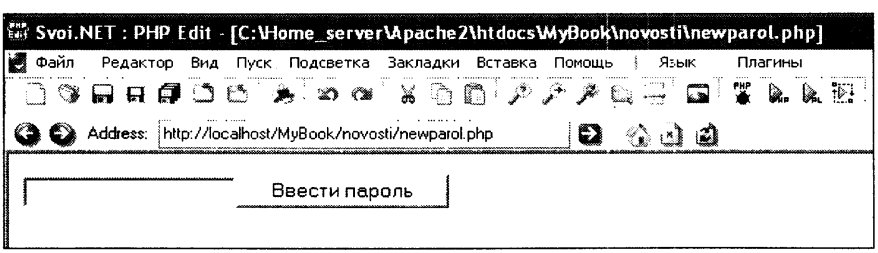


Рис. 5.5

В программе мы использовали функцию необратимого шифрования *md5*. Обратной расшифровке пароль не подлежит, да этого нам и не нужно.

При нажатии кнопки «Ввести пароль», мы направимся на эту же страницу, поскольку в теге *<form>* указан параметр *action=newparol.php*. В элементе суперглобального массива *$_GET['parol']* будет находиться введенный в текстовое поле пароль. Именно имя *parol* мы присвоили текстовому полю для ввода пароля (*name=parol*).

Давайте разберем строчки кода в листинге 5.5:

if(!$_GET['parol']) – сначала проверяем, не существует ли элемента суперглобального массива, в котором находится введенный пароль. Естественно, если мы только запустим программу, но еще не нажмем на кнопку формы, этот элемент будет еще не определен, поскольку данные из формы еще не переданы. Значит, условие истинно (перед элементом *$_GET['parol']* стоит символ отрицания, т. е. восклицательный знак). Условие будет означать следующее: если элемента суперглобального массива *$_GET['parol']* еще не существует, то выполнится блок команд в фигурных скобках (под номером 1). А в фигурных скобках у нас вывод формы в браузер;

1{

echo "<form action=newparol.php method=GET>"; – при помощи операторов *echo* выводим форму, данные из которой при нажатии на кнопку в форме отправятся методом *GET* на страницу *newparol.php*, т. е. на эту же страницу;

echo "<input type=password name=parol>"; – текстовое поле для ввода пароля. Даем имя этому полю *name=parol*;

echo "<input type=submit value='Ввести пароль'>"; – отображаем кнопку для отправки данных из этой формы. Заметьте, что фраза *'Ввести пароль'* берется в одинарные кавычки, поскольку двойные мы уже используем. Применение двойных кавычек внутри двойных может вызвать ошибку;

*echo "</form>

";*

1} – закрываем форму и завершаем этот блок программ. Весь этот блок будет выполняться, если кнопка еще не нажата!

else – если же элемент массива *$_GET['parol']* уже существует, т. е. мы уже нажали кнопку и отправили пароль на эту же страницу, то выполнится только блок программ, представленный ниже (в фигурных скобках под номером 2). Прошлый же блок программ, представленный выше, выполняться уже не будет, т. е. форма выведена уже не будет. Вместо нее мы выведем в браузер простую надпись, что пароль уже введен;

2{

echo "ПАРОЛЬ ВВЕДЕН!!!"; – вывод надписи в браузер;

$parol=$_GET['parol']; – переменной *$parol* присваиваем значение суперглобального массива *$_GET['parol']*, т. е. введенный нами пароль;

$file="parol.txt";* – переменной *$file присваиваем путь к текстовому файлу, где у нас будет зашифрованный пароль. Пусть он будет у нас в той же папке *novosti*, что и эта программа;

$newfile=fopen($file, "w+"); – открывается текстовый файл для пароля, стирая там все содержимое, или создается новый файл, если пароль вводим впервые;

$write=fwrite($newfile, md5($parol));* – записываем в открытый текстовый файл пароль, хранящийся у нас в переменной *$parol, зашифровав его при помощи функции *md5*;

fclose($newfile); – закрываем текстовый файл с паролем.

2}

Итак, мы создали текстовый файл с зашифрованным паролем. Созданную страницу *newparol.php* нужно оставить только на своем компьютере (!) и не загружать ее на сервер в Интернете, куда вы потом будете загружать свой сайт, оплатив хостинг. На сервер нужно будет загрузить текстовый файл с зашифрованным паролем *parol.txt*. Потом, если вы вдруг забудете пароль или захотите его просто поменять, при помощи программы на вашем компьютере *newparol.php* создадите новый файл с паролем и с тем же названием *parol.txt* и закачаете его на свой сервер в Интернете, заменив им старый файл с паролем.

Перейдем непосредственно к созданию интерфейса для редактирования новостей сайта, т. е. создадим программу, с помощью которой было бы легко и удобно заполнять сайт новостями или какой-либо другой информацией.

При запуске программы, которую мы создадим, в браузере сначала появится текстовое поле для ввода пароля, как на рис. 5.6.

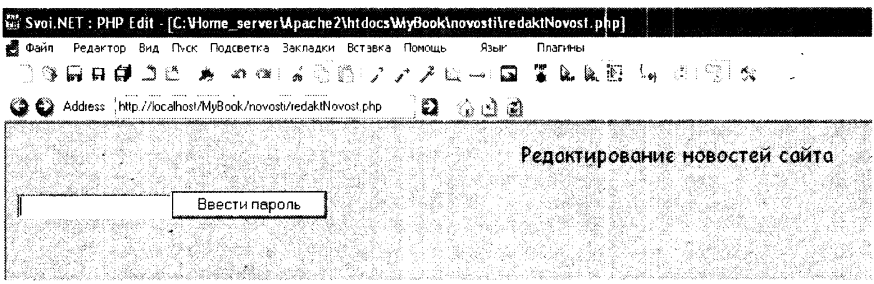


Рис. 5.6

После ввода правильного пароля и нажатия на кнопку «*Ввести пароль*», появится большое текстовое поле, в которое загрузится для редактирования текстовый файл с новостями (в нашем случае это *novosti.txt*). Получится, примерно, как на рис. 5.7.

Внизу мы пишем очередную новость и нажимаем кнопку «*Добавить*». После этого на нашей web-страничке для новостей сайта появится свежая запись, причем она будет стоять первой. В конце записи будет указана дата появления данной записи. Если же нам не нужно добавлять новую запись, а надо просто изменить уже существующие, то перед тем как нажать на кнопку, уберем галочку с надписи «*Новая запись*».

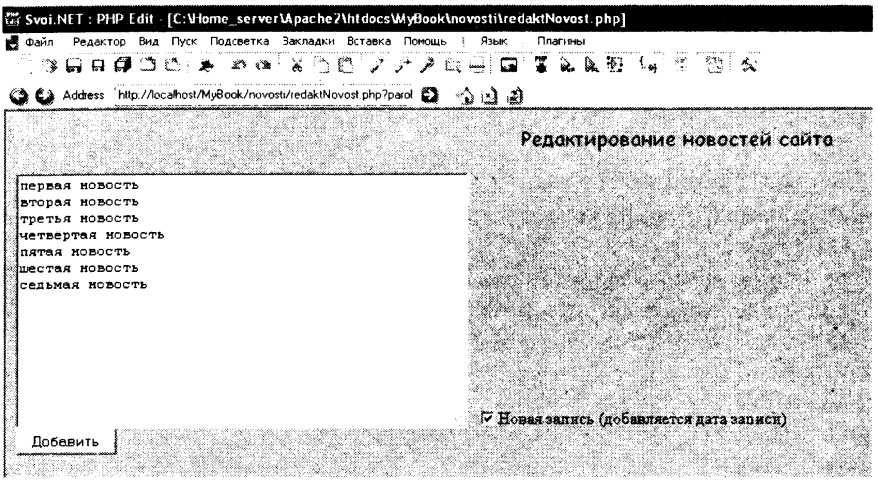


Рис. 5.7

Для написания кода в PHP-редакторе создайте новый файл и сохраните его в папке *novosti* под именем, например, *redaktNovost.php*. Код программы в листинге 5.6.

Листинг 5.6 (файл redaktNovost.php)

```
<html>
<head>
<title>Редактор новостей </title>
<style type="text/css">
#lolo{font-size:14pt; color:red; font-family:Comic Sans MS}
</style>
</head>
<body bgcolor="cyan">
<center><div id="lolo">Редактирование новостей сайта</div></center>
<?
if((!$_GET['novosti']) and (!$_GET['parol']))
{
 echo "<form action=redaktNovost.php method=GET>";
 echo "<input type=PASSWORD name=parol>";
 echo "<input type=SUBMIT value='Ввести пароль'>";
 echo "</form><br><br>";
}
$parol=$_GET['parol'];
$file="parol.txt";
```

```
$open=fopen($file, "r");
$read=fread($open, filesize($file));
fclose($open);
if(md5($parol)==$read)
{
 $nov=fopen("novosti.txt", "r+");
 @$redakt=fread($nov, filesize("novosti.txt"));
 fclose($nov);
 echo "<form action=redaktNovost.php method=GET>";
 echo "<textarea name=novosti cols=50 rows=15 wrap=virtual>$redakt</textarea>";
 echo "<input type=HIDDEN name=parol value=$parol>";
 echo "<input type=CHECKBOX name=new CHECKED>Новая запись
(добавляется дата записи)<br>";
 echo "<input type=SUBMIT value=Добавить>";
 echo "</form>";
 if(isset($_GET['novosti']))
 {
  $novosti=$_GET['novosti'];
  $data=date('d.m.Y');
  $nov=fopen("novosti.txt", "w+");
  if($_GET['new']==true)
  {
   fwrite($nov, "$novosti"."($data)"."\n");
  }
  else
 {
  fwrite($nov, "$novosti");
 }
  fclose($nov);
 }
}
else if($parol)
{
 echo "Неправильный пароль!";
}
?>
</body>
</html>
```

Разберем теперь строчки этого кода:

<html> – начало HTML-кода;

<head> – начало заглавной части кода;

<title>Редактор новостей </title> – заголовок страницы;

<style type="text/css"> – начало таблицы стилей;

#lolo{font-size:14pt; color:red; font-family:Comic Sans MS} – вводим идентификатор стиля, чтобы потом использовать его для задания шрифта текста;

</style> – закрываем тег таблицы стилей;

</head> – конец заглавной части кода;

<body bgcolor="cyan"> – начало основной части HTML-кода. Задаем цвет фона страницы;

* Редактирование новостей сайта * – выводим в центре надпись в верху окна браузера. Надпись выводится внутри отдельного контейнера при помощи тега * *. Этот контейнерный тег служит для выделения части документа, в нашем случае большого заголовка на странице. Шрифт надписи задается идентификатором стиля *lolo*;

<? – начало PHP-кода;

if (!$_GET['parol']) – вначале проверяется условие, не вводился ли уже пароль. Естественно, вначале при загрузке этой странице в браузер пароль еще не вводился (см. начало кода прошлого листинга 5.5). Значит, условие в круглых скобках истинно (перед элементом *$_GET['parol']* стоит символ отрицания, т. е. восклицательный знак). В этом случае выполнится блок команд в фигурных скобках ниже, т. е. вначале при загрузке этой страницы в браузер выведется форма с текстовым полем для ввода пароля и кнопки;

1{ – чтобы не было путаницы в фигурных скобках, я решил их попарно пронумеровать, т. е. номеру открывающейся скобки будет соответствовать номер закрывающейся скобки внизу;

echo "<form action=redaktNovost.php method=GET>"; – при помощи операторов *echo* выводим форму, данные из которой при нажатии на кнопку в форме отправятся методом *GET* на страницу *redaktNovost.php*, т. е. на эту же страницу;

echo "<input type=PASSWORD name=parol>"; – текстовое поле для ввода пароля. Даем имя этому полю *name=parol*. Слово *PASSWORD* можно вводить как заглавными, так и строчными буквами, значения не имеет;

echo "<input type=SUBMIT value='Ввести пароль'>"; – отображаем кнопку для отправки данных из этой формы;

*echo "</form>

";*

1} – закрываем форму и завершаем этот блок программ.

Если мы введем пароль и нажмем на кнопку «*Ввести пароль*», то опять перейдем на эту же страницу *redaktNovost.php*, но текстовое поле для пароля и нажатая кнопка исчезнут, поскольку блок команд в фигурных скобках выше исполняться уже не будет (условие *if(!$_GET['parol'])* станет ложным). Теперь выполнится код ниже:

$parol=$_GET['parol']; – присваиваем переменной *$parol* элемент суперглобального массива *$_GET['parol']*. Этому элементу, при нажатии кнопки «*Ввести пароль*» было присвоено содержание текстового поля для ввода пароля с именем *name=parol* (см. выше). Теперь пароль у нас будет храниться в переменной *$parol*;

$file="parol.txt";* – в переменной *$file будет содержаться путь к текстовому файлу, где у нас хранится зашифрованный пароль. Этот файл, если вы помните, был создан заранее;

$open=fopen($file, "r"); – открываем файл с паролем для чтения и создаем его идентификатор в виде переменной *$open*;

$read=fread($open, filesize($file)); – используя функцию *fread*, считываем содержимое файла с паролем и присваиваем его переменной *$read*. Функция *filesize($file)* возвращает общее количество байтов в файле. Получается в нашем случае, что функция *fread* считывает все байты файла, ибо в качестве второго аргумента этой функции стоит общее число байтов в файле. Можно, конечно, вторым аргументом использовать просто число, например 100 (что мы и делали раньше). Сто байт будет достаточно для прочтения содержимого файла с паролем;

fclose($open); – закрываем файл с паролем;

if(md5($parol)==$read) – здесь проверяется условие, соответствует ли введенный вами пароль, который мы шифруем функцией *md5*, тому, что находится у нас в переменной *$read*. А в эту переменную было записано содержимое файла с паролем, который мы создали раньше. Причем пароль там тоже зашифрован функцией *md5*. Естественно, если зашифрованные пароли совпадают друг с другом, то данное условное выражение будет истинным и выполнится большой блок команд ниже в фигурных скобках под номером 2;

2{ – для удобства понимания (это я указал чуть выше) я нумерую фигурные скобки, чтобы было видно, где начинается, а где заканчивается данный блок команд. Это очень важно, поскольку внутри этого блока будут другие подблоки со своими фигурными скобками. Можно легко запутаться;

$nov=fopen("novosti.txt", "r+"); – итак, если пароль введен правильно, открываем файл с новостями *novosti.txt* для чтения;

@$redakt=fread($nov, filesize("novosti.txt")); – считываем этот файл полностью и присваиваем считанное переменной *$redakt*. Символ амперсанда пе-

ред командой @ подавляет возникающие ошибки. Дело в том, что если файл с новостями еще пуст (допустим, вы еще не записали ни одной новости), то данная команда вызовет ошибку, поскольку функция *fread* не сможет прочесть пустой файл. Эта ошибка в общем-то незначительная, и ее можно подавить символом @:

fclose($nov); – закрываем файл с новостями;

echo "<form action=redaktNovost.php method=GET>"; – выводим форму для редактирования файла с новостями;

echo "<textarea name=novosti cols=50 rows=15 wrap=virtual>$redakt </textarea>"; – создаем большое текстовое поле или текстовую область с именем *name=novosti* шириной в 50 символов и высотой в 15 строк. Здесь можете задать другие, удобные для вас параметры. Атрибут *wrap* включает автоматический перенос слов на другую строку. В эту текстовую область уже будет выведено содержимое файла с новостями, которое у нас в переменной *$redakt*. Эта переменная, как вы видите, указывается между тегами *<textarea>* и *</textarea>*;

echo "<input type=HIDDEN name=parol value=$parol>";* – вводим скрытое поле. Оно нам нужно для передачи переменной с паролем. В данном случае на страницу, которую мы указали в теге *<form>* (т. е. на эту же страницу), передастся переменная *$parol. Она будет находиться в элементе суперглобального массива *$_GET['parol']*. Дело в том, что если мы эту переменную не передадим, то после нажатия кнопки этой формы и перехода опять на эту же страницу (страница как бы заново перезагрузится) элемент суперглобального массива *$_GET['parol']* снова не будет определен, и в браузер снова выведется форма с текстовым полем для пароля (см. начало кода);

*echo "<input type=CHECKBOX name=new CHECKED>Новая запись (добавляется дата записи)
";* – создает флажок в виде маленького квадратика. Атрибут *checked* указывает, что данный флажок отмечен галочкой. Зачем нам этот элемент, поймете позже;

echo "<input type=SUBMIT value=Добавить>"; – кнопка для отправки данных из этой формы. Добавив в текстовую область новую информацию, нажимаем на эту кнопку и вновь переходим на эту же страницу;

echo "</form>"; – закрываем форму;

if(isset($_GET['novosti'])) – снова условный оператор, который проверяет при помощи функции *isset*, существует ли элемент суперглобального массива *$_GET['novosti']*. В этом элементе, как вы уже догадались, находится уже измененное содержимое текстовой области, которой (см. выше) мы присвоили имя *name=novosti*. Если условие истинно, то выполнится блок команд в фигурных скобках под номером 3;

3{

$novosti=$_GET['novosti']; – переменной *$novosti* присваиваем измененное содержимое текстовой области. Это содержимое уже будет отличаться от содержимого в переменной *$redakt* (в этой переменной, как указывалось выше, находится содержимое текстового файла новостей до внесенных изменений). Теперь нам остается заменить в файле novosti.txt старое содержимое новым;

$data=date('d.m.Y');* – в переменную *$data, при помощи функции date записываем текущую дату в формате, как указанно в круглых скобках (день, месяц, год);

$nov=fopen("novosti.txt", "w+"); – открываем текстовый файл с новостями, стирая старое содержимое;

if($_GET['new']==true) – снова условный оператор. В элементе суперглобального массива *$_GET['new']* содержится информация о состоянии флажка *CHECKBOX* в форме. Если помните, мы присвоили ему имя *name=new*. Если флажок отмечен галочкой, то элемент *$_GET['new']* возвращает значение *true*. В этом случае условие истинно и выполнится код между фигурными скобками под номером 4;

4{

fwrite($nov, "$novosti"."($data)"."\n");* – а если условие истинно (флажок отмечен), то делаем запись в текстовый файл, которому мы присвоили идентификатор *$nov. Эта запись состоит (второй аргумент функции) из нового содержимого текстовой области (в переменной *$novosti*). К этому содержимому методом конкатенации (с помощью точки) добавляется текущая дата. Затем добавляется символ перевода строки «\n». В результате выполнения этой команды в файл *novosti.txt* запишется новая информация, в конце которой в круглых скобках будет стоять дата записи. Затем курсор перейдет на новую строку, чтобы потом начать с нее новую запись;

4}

else – если прошлое условие ложно, т. е. мы убрали галочку с флажка *CHECKBOX* (нам не нужно добавлять новую запись, а просто отредактировать старые), то новую запись запишем по-другому;

5{

fwrite($nov, "$novosti"); – в данном случае мы просто запишем в текстовый файл *novosti.txt* обновленное содержимое. Дату и перевод строки ставить не нужно;

5}

fclose($nov); – закрываем файл с новостями;

3} – закрывается скобка под номером 3, т. е. прекращается блок команд условного оператора *if(isset($_GET['novosti']))* (см. выше);

2} – прекращается блок команд истинности условного оператора *if(md5($parol)==$read)*. Весь этот большой блок – от открывающейся до закрывающейся фигурной скобки под номером 2 – выполнится, если вы правильно ввели пароль;

else if($parol) – если пароль был введен неправильно и переменная *$parol* существует, то выполнится команда в фигурных скобках под номером 6. Можно было, конечно, после оператора *else* не задавать дополнительного условия *if($parol)*, проверяющего существование переменной *$parol*, но в этом случае при запуске программы выполнилась бы и приведенная ниже строчка. Когда мы запускаем эту программу, то переменной *$parol* еще не существует, но в коде-то она присутствует при проверке условия *if(md5($parol)==$read)* и получается, что сразу при запуске программы это условие, естественно, ложно, поэтому вместе с текстовым полем для ввода пароля, мы увидим и надпись, что введен неправильный пароль. Поэтому мы и проверяем существование переменной *$parol*, чтобы приведенная ниже строчка не выполнялась сразу же после запуска этой программы;

6{

echo "Неправильный пароль!"; – надпись в браузере в случае ввода неверного пароля;

6}

?> – конец PHP-кода;

</body> – конец основной части документа;

</html> – конец HTML-кода.

Прежде чем запустить эту программу, полностью очистите содержимое текстового файла *novosti.txt* и сохраните его. Теперь запустите разобранную выше программу *redaktNovost.php*. Введите правильно пароль (тот, который вы вводили и шифровали при помощи программы *newparol.php*). Откроется текстовая область. Она будет пуста, поскольку вы должны были стереть содержимое файла новостей. Напишите там фразу «*первая новость*» и нажмите на кнопку «*Добавить*». В браузере вы должны увидеть свою надпись (рис. 5.8).

Как видите, к записи автоматически добавилась текущая дата, а курсор перешел на другую строчку. Добавьте несколько других строк, например, «*вторая новость*», потом «*третья новость*» и т. д. до строчки «*шестнадцатая новость*». Не поленитесь это сделать, пригодится.

Уберите галочку у надписи «*Новая запись...*», измените какую-нибудь надпись и нажмите на кнопку «*Добавить*». В этом случае дата не должна добавляться. Перезапустите данную программу и введите неправильный пароль. После нажатия кнопки «*Ввести пароль*» в браузер выведется надпись: «*Неправильный пароль!*».

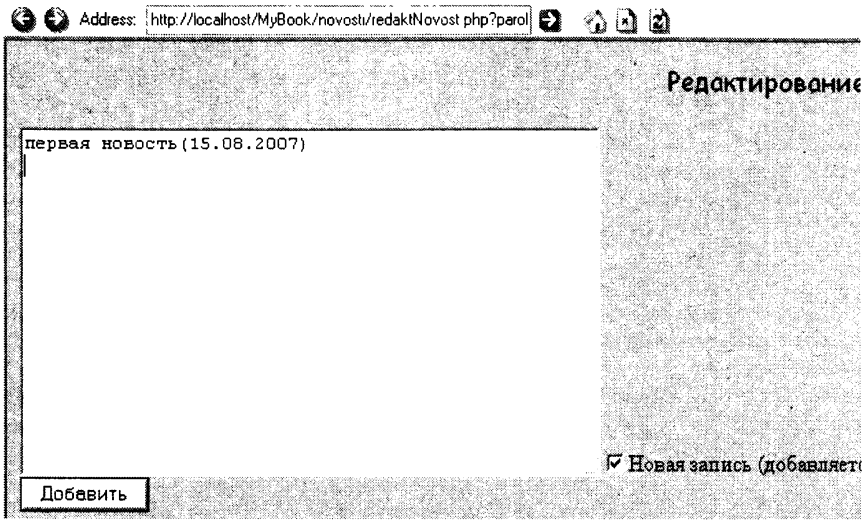


Рис. 5.8

Откройте файл *novosti.txt*. Там вы увидите все введенные вами строчки. Теперь запустите программу, где все эти строчки выводятся в окно браузера. Если помните, это программа *nov2.php* (листинг 5.4). После запуска в браузер выведутся последние 5 новостей. Если нужно, чтобы в браузер были выведены все новости, то измените переменную *$chislo*, которая определена в файле *nov1.php* (листинг 5.3). Там указано, что *$chislo=5*. Измените это число, например, на 1000 и сохраните файл. Запустите программу *nov2.php*. В одноколонную таблицу будут выведены все новости, по крайней мере пока их меньше, чем 1000. Все это хорошо, но вы, наверное, согласны с тем, что выводить все новости на одну страницу нецелесообразно. Допустим, у вас 1000 новостей или каких-нибудь других записей. Вы хотите, чтобы на странице было не более пяти новостей. Остальные новости (тоже по пять) находились бы на других страницах, причем на каждой из страниц должны быть ссылки на остальные страницы. Эти остальные страницы создавать самому не нужно. Как я уже указывал выше, страница будет всего одна, на которую будет выводиться та или иная пятерка новостей, в зависимости от параметра. Ссылки на другие страницы – это ссылки на одну и ту же страницу (на страницу с новостями), но с другими параметрами. Все это вполне реализуемо. Вы будете только записывать новости в текстовый файл при помощи созданного нами интерфейса *redaktNovost.php*. Все остальное за вас сделает программа! Эту программу мы с вами сейчас и создадим.

Для начала нам нужно немного изменить программу *nov1.php*. В коде этой программы (листинг 5.3) удалите строчки цикла *foreach($lolo as $line2)*.

Этот цикл, если вы помните, выводит последние 5 новостей в браузер, но нам теперь нужно, чтобы зыводились все новости на нескольких страницах, по 5 новостей на каждой. Следовательно, в коде листинга 5.3 останутся всего 4 строчки:

Листинг 5.7 (исправленный листинг 5.3 программы nov1.php)

```
<?php
$filen="novosti.txt";
$chislo=5;
$lolo=file($filen);
krsort($lolo);
?>
```

Данная программа будет только формировать массив новостей (строк) из текстового файла и сортировать в порядке, обратном их поступлению. Что же касается программы *nov2.php*, которая выводит новости в одноколонную таблицу, то нам ее придется почти полностью переделать, а лучше этот файл вообще удалить и создать новый, что мы и сделаем. В нашем PHP-редакторе создайте новую страницу. Назовите ее *new1.php* и сохраните ее в папке *novosti*. В листинге 5.8 приведена программа для страницы с новостями *new1.php*. Постарайтесь сначала разобраться сами в этом коде (если не сможете, разбор кода приведен ниже, после листинга 5.8), а уж потом набрать его в PHP-редакторе.

Листинг 5.8 (new1.php)

```
<html>
<head>
<title>Новости сайта</title>
<style type="text/css">
#lolo{font-size:12pt; color:brown; font-family:Comic Sans MS}
</style>
</head>
<body bgcolor="cyan">
<center><b id="lolo" style="font-size:20pt">Новости сайта</b></center><br>
<table width="50%" border="2" bordercolor="green" bgcolor="silver"
id="lolo"><tr><td>
<?php
if (!$_GET['i'])
{
 $i=1;
}
```

```
else
{
 $i=$_GET['i'];
}
include ("nov1.php");
foreach($lolo as $line2)
{
 $j++;
 if(($j>$chislo*$i-$chislo)&&($j<=$chislo*$i))
 {
  echo $line2."<br><br>";
 }
}
echo"</td></tr></table>";
echo "<center><p>";
for($m=1; $m<=$j/$chislo+1; $m++)
{
 if ($m==$i)
 {
  echo "<b>$m</b>";
 }
 else
 {
  echo "<a href='new1.php?i=$m'><b>$m</b></a>";
 }
}
echo "</p></center>";
?>
</body>
</html>
```

Наибольшая часть кода вам должна быть понятна. Вначале при помощи html-тегов создаем одноколонную таблицу. В эту таблицу записывается 5 новостей. Какие это будут по счету новости, зависит от значения переменной *$i*. Значение этой переменной (параметр *i*) мы передадим странице с новостями *new1.php* в адресной строке браузера (см. разд. 5.1). Разберем подробно листинг 5.8. Постарайтесь понять сам принцип передачи параметра *i* на страницу. Это очень важно, поскольку этот прием мы будем использовать при создании форума или при размещении на своих web-страницах картинок, фотографий и т.п.

<html> – начало HTML-кода. Некоторые строчки ниже оставляю без комментария, поскольку их назначение был рассмотрено в прошлых программах;

```
<head>
<title>Новости сайта</title>
<style type="text/css">
#lolo{font-size:12pt; color:brown; font-family:Comic Sans MS}
</style>
</head>
<body bgcolor="cyan">
<center><b id="lolo" style="font-size:20pt">Новости
сайта</b></center><br>
```

<table width="50%" border="2" bordercolor="green" bgcolor="silver" id="lolo"><tr><td> – создаем одноколонную таблицу, задавая ее ширину и цвета по своему усмотрению. При помощи тегов *<tr>* и *<td>* открываем, соответственно, строку и единственную в этой строке ячейку;

<?php – начало PHP-кода;

if (!$_GET['i']) – при загрузке данной страницы, мы должны сначала определиться со значением переменной *$i*. Ведь от нее зависит, какие новости будут выводиться. Смысл условия следующий: если переменная *$i* не получена (или мы ее не указали), то мы присваиваем ей значение единица;

```
{
$i=1;
}
```

else – иначе, если мы задали значение *i* в строке браузера, например, *new1.php?i=2*, то присваиваем переменной *$i* указанное значение (это значение будет храниться в элементе суперглобального массива *$_GET['i']*);

```
{
$i=$_GET['i'];
}
```

include ("nov1.php"); – вставляем в наш код содержимое файла *nov1.php*. В том файле, как нам известно, содержится код, который формирует массив новостей *$lolo* из текстового файла и сортирует новости;

foreach($lolo as $line2) – функция *foreach* обходит массив, указанный в переменной *$lolo*, построчно (см. код программы *nov1.php* в листинге 5.7), т. е. выполняется цикл, причем столько раз, сколько строчек, т. е. записанных новостей, в массиве *$lolo*. При каждом выполнении цикла, переменной *$line2* присваивается содержимое очередной строки, которое потом будет выводиться в окно браузера;

1{ – начало цикла *foreach*. Весь приведенный ниже небольшой блок команд будет исполнен до закрывающейся фигурной скобки под номером 1. Здесь я снова нумерую попарно фигурные скобки, чтобы вы в них не запутались;

$j++; – вводим счетчик цикла для подсчета количества новостей или других записей. При каждом исполнении цикла он будет увеличиваться на единицу;

if(($j>$chislo$i-$chislo)&&($j<=$chislo*$i))* – условие, указывающее на интервал номеров выводимых новостей на страницу. Например, при *$i=1* условие будет истинно, если номера новостей (*$j*) будут соответствовать интервалу от 1 до 5. Переменная *$chislo*, если помните, равна 5. Это количество выводимых на страницу новостей. Переменная *$chislo* задана в файле *nov1.php*, а сам файл был вставлен в эту страницу при помощи оператора *include*;

2{

*echo $line2."

";* – если условие *if* истинно, новость выведется на страницу;

2} – окончание условия *if* для вывода новостей на страницу;

1} – окончание цикла *foreach*;

echo"</td></tr></table>"; – заканчиваем (закрываем) таблицу для новостей на странице;

echo " <p>"; – после таблицы с 5 новостями, мы должны разместить ссылки на другие таблицы с другими 5 новостями. Ссылки выводим в центре, в отдельном блоке <p>;

for($m=1; $m<=$j/$chislo+1; $m++) – при помощи цикла *for* выводим ссылки на остальные страницы с новостями. Страниц с новостями у нас будет несколько, но все они как бы будут объединены в одну. Будет меняться только параметр *$i* и в зависимости от этого параметра на страницу будет выводиться та или иная порция из пяти новостей. В переменной *$j* у нас хранится общее число всех новостей в массиве *$lolo*. Число *$j/$chislo+1* означает, сколько всего должно быть страниц с новостями. Отсюда счетчик $m означает у нас как бы номер страницы с новостями;

3{

if ($m==$i) – проверяется условие, которое я сейчас попытаюсь объяснить. Итак, на странице, например *new1.php?i=1*, будут ссылки на другие таблицы с новостями (на страницы с другим параметром *$i*). Нам нужно сделать так, чтобы не срабатывала бы ссылка на ту страницу, на которой находится посетитель. Например, если посетитель находится на второй странице (в этом случае *$i=$m=2*), то зачем ему ссылка на вторую страницу? Выведется только числовой номер страницы (выполнится команда между фигурными скобками под номером 4);

4{

echo "$m";

4}

else

5{

echo "$m";* – выводим ссылки на страницы с другими параметрами *$i, на которых посетитель еще не находится;

5}

3} – окончание цикла вывода ссылок;

echo "</p> "; – закрываем блок ссылок;

?> – конец PHP-кода;

Результат работы программы приведен на рис. 5.9. При помощи программы *redaktNovost.php* (листинг 5.6) я ввел 23 строки, т. е. 23 новости. На новостной странице находятся последние 5 новостей. На следующих страницах (с другим параметром *i*) находятся более старые новости (рис. 5.10).

Просмотрите другие ссылки на страницы с новостями и посмотрите в строке браузера, как при этом меняется параметр *i*.

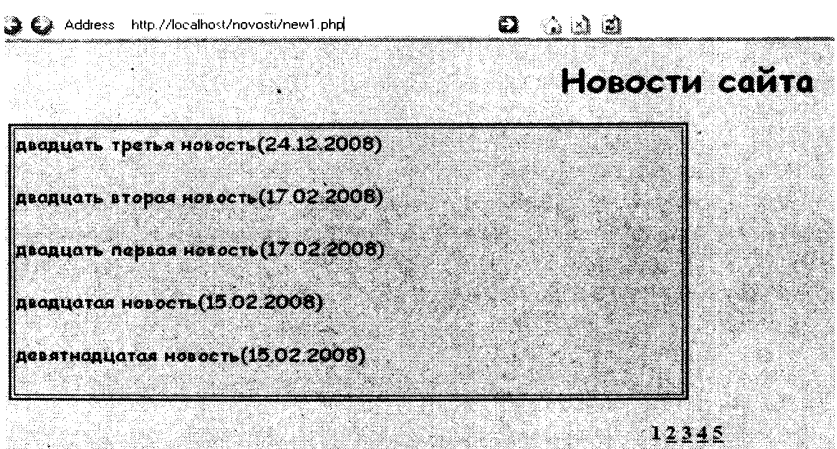


Рис. 5.9

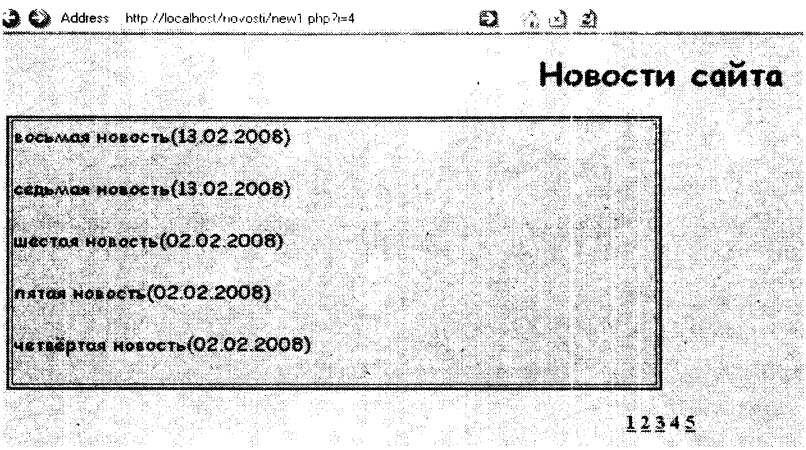


Рис. 5.10

Если вы разобрались в листинге 5.8, то должно быть понятно, что программа сначала добавляет новую запись в массив новостей, а затем все новости записываются разом в файл *novosti.txt*, полностью удаляя оттуда старую информацию. В конце ставится текущая дата. Я так сделал потому, чтобы вы могли не только записывать новые записи, но и редактировать старые при помощи того же интерфейса. Можно было бы упростить программы *new1.php* и *redaktNovost.php* так, чтобы вы только добавляли новые записи к старым, не перезаписывая текстовый файл с новостями. Нечто подобное мы рассмотрим в следующей главе, когда будем создавать книгу отзывов посетителей на странице со статьями.

И последнее, откройте нашу главную страницу сайта *index.php*. Нажмите на надпись (ссылку) «*Новости сайта*». Вы должны оказаться на странице с новостями *new1.php*. Если такого не произошло, проверьте, правильно ли указана ссылка (листинг 3.6). В нашем случае строка кода, указывающая на страницу с новостями, выглядит так:

*<A href="novosti/new1.php">Новости сайта</A>

.*

Если у вас путь к файлу *new1.php* другой, то измените его в этой строке.

Итак, страница с новостями создана. Вы можете оформить ее по своему усмотрению, например добавить баннеры, какие-нибудь картинки, рисунки.

Глава 6. ПОДБОРКА СТАТЕЙ НА САЙТЕ. ОЦЕНКА СТРАНИЦ ПОСЕТИТЕЛЯМИ

6.1. ОСНОВЫ БЕЗОПАСНОСТИ ДЛЯ САЙТА

В этой главе мы разработаем с вами еще одну страницу для сайта, на которую вы попадете, нажав на главной странице *index.php* надпись «Статьи». Создайте в рабочей папке *htdocs* подпапку *statji*. В эту подпапку вы будете загружать любые статьи. В браузер будут выводиться название статьи, автор статьи. Дополнительно мы создадим книгу отзывов посетителей и создадим возможность посетителям оценивать страницу по пятибалльной шкале. Программы для книги отзывов и оценки страницы создадим универсальные, т. е. их можно будет использовать на любых страницах вашего сайта.

Теоретические знания, которые нам понадобятся в этой главе, у вас уже должны быть, если вы разобрались с предыдущими главами. Однако кое-что нам нужно еще усвоить.

В книге отзывов каждый желающий может оставить на странице свое сообщение, которое будет потом отображаться в браузере. Он указывает свои данные (имя, электронный адрес) и записывает свое сообщение в текстовую область формы. Затем эти данные будут обрабатываться, и если текстовая область не была пустой, сообщение с указанными данными посетителя, будет выведено в окно браузера. Создание книги отзывов не вызовет у вас затруднений, но есть некоторые нюансы. Какой-нибудь недоброжелатель, вместо своего имени, адреса или сообщения может отослать вредоносный код, написанный на языке php, perl, и т. п. Поэтому данные, введенные пользователем, мы должны обработать, а именно, теги, которые посетитель может ввести, должны быть удалены. При помощи тегов злоумышленник может ввести в текстовые поля код на php, html, JavaScript, VBScript, который может нанести вред вашей странице или даже всему вашему сайту. Напишем, к примеру, небольшой код. Создадим форму с простым текстовым полем и кнопкой «добавить». После отправки данных из формы на эту же страницу содержимое текстового поля отобразится в браузере. В общем-то, безобидный на первый взгляд код листинга 6.1.

Листинг 6.1

```
<?php
$filen="test.txt";
if(!file_exists($filen))
{
 $open=fopen($filen, "w");
```

```
 fclose($open);
}
if(isset($_POST['imja']))
{
 $imja=$_POST['imja'];
 $open=fopen($filen, "a+");
 $rec=fwrite($open, "Здравствуй: ".$imja."<br>");
 fclose($open);
}
$open=fopen($filen, "a+");
$read=fread($open, 1000);
fclose($open);
echo $read;
echo "<form action=obychenie2.php method=POST>";
echo "Ваше имя: "."<input type=text name=imja><br>";
echo "<input type=SUBMIT value=Добавить>";
echo "</form>";
?>
```

Я лишь вкратце разъясню этот код. Он должен быть вам понятен. Сначала проверим, существует ли файл *test.txt* (можете назвать его, как хотите). Если нет, а при самом первом запуске программы это действительно так, то создаем его. Далее проверяется, отправлены ли данные из формы, т. е. была ли нажата кнопка «добавить». Если это так, то в текстовый файл *test.txt* записывается содержимое текстового поля, которое было введено в форме. Далее открываем этот файл, считываем и выводим в браузер это содержимое. Кстати, содержимое текстового файла будет после выводиться в браузер и до ввода данных в форму, поскольку даже если условие *if(isset($_POST['imja']))* ложно (мы только запустили эту программу, но не ввели еще данные в форму), код после последней закрывающейся скобки будет выполняться.

Сохраните в рабочей папке *htdocs* и запустите эту программу. В текстовое поле введите любое имя, например «*Иван*». Нажмите на кнопку «*добавить*». Веденное имя отбразится в браузере, как показано на рис. 6.2. Кстати, если вы используете сейчас компьютер, не подключенный к Интернету, то после нажатия кнопки в форме может появиться предупреждение, подобное на рис. 6.1.

Это может появиться при использовании метода отправки в теге формы *method=POST*. Если нажмете кнопку «*Повтор*», то данные будут отправлены повторно, в результате чего в текстовый файл эти данные будут записаны дважды, а в браузер выведется 2 раза имя «*Иван*». Лучше жмите «*Отмена*» для возврата к странице, чтобы данные из формы не отправлялись повторно.

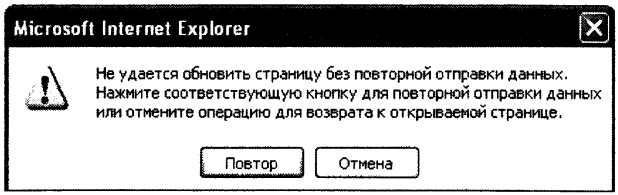


Рис. 6.1

Итак, после возврата к данной странице, вы увидите примерно так, как на рис. 6.2.

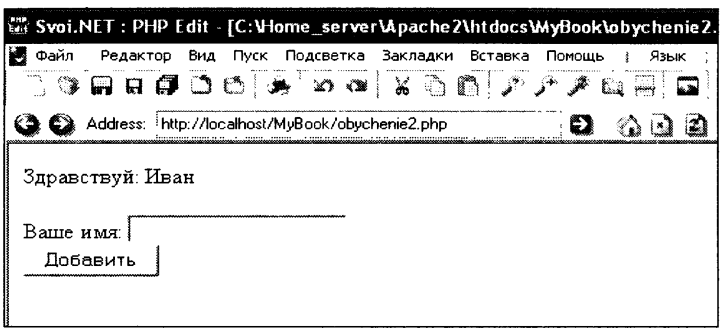


Рис. 6.2

Теперь представьте, что некий Пупкин введет в текстовое поле следующее:
<body bgcolor=black text=red>Пупкин</body>.

Нажмите на кнопку «Добавить». Вся ваша страница станет черной, цвет текста красным. Перезапустите программу. Увы, дизайн так и остался измененным, поскольку введенный HTML-код, который устанавливает цвет фона и шрифта, будет теперь выполняться при каждой загрузке этой страницы, и дизайн может быть испорчен. И это еще меньшее из зол, что может сделать злоумышленник.

Избежать подобных действий со стороны «товарища Пупкина» поможет функция *htmlspecialchars()*. Эта функция заменяет все теги и обратные слеши на безопасные буквенно-символьные выражения (обратные слеши используются для программирования на Perl). Измените переменную *$rec* в листинге 6.1 следующим образом:

*$rec=fwrite($open, "Здравствуй: ".htmlspecialchars($imja)."
");*

Снова запустите программу и опять введите в текстовое поле:
<body bgcolor=black text=red>Пупкин</body>.

В результате выполнения программы в окно браузера будет выведена приведенная выше строка. Теги выведутся тоже, но исполняться не будут! Если вы хотите, чтобы текст выводился определенного шрифта и цвета, ставьте нужные вам теги перед и после функции *htmlspecialchars()*. Например:

*$rec=fwrite($open, "Здравствуй: "."<B style='color: #000066; font-family: Arial'>".htmlspecialchars($imja)."</B>
");*

Перед функцией *htmlspecialchars($imja)* при помощи соединительного оператора конкатенации (точки), в кавычках вставлен тег *<B>*, который подразумевает жирный текст. Кроме того, в этом теге определен стиль текста (вид шрифта *Arial* темно-синего цвета). После функции обязательно нужно поставить закрывающий тег *</B>*.

Снова запустите данную программу, напишите любое имя в текстовое поле формы и опять нажмите кнопку «*Добавить*». Имя будет выведено шрифтом *Arial*, темно-синего цвета.

Функцию *htmlspecialchars()* мы будем использовать и в дальнейшем, чтобы обеспечивать безопасность сайта от вводимых пользователями данных. Есть еще другая функция *strip_tags()*, которая, в отличие от предыдущей функции, не заменяет, а удаляет все теги. Мы ее тоже будем применять на практике.

6.2. ПРИМЕНЕНИЕ СЛОЖНОЙ СХЕМЫ БЛОКИРОВКИ ФАЙЛА. ФУНКЦИИ ДЛЯ РАБОТЫ С КАТАЛОГАМИ

Теперь о другой проблеме. В книге посетителей при записи отзывов посетителей в текстовый файл мы применим функцию блокировки этого текстового файла, чтобы избежать одновременного использования программы для записи в файл несколькими посетителями, что привело бы к разрушению информации в файле (о значении блокировки файла см. в гл. 4). Однако применять блокировку в таком простом виде, которую мы применяли при создании счетчика посещений страницы, не годится. Допустим, два разных посетителя заполнили поля формы книги отзывов и одновременно нажали на кнопку типа «*Отправить*». Данные первого посетителя, который оказался проворнее на микросекунду, отправятся программе для обработки. Эта программа временно заблокирует текстовый файл для записи отзывов. Файл заблокируется всего на микросекунды, но если в это время придут данные от второго посетителя, то программа не сможет их записать в этот текстовый файл. Данные второго посетителя могут быть утеряны и ему снова придется их вводить, т. е. заполнять форму заново. Здесь нужно будет применить другую схему блокировки, которая будет разъяснена чуть позже в гостевой книге, при разборке листинга 7.3 программы *gostev.php*.

В этой главе нам еще понадобятся функции для работы с каталогами, т. е. папками. Прежде чем работать с какой-либо папкой, ее нужно сначала открыть. Для этого существует функция opendir(dir), где в качестве аргумента dir содержиться путь к открываемому каталогу. Эта функция возвращает идентификатор, с которым и нужно работать в дальнейшем. Функция readdir($iden) считывает содержимое каталога и возвращает имена всех элементов в каталоге. Здесь $iden – идентификатор открытого каталога. Мы будем использовать функцию readdir для вывода списка файлов и папок, содержащихся в указанном в ее параметре каталоге. При каждом своем вызове она выдает имя случайно выбранного файла (или вложенной папки) указанного каталога, каждый раз – новое, до тех пор, пока не перечислит имена всех файлов и вложенных папок. Среди выданных функцией readdir имен будут и ссылки на текущий и родительский (т. е. включающий в себя текущий) каталог, обозначаемые соответственно одной и двумя точками (так уж работает web-сервер). Поскольку нас будут интересовать только файлы каталога, то данные ссылки из списка файлов следует исключить, добавив проверку состава имени файла:

```
while ($file=readdir($iden))
{
if (($file!=".") and ($file!="..")) $c[]=$file;
}
```

В этих строчках кода цикл *while* будет выполняться, пока не будут перебраны все элементы каталога, задаваемые идентификатором *$iden*. Если элементы не являются ссылками на текущий и родительский каталог (условие *if* истинно), то они поочередно будут записываться в массив *$c[]*.

Вообще вместо этих двух функций (*opendir* и *readdir*) можно использовать одну *scandir(dir, sort)*, которая возвращает массив имен и каталогов, расположенных в каталоге *dir*. Обратите внимание, что в качестве первого аргумента используется именно каталог или путь к нему, но не идентификатор! Параметр *sort* указывает на порядок сортировки массива. Если этот параметр не указан, то сортировка выполняется в алфавитном порядке по возрастанию. Если же параметру *sort* присвоить значение 1, то сортировка выполнится по убыванию.

В нашей рабочей папке сервера создайте временно подпапку, например *risynki*. Скопируйте в эту подпапку несколько любых рисунков или фотографий с вашего компьютера. В PHP-редакторе наберите код:

Листинг 6.2

```
<?php
$iden=opendir("risynki");
while ($file=readdir($iden))
```

```
{
 if (($file!=".") and ($file!=".."))
 {
  echo "<a href=risynki/$file target='_blank'>$file</a><br>";
 }
}
?>
```

Строчки приведенного выше кода должны быть понятны. В окно браузера будут выведены ссылки на ваши рисунки в папке *risynki*. Причем при переходе на ссылку рисунок откроется в отдельном окне, поскольку в теге *<a href...>* мы указали параметр *target=_blank*. Запомните его. Если вы разместите на своем сайте ссылки на другие сайты, то эти другие сайты при использовании параметра *target=_blank* будут открываться в отдельном окне.

Сохраните программу обязательно в рабочей папке *htdocs* под любым названием и запустите ее. В браузер, как уже было отмечено, выведутся названия-ссылки на рисунки. Нажмите на любую из ссылок – в отдельном окне появится изображение соответствующей картинки (рис. 6.3). У меня, например, в папке *risynki* размещено 12 картинок.

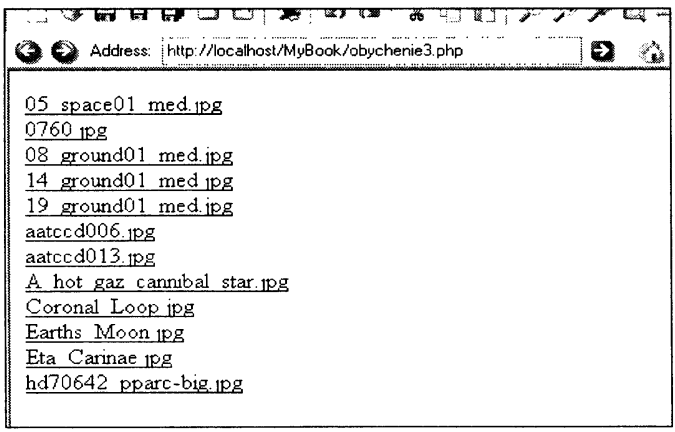


Рис. 6.3

Вообще нежелательно, чтобы названия файлов и папок начинались с нуля, так как функция *readdir* может их проигнорировать! Можете теперь удалить эту программу и папку *risynki*, чтобы не засорять наш рабочий каталог.

6.3. СОЗДАНИЕ СБОРНИКА СТАТЕЙ И ВЫВОД ИХ НА САЙТ

Итак, допустим. вы являетесь администратором сайта, на котором размещаются статьи разных авторов. В таком случае при поступлении новой статьи, если не прибегать к помощи php, вам приходится, помимо размещения на web-сервере (или выделенном вам аккаунте) ее самой, еще и обновлять страницу со списком этих статей, добавив ссылку на новоразмещенную web-страницу со статьей – иначе ведь попасть на новую статью с сайта будет невозможно. А если статьи поступают часто? Да еще и не только поступают, но и удаляются, или в них меняется название? Тогда ведь для отслеживания правильности содержания приходится прилагать немало усилий. Создадим программу, которая автоматизировала бы этот процесс. В рабочей папке *htdocs* вы должны были создать подпапку *statji*. А в этой подпапке создайте еще 6 папок с названиями *1, 2, 3, 4, 5, 6*, как на рис. 6.4.

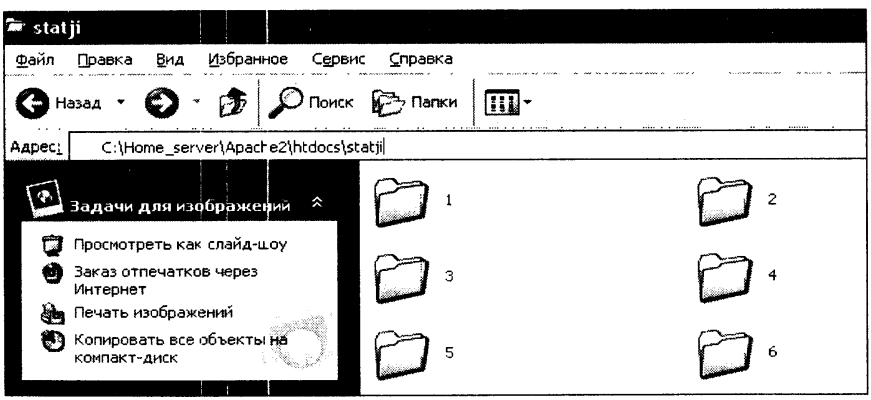


Рис. 6.4

В каждой из шести папок будет находиться по статье в виде web(html)-документа. Вы спросите, почему я каждую статью размещаю в отдельной папке? Да потому, что к каждой статье могут прилагаться рисунки. Если все «слить» в одну общую папку, то возникнет путаница. А так у нас каждая статья и прилагающиеся к ней рисунки в отдельной папке. В браузер будут выводиться ссылки на web-страницы со статьями, находящимися в разных папках. Однако для составления списка статей информации только об именах файлов мало. Ведь в таком списке желательно указать хотя бы название статьи и имя ее автора. Чтобы это сделать, можно, например, указывать эти данные в тегах *meta*, включаемые в каждый файл со статьей. Эти теги располагают между тегами заголовочной части *<head>* и *</head>* HTML-документов. Для данного примера я скачал из Интернета шесть HTML-доку-

ментов с различными статьями. Первый HTML-документ со статьей я разместил в первой папке и назвал его *statja1.html*. Второй HTML-документ со статьей я разместил во второй папке и назвал его *statja2.html* и т.д. Вы можете проделать то же самое. В каждом из HTML-файлов со статьей разместим *meta*-теги, примерно так:

```
<head>
<title>Статья</title>
<meta name="nazvanie" content="Название статьи">
<meta name="avtor" content="Автор статьи">
</head>
```

Содержимое данных тегов нужно указать в программе, где мы будем выводить названия статей, с помощью функции *get_meta_tags* (*имя файла со статьей*). Функция *get_meta_tags* создает массив, элементы которого соответствуют названиям мета-тегов (*name*), которые были указаны в файле со статьей (если, конечно, такие meta-теги там есть), а значения этих элементов – соответствующим значениям мета-тегов (*content*). Строки сценария, вытаскивающие из файла содержимое этих тегов и помещающие их на страницу, будут выглядеть так:

```
$statji="statji/$i";
$meta=get_meta_tags("$statji/$file");
echo "<a href=$statji/$file target='_blank'>$meta[nazvanie]</a>
$meta[avtor]<br>";
```

В переменной *$statji* будет находиться путь к одной из шести папок. В переменной *$file* – название файла со статьей в какой-либо из шести папок. Устанавливаем ссылку на этот файл, выводим название статьи, которое будет в переменной *$meta[nazvanie]* и (уже без ссылки) автора статьи из переменной *$meta[avtor]*. В реальной программе, в листинге 6.3, мы все эти названия и авторов разместили для удобства в ячейках таблицы.

После таблицы со статьями разместим книгу отзывов о странице и форму для оценки по пятибалльной шкале. Постарайтесь набрать код сами, после того как в нем разберетесь. Разбор кода сразу после листинга 6.3. В PHP-редакторе создайте новый файл и сохраните его в рабочей папке *htdocs* под названием *statji.php*.

Листинг 6.3 (файл statji.php)

```
<html>
<head>
 <title>Подборка статей</title>
 <link rel="stylesheet" type="text/css" href="stil.css">
</head>
```

```
<body bgcolor="cyan">
<center><table width="80%" border="2" bordercolor="green"><tr>
<td align="CENTER" id="lolo3">Название статьи</td>
<td align="CENTER" id="lolo3">Автор статьи</td></tr>
<?php
for($i=1; $i<=100; $i++)
{
 $statji="statji/$i";
 if(!file_exists($statji)) break;
 $dir=opendir($statji);
 $raz=array(".html", ".htm", ".shtml", ".php");
 while(($file=readdir($dir))!==false)
 {
  if(($file !=".")&&($file !=".."))
  {
   if(in_array(strrchr($file, "."), $raz))
   {
    $meta=get_meta_tags("$statji/$file");
    echo "<tr><td id=lolo><a href=$statji/$file
target='_blank'>$meta[nazvanie]</a></td><td id=lolo>$meta[avtor]</td></tr>";
   }
  }
 }
closedir($dir);
}
echo "</table></center><br>";
$ind=basename($_SERVER['PHP_SELF'], ".php");
include ("soobshenie.php");
include ("ocenka.php");
?>
</body>
</html>
```

Разберем этот код:

<html>

<head>

*<title>*Подборка статей*</title>*

<link rel="stylesheet" type="text/css" href="stil.css"> – указываем лист стилей, который будем использовать. Файл листа стилей *stil.css* мы создали раньше. Что это такое и для чего он нужен, было рассказано в гл. 3;

</head>

<body bgcolor="cyan"> – устанавливаем цвет фона страницы;

* <table width="80%" border="2" bordercolor="green"><tr>* – создаем двухколонную таблицу, открываем первую строку, которая будет заголовочной;

<td align="center" id="lolo3">Название статьи</td> – в центре первой ячейки первой строки выводим текст «*Название статьи*». Идентификатор *lolo3* задает тип и цвет шрифта для вводимого в ячейку текста. Этот идентификатор и его параметры должны быть указаны в файле стилей *stil.css*. Если такого идентификатора там нет, то создайте его (как это сделать, см. в гл. 3) или здесь вместо *lolo3* поставьте другой, который там есть;

<td align="center" id="lolo3">Автор статьи</td></tr> – аналогично во второй ячейке верхней строки выводим текст «Автор статьи»;

<?php – начало PHP-кода;

for($i=1; $i<=100; $i++) – начало цикла, счетчиком которого является переменная *$i*. Данный счетчик пробегает папки со статьями. Мы не зря дали имена шестерым папкам в виде последовательности чисел (1, 2, 3...). Их так легче считать. Если общее число статей у вас не будет превышать 100, то можно оставить *$i<=100*. У нас пока шесть статей, но начиная с *$i=7* цикл в холостую работать не будет, ибо мы его прервем;

1{ – открывается скобка начала цикла. Для удобства скобки попарно нумеруются;

$statji="statji/$i"; – присваиваем переменной *$statji* (назовите как угодно) путь к папке со статьей (например, при *$i=1* переменной *$statji* будет присвоен путь к первой папке, т. е. *$statji=statji/1*);

if(!file_exists($statji)) break;* – проверяется условие, существует или нет такая папка (такой путь к папке). Если такой папки еще нет, например при *$i>6, то цикл *for* прерывается. Если папка существует, то выполнение цикла продолжится дальше;

$dir=opendir($statji); – открываем папку со статьей. Например, при *$i=1* откроится первая папка, при *$i=2* будет открыта уже вторая папка со второй статьей. Открытой папке присваивается идентификатор *$dir*;

$raz=array(".html", ".htm", ".shtml", ".php");* – создаем массив *$raz с расширениями web-файлов со статьями. Зачем это нужно, увидите через пару строк кода. Статьи могут содержаться не только в HTML-файлах, но и в файлах с расширением *htm, shtml, php*. Я думаю, этого достаточно;

while(($file=readdir($dir))!==false) – начинается цикл, суть которого была рассмотрена в разд. 6.1–6.2. При каждом выполнении этого цикла переменной *$file* присваивается какой-нибудь файл или каталог, содержащийся в открытой на данный момент папке (при *$i=1* будет открытая первая папка);

2{

if(($file !=".")&&($file !="..")) – как отмечалось выше, среди выданных функцией *readdir* имен будут и ссылки на текущий и родительский каталоги. Если переменная *$file* не содержит текущий или родительский каталог, то выполнится блок команд за открывающейся фигурной скобкой под номером 3;

3{

if(in_array(strchr($file, "."), $raz)) – снова проверка условия. Дело в том, что в папке, где содержится файл со статьей, могут также содержаться и другие файлы и каталоги, на которые могут указывать ссылки из данной статьи (картинки, рисунки и т.п.). Мы должны вывести в браузер ссылки только на файлы со статьей, остальные файлы отсечь. Функция *strchr*, как отмечалось в гл. 4, возвращает часть строки, начиная с символа «.» и до конца строки. Строка находится в переменной *$file*. Это название файла с расширением в открытой на данный момент папке. Данная функция возвратит только то, что следует от точки в названии файла, т. е. его расширение, включая точку вначале. Функция *in_array*, напомню, ищет значение, полученное при помощи функции *strchr*, в массиве *$raz* и возвращает *true*, если это значение найдено. Иными словами, если расширение файла, содержащегося в переменной *$file*, совпадет хотя бы с одним элементом массива *$raz (".html", ".htm", ".shtml", ".php")*, то общее выражение в круглых скобках после оператора *if* будет истинно и выполнится блок команд за открывающейся фигурной скобкой под номером 4;

4{ – этот блок команд внизу выполнится, если выбранный файл имеет расширение либо *html*, либо *htm*, либо *shtml*, либо *php*, т. е. файл со статьей;

$meta=get_meta_tags("$statji/$file"); – вытаскиваем из файла содержимое тегов <*meta*>. Эти теги должны быть в каждом файле со статьей;

echo "<tr><td id=lolo>$meta[nazvanie] </td><td id=lolo>$meta[avtor]</td></tr>"; – при помощи оператора echo выводим в браузер содержимое тегов <meta>. Сначала открываем новую строку таблицы. В первую ячейку новой строки выводится ссылка на файл со статьей и содержимое тега <meta> с именем name=nazvanie. В нем должно содержаться название статьи. Далее действие ссылки заканчивается (закрывающийся тег). Во вторую ячейку новой строки выводится содержимое тега <meta> с именем name=avtor, т. е. указывается автор данной статьи;

4} – заканчивается блок команд в фигурных скобках под номером 4;

3} – заканчивается блок команд в фигурных скобках под номером 3;

2} – заканчивается блок команд в фигурных скобках под номером 2, т. е. как только заканчиваются файлы в открытой папке, цикл *while* завершается;

closedir($dir); – закрываем открытую папку;

1} – завершается самый первый цикл *for* (см. начало кода) для данного значения переменной *$i*. После переменная *$i* увеличится на единицу и цикл снова весь повторится, если, конечно, папка со статьей с таким номером *$i* существует;

*echo "</table>
";* – закрываем таблицу. Все, таблицу с названиями статей и их авторов вывели. Далее выводим книгу отзывов и форму для оценки;

$ind=basename($_SERVER['PHP_SELF'], ".php"); – элемент суперглобального массива;

$_SERVER['PHP_SELF'] возвращает полный путь к текущей странице, где находится данный код (*statji.php*), а функция *basename* урезает его до названия страницы без расширения (*statji*). Все эти функции были рассмотрены в гл. 4. Переменной *$ind* присваиваем название данной страницы с кодом (*statji).* Она будет использоваться в кодах для книги отзывов *soobshenie.php* и для формы оценки *ocenka.php*. Эти коды мы напишем позже и сделаем их универсальными, т. е. вы можете вставлять их на любые страницы сайта (на разных страницах и переменная *$ind* будет различной);

include ("soobshenie.php");

include ("ocenka.php"); – вставляем коды страниц, которые напишем чуть позже. Пока их еще нет.

Последние 2 строчки лучше пока временно удалить и обратно вставить их после написания всех программ, иначе браузер будет выдавать ошибку!

6.4. СОЗДАНИЕ КНИГИ ОТЗЫВОВ ПОСЕТИТЕЛЕЙ САЙТА

Теперь напишем код для книги отзывов. В браузер будет выведена одноколонная таблица с последними 10 отзывами (можете сделать больше или меньше). Далее расположится форма с текстовыми полями для ввода данных посетителя. Ну и потом следует текстовая область для ввода отзыва и кнопка для запуска программы обработки данных. Сделает все это программа *soobshenie.php*. На рис. 6.5 изображена часть страницы *statji.php*, а именно, то место где расположены отзывы, поля для ввода данных от посетителя и текстовая область с кнопкой для ввода и отправки отзыва.

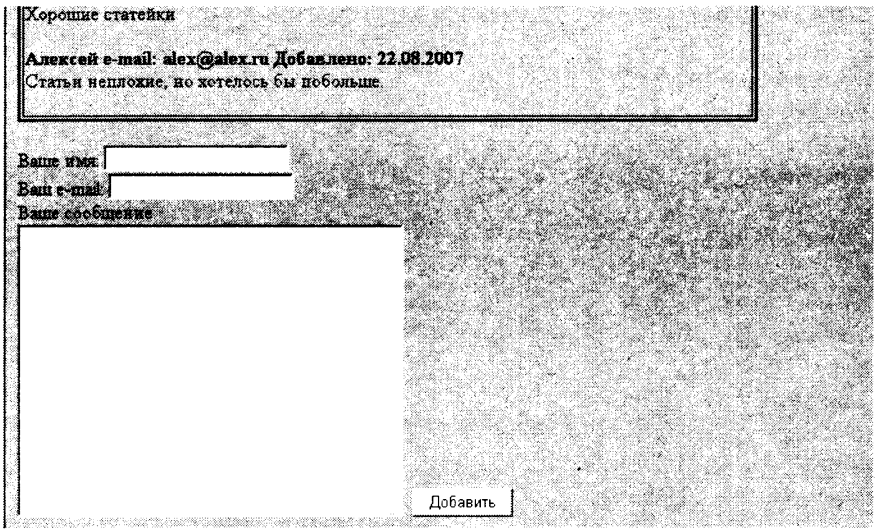


Рис. 6.5

Но сначала в рабочей папке *htdocs* создайте новую подпапку *otzivi*. Там у нас будут храниться текстовые файлы отзывов. В уже знакомом нам PHP-редакторе создайте новый файл и сохраните его под названием *soobshenie.php*. Вообще можете назвать этот файл как угодно, например *otziv.php*. Естественно, и вставлять его на страницы нужно будет под этим именем. Но я назвал файл вывода отзывов *soobshenie.php*. С этим именем, при помощи оператора *include*, я его и вставил на страницу *statji.php*. Код вам должен быть уже понятен. Данная программа (листинг 6.4) осуществляет вывод в колонку отзывов, сохраненных в обыкновенном текстовом файле, выводит форму для ввода данных пользователем и обрабатывает эти данные.

Листинг 6.4 (файл soobshenie.php)

```
<html>
<head>
 <title>Сообщения</title>
</head>
<body>
<div id="lolo3">Здесь вы можете оценить подборку и оставить свое
сообщение</div><br>
<table width="50%" border="2" bordercolor="green" bgcolor="silver"><tr><td>
<?php
$filen="otzivi/$ind".".txt";
$chislo=10;
```

```
if(!file_exists($filen))
{
$open=fopen($filen, "w");
fclose($open);
}
$lolo=file($filen);
krsort($lolo);
foreach($lolo as $line2)
{
 $j++;
 if($j<=$chislo)
 {
  echo $line2."<br><br>";
 }
}
echo "</td></tr></table>";

if(isset($_POST['otziv']))
{
 $ind=$_POST['ind'];
 $filen="otzivi/$ind".".txt";
 $otziv=$_POST['otziv'];
 $data=date('d.m.Y');
 $imja=$_POST['imja'];
 $adress=$_POST['adress'];
 $nov=fopen($filen, "a");
 $danie=strip_tags($imja)." e-mail: ".strip_tags($adress)." Добавлено: ".$data;
 fwrite($nov, "<B
style=color:#000066>".$danie."</B><br>".strip_tags($otziv).chr(13).chr(10));
 fclose($nov);
 echo "<a href=# onClick='history.back()'><button>Вернуться</button></a>";
}
else
{
 echo "<form action=soobshenie.php method=POST>
 Ваше имя: <input type=text name=imja><br>
 Ваш e-mail: <input type=text name=adress><br>
 Ваше сообщение<br>
 <textarea name=otziv cols=50 rows=15 wrap=virtual></textarea>
 <input type=hidden name=ind value=$ind>
```

```
<input type=SUBMIT value=Добавить>
</form>";
}
?>
</body>
</html>
```

Я объясню лишь некоторые строки кода:

* Здесь вы можете оценить подборку и оставить свое сообщение
* – выводим надпись в контейнере * *. Идентификатор *lolo3* задает тип и цвет шрифта надписи. Этот идентификатор и его параметры должны быть указаны в файле стилей *stil.css*. А сам файл стилей указан на странице *statji.php*, куда будет вставлен этот код;

<table width="50%" border="2" bordercolor="green" bgcolor="silver"><tr><td> – делаем одноколонную таблицу, задаем ее цвет линий и фона;

$filen="otzivi/$ind".".txt"; – присваиваем переменной *$filen* путь к текстовому файлу отзывов. Он у нас будет находиться в папке *otzivi*. Название этого файла будет соответствовать названию той страницы, куда мы вставим этот код. Название страницы без расширения у нас в переменной *$ind*. Эта переменная у нас определена на странице *statji.php*. А поскольку данный код мы вставляем именно на ту страницу, то переменная *$ind* будет определена и в этом коде. Расширение *«.txt»* мы присоединяем к названию файла при помощи точки или оператора конкатенации. Само расширение тоже, естественно, должно начинаться с точки. В итоге в нашем случае текстовый файл с отзывами будет называться *statji.txt*;

$chislo=10; – в этой переменной будем хранить число вывода последних отзывов в браузер. Можете изменить его на свое усмотрение;

if(!file_exists($filen)) – если файла с отзывами еще нет (данный код запускается впервые), то он будет создан (следующие две строки в фигурных скобках под номером 1);

```
1{
$open=fopen($filen, "w");
fclose($open);
1}
```

$lolo=file($filen); – при помощи функции *file* создаем массив строк (отзывов) и присваиваем этот массив переменной *$lolo*;

krsort($lolo); – сортируем массив в обратной последовательности, чтобы более свежие отзывы были наверху;

foreach($lolo as $line2) – далее идет известный вам цикл, который построчно выводит 10 отзывов в окно браузера. Принцип его работы я объяснять не буду, поскольку это уже было сделано в прошлых главах;

```
2{
$j++;
if($j<=$chislo)
3{
echo $line2."<br><br>";
3}
2}
```

echo "</td></tr></table>"; – закрываем таблицу с отзывами посетителей;

if(isset($_POST['otziv'])) – далее идет условие. Мы проверяем, получен ли элемент суперглобального массива *$_POST['otziv']*. Дело в том, что именем *otziv* мы назвали текстовую область формы (форма создается ниже, в конце этого же кода), в которую посетитель будет вводить новый отзыв. Программный код в фигурных скобках под номером 4 выполнится только тогда, когда посетитель записал свой отзыв и нажал на кнопку формы «*Добавить*». Если кнопка еще не нажата и данные формы не отправлены, этот программный код (в фигурных скобках под номером 4) выполняться не будет!

```
4{
```

$ind=$_POST['ind']; – здесь мы переменной *$ind* присваиваем значение элемента суперглобального массива *$_POST['ind']*. Вы спросите, а зачем мы это делаем, ведь этой переменной мы присвоили название файла, куда мы вставляем весь этот код листинга 6.4? Дело в том, что при нажатии кнопки «*Добавить*» посетитель перейдет снова на эту же страницу (*soobshenie.php*), поскольку в теге формы *<form>* (см. окончание этого кода) мы указали именно эту страницу, а не *statji.php*, где мы определили переменную *$ind*. Получается, что после нажатия кнопки «*Добавить*», переменная *$ind* опять будет не определена. Но до нажатия кнопки эта переменная еще будет иметь значение *statji*. Как же это значение вновь передать на эту же страницу *soobshenie.php* после нажатия кнопки? Сделать это просто. В форме у нас есть скрытое поле (*type=hidden)* с именем *name=ind*. С помощью этого поля в параметре *value* мы и передаем при отправке данных значение *statji* на эту страницу. Как и все передаваемые из формы данные, это значение помещается в суперглобальный массив *$_POST* (в теге формы был выбран метод передачи *method=POST)*. И поскольку мы присвоили скрытому полю имя *ind*, значение *statji* будет находиться в элементе этого массива *$_POST['ind']*. Вот и все, теперь значение *statji* мы опять присваиваем как бы вновь созданной переменной *$ind*;

$filen="otzivi/$ind".".txt"; – снова присваиваем переменной *$filen* путь к текстовому файлу отзывов;

$otziv=$_POST['otziv']; – переменной *$otziv* присвоено значение суперглобального массива *$_POST['otziv']*, в котором находится введенный посетителем отзыв;

$data=date('d.m.Y');* – в переменной *$data сохраняем текущую дату в формате: *день.месяц.год*;

$imja=$_POST['imja']; – в переменной *$imja* сохраняем содержимое текстового поля с именем *name=imja*, куда посетитель вводит имя;

$adress=$_POST['adress']; – в переменной *$ adress* сохраняем содержимое текстового поля с именем *name=adress*, куда посетитель вводит свой e-mail;

$nov=fopen($filen, "a"); – итак, данные из формы получены. Теперь их надо записать в текстовый файл для отзывов. Открываем файл для записи. Второй аргумент (*"a"*) функции fopen говорит о том, что новые данные будут записываться в конец файла. Открытому файлу присваиваем идентификатор *$nov*;

$danie=strip_tags($imja)." e-mail: ".strip_tags($adress)." Добавлено: ".$data; – создаем строку с полученными данными от посетителя и датой отправки. Эту строку мы добавим перед отзывом. В начале строки ставим введенное посетителем имя. Функция *strip_tags* удалит все теги и слэши, если вдруг они были введены посетителем. Зачем это нужно, объяснено в разд. 6.1. Затем, используя точку (оператор конкатенации) и кавычки для ввода текста, ставим введенный пользователем e-mail. После этого добавляем в эту строку текущую дату;

*fwrite($nov, "<B style=color:#000066>".$danie." </B>
".strip_tags($otziv).chr(13).chr(10));* – используя функцию *fwrite*, записываем в текстовый файл для отзывов то, что написано во втором аргументе этой функции. А второй аргумент у нас длинный (в одну строчку не уместился). На самом деле ничего сложного в нем нет. Созданную в прошлой программной строке переменную *$danie* помещаем между тегами *<B>* и *</B>*, которые задают тип и цвет шрифта. Это сделано для красоты. Цвет и шрифт можете задать свой. Оператор *
* отделяет строку *$danie* от текста отзыва, который у нас в переменной *$otziv*. Этот текст тоже нужно обработать функцией *strip_tags*. Далее идут 2 символа *chr(13)* и *chr(10)*. Вместе они обозначают перевод строки. Нам нужно, чтобы отзыв очередного посетителя начался с новой строки. Вместо данных символов можно применить символ «\n», как мы это делали при записи в файл новостей в прошлой главе, но использование комбинации из двух символов *chr(13)* и *chr(10)* более предпочтительно;

fclose($nov); – после записи закрываем текстовый файл;

echo "<button>Вернуться</button>"; – вывод в браузер кнопки для возврата на предыдущую страницу. Запомните эту строку кода. После того как мы нажмем кнопку «*Добавить*» для отправки данных в форме, мы попадем на эту же страницу *soobshenie.php*. Но нам же нужно вернуться опять на страницу *statji.php*. Для этого и существует кнопка «*Вернуться*». Данная команда универсальна. С какой бы страницы вы ни пришли, т. е. в какую бы страницу вы ни вставили бы данный код *soobshenie.php*, вы все равно вернетесь на предыдущую страницу;

4} – конец блока команд, который будет выполняться только тогда, когда в форме нажата кнопка «*Добавить*»;

else – если кнопка «*Добавить*» еще не нажата, то просто загрузится форма для ввода данных и отзыва посетителя. Эта форма появится сразу после загрузки страницы со статьями *statji.php*, поскольку страница с этим кодом (*soobshenie.php*) будет вставлена в нее при помощи оператора *include* (см. листинг 6.3);

{

echo "<form action=soobshenie.php method=POST> – далее выводится форма. Смысл вводимых элементов (2 текстовых поля для ввода имени и адреса, текстовой области для записи отзыва, скрытого поля и кнопки «*Добавить*») разобран выше в этом коде.

*Ваше имя: <input type=text name=imja>
*

*Ваш e-mail: <input type=text name=adress>
*

*Ваше сообщение
*

<textarea name=otziv cols=50 rows=15 wrap=virtual></textarea>

<input type=hidden name=ind value=$ind>

<input type=SUBMIT value=Добавить>

</form>";

}

?>

</body>

</html>

6.5. ПРОГРАММА ДЛЯ ОЦЕНКИ ПОСЕТИТЕЛЯМИ WEB-СТРАНИЦЫ

Ну и теперь составим программу (*ocenka.php*) для оценки посетителем вашей страницы по пятибалльной шкале. Эта программа выводит небольшую форму, состоящую из выпадающего списка с оценками (от 1 до 5) и кнопкой «*Оценить*», при нажатии на которую запускается эта же программа

(*ocenka.php*), которая высчитывает средний балл и выводит его в окно браузера (рис. 6.6).

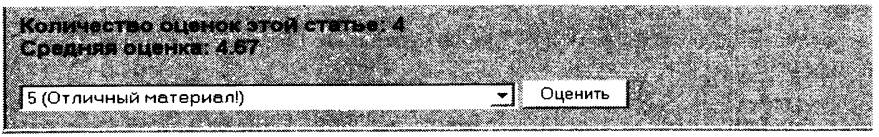


Рис. 6.6

В окно браузера мы выведем также и общее количество оценок, т. е. сколько раз эта страница была уже оценена.

Сначала в рабочей папке сервера *htdocs* создайте подпапку *ocenka*. Там у нас будет храниться текстовый файл (он будет создан программно) с количеством оценок и средней оценкой.

Программа несложная, и если вы разобрались с программой второго варианта счетчика (см. листинг 4.7), то разобраться с приведенным внизу листингом (6.5) не составит труда. Создайте в редакторе новый файл и сохраните его в рабочей папке *htdocs* под названием *ocenka.php*.

Листинг 6.5 (файл ocenka.php)

```
<?php
$dir="ocenka";
$ocenka=$ind.".txt";
$way="$dir/$ocenka";
if(!isset($_POST['ball']))
{
 if(file_exists($way)==true)
 {
  $file=file($way);
  $vsego=(int)$file[0];
  $srednee=(float)$file[1];
  echo "<B id=lolo2>Количество оценок этой статье: $vsego</B><br>";
  echo "<B id=lolo2>Средняя оценка: $srednee</B><br>";
 }
 echo "<form method=POST action=ocenka.php>
 <input name=ind type=HIDDEN value=$ind>
 <select name=ball>
 <option value=5>5 (Отличный материал!)</option>
 <option value=4>4 (В общем, хорошо!)</option>
 <option value=3>3 (Неплохо, но можно было бы и лучше!)</option>
```

```
<option value=2>2 (Плохо, зря я только зашел сюда!)</option>
<option value=1>1 (Лучше бы ты землю пахал, чем делал бы сайты!)</option>
</select>
<input name=submut type=SUBMIT value=Оценить>
</form>";
}
else
{
$ball=$_POST['ball'];
$ind=$_POST['ind'];
$ocenka=$ind.".txt";
$way="$dir/$ocenka";
if(file_exists($way)!=true)
{
$vsego=1;
$srednee=$ball;
$count=$vsego."\n".$srednee;
$open=fopen($way, "w+");
fwrite($open, $count);
fclose($open);
}
else
{
$file=file($way);
$vsego=(int)$file[0];
$srednee=(float)$file[1];
$vsego++;
$srednee=($srednee*($vsego-1)+$ball)/$vsego;
$srednee=round($srednee, 2);
$count=$vsego."\n".$srednee;
$open=fopen($way, "r+");
flock($open, LOCK_EX);
fwrite($open, $count);
flock($open, LOCK_UN);
fclose($open);
}
echo "<div id='lolo'>Благодарим вас за оценку</div><br>";
echo "<a href=# onClick='history.back()'><button>Вернуться</button></a>";
}
?>
```

Разберем строчки кода. Многие из них вам должны быть уже понятны.

<?php

$dir="ocenka";* – присваиваем переменной *$dir название папки, где будет храниться текстовый файл с количеством оценок и средней оценкой. Эта папка (*ocenka*) должна быть вами создана заранее;

$ocenka=$ind.".txt"; – переменной *$ocenka* (не путать с названием папки!) присваиваем название текстового файла с количеством оценок и средней оценкой. Переменная *$ind* у нас определена (пока не нажата кнопка «*Оценить*») на странице *statji.php*. Текстовый файл будет иметь такое же название, но, естественно, с расширением *txt*. Не пропустите точки! В этой строке их две: первая – знак конкатенации или соединения, вторая внутри кавычек (расширение любого файла начинается с точки);

$way="$dir/$ocenka"; – переменной $way присваиваем путь к текстовому файлу. Вообще можно было 3 строчки соединить в одну *($way="ocenka"/$ind.".txt")*, а можно написать так, как я. Выбирать вам;

if(!isset($_POST['ball'])) – проверяется условие, которое истинно, если элемента суперглобального массива *$_POST['ball']* еще не существует, т. е. кнопка в форме «*Оценить*» еще не нажата и данные из формы еще не получены. Именем *name=ball* мы назвали выпадающий список *select* (сама форма будет выводиться несколькими строчками ниже);

1{ – итак, код между фигурными скобками под номером 1 будет выполняться, если кнопка «*Оценить*» еще не нажата;

if(file_exists($way)==true) – внутри первого условия ставим второе, которое истинно, если текстовый файл с количеством оценок и средним баллом уже создан, а создается он при первой же оценке (смотри этот код чуть ниже);

2{ – итак, код между фигурными скобками под номером 2 будет выполняться, если текстовый файл уже был создан;

$file=file($way); – функция *file*, если помните, создает массив строк из содержимого текстового файла (путь к нему находится в переменной *$way*). В текстовом файле (он будет создан ниже) будет две строки. В первую строку будет записано общее число оценок, во вторую – средний балл. Таким образом, в переменной *$file* (не путать с функцией *file*!) будет простой массив с двумя элементами;

$vsego=(int)$file[0]; – присваиваем переменной *$vsego* нулевой элемент массива *$file*. Оператором *(int)* мы указываем, что нулевой элемент – целочисленное число. Это действительно так, поскольку общее количество оценок не может быть дробным числом;

$srednee=(float)$ file [1]; – присваиваем переменной *$srednee* первый элемент массива *$file*. Оператором *(float)* мы указываем, что первый элемент – деся-

тичное число. Это действительно так, поскольку средний балл у нас будет дробным числом;

*echo "<B id=lolo2>Количество оценок этой статье: $vsego</B>
";*

*echo "<B id=lolo2>Средняя оценка: $srednee</B>
";* – выводим элементы массива в браузер, т. е. общее число оценок и средний балл;

2}

echo "<form method=POST action=ocenka.php> – создаем форму, данные которой методом *POST* отсылаем на эту же страницу *ocenka.php*;

<input name=ind type=HIDDEN value=$ind>* – вводим скрытое поле для передачи на эту же страницу после нажатия кнопки «*Оценить*» значение переменной *$ind. Зачем это нужно объяснялось в разборе прошлого листинга 6.4;

<select name=ball> – создаем раскрывающийся список с именем *name=ball*, состоящий из пяти элементов, заданных тегом *<option>*. Каждому элементу списка присваивается свое значение *value* (от одного до пяти). Текст между тегами *<option>* и *</option>* – это то, что увидит посетитель в списке. При нажатии кнопки «*Оценить*» на страницу, указанную в теге *<form>*, т. е. на эту же страницу, передастся элемент суперглобального массива *$_POST['ball']*, которое будет содержать то значение *value* (балл), которое выберет посетитель;

<option value=5>5 (Отличный материал!)</option>

<option value=4>4 (В общем, хорошо!)</option>

<option value=3>3 (Неплохо, но можно было бы и получше!)</option>

<option value=2>2 (Плохо, зря я только зашел сюда!)</option>

<option value=1>1 (Лучше бы ты землю пахал, чем делал бы сайты!)</option>

</select> – окончание раскрывающегося списка;

<input name=submut type=SUBMIT value=Оценить> – кнопка для отправки данных из этой формы. В данном случае у нас двое данных: выбранная оценка, которая поместится в элемент суперглобального массива *$_POST['ball']* и значение переменной *$ind*, которое поместится в элемент суперглобального массива *$_POST['ind']*;

</form>";

1} – окончание кода между фигурными скобками под номером 1, который будет выполняться, если кнопка «*Оценить*» еще не нажата;

else – если кнопка «*Оценить*» нажата и данные из формы отправлены, то выполнится код между фигурными скобками под номером 3;

3{

$ball=$_POST['ball']; – переменной *$ball* присваиваем значение элемента суперглобального массива *$_POST['ball']*, т. е. оценку, которую дал посетитель;

$ind=$_POST['ind']; – заново определяем переменную *$ind*, присваиваем значение элемента суперглобального массива *$_POST['ind']*;

$ocenka=$ind.".txt"; – поскольку прошлую переменную *$ind* мы определи или заново, так же заново нужно определить переменные *$ocenka* и *$way*;

$way="$dir/$ocenka";

if(file_exists($way)!=true) – если текстовой файл с количеством оценок и средней оценкой еще не создан, то создим его в блоке команд в фигурных скобках под номером 4;

4{ – весь код в фигурных скобках под номером 4 выполнится только один-единственный раз, когда страницу оценят впервые;

$vsego=1;* – переменной *$vsego присваиваем единицу. Текстовый файл создается только один раз, когда странице впервые дают оценку. Естественно, общее число оценок становится равным единице;

$srednee=$ball; – переменной *$srednee*, где потом мы будем хранить средний балл, присваиваем введенный посетителем первый балл. Поскольку у нас только одна оценка, средний балл, естественно, будет равен первой оценке;

$count=$vsego."\n".$srednee; – создаем строку, вернее, две строки, разделенные символом «\n»;

$open=fopen($way, "w+"); – создаем и открываем, наконец, новый текстовый файл, путь к которому указан в переменной *$way*;

fwrite($open, $count);* – записываем в созданный файл значение переменной *$count. После этой команды в текстовом файле будет две строки. В первой строке будет единица, во второй – средний балл, равный первой оценке;

fclose($open); – закрываем текстовый файл;

4}

else – если текстовый файл с количеством оценок и средней оценкой уже создан, т. е. страницу оценивают уже не впервые, то выполнится блок команд в фигурных скобках под номером 5;

{5

$file=file($way); – как и раньше (смотри данный код чуть выше), создаем массив строк из текстового файла и присваиваем элементы этого массива переменным *$vsego* и *$srednee*;

$vsego=(int)$ file [0];

$srednee=(float)$ file [1];

$vsego++; – увеличиваем общее число оценок на единицу, поскольку посетитель сделал очередную оценку;

$srednee=($srednee($vsego-1)+$ball)/$vsego;* – здесь вычисляем среднее значение оценок, т. е. средний балл. Постарайтесь разобраться в этой строке са-

ми. Здесь чистая математика. Мы получаем десятичное число с множеством цифр после запятой;

$srednee=round($srednee, 2); – при помощи математической функции *round* округляем полученное число *$srednee* до двух знаков после запятой (второй аргумент этой функции – число 2). Думаю, этого достаточно;

$count=$vsego."\n".$srednee; – опять создаем две строки с уже новыми данными;

$open=fopen($way, "r+"); – открываем текстовый файл для записи;

flock($open, LOCK_EX); – делаем блокировку текстового файла. Зачем это нужно, объяснено в разд. 4.3;

fwrite($open, $count); – записываем новые данные в текстовый файл;

flock($open, LOCK_UN); – разблокируем текстовый файл;

fclose($open); – закрываем файл;

5} – окончание блока команд, выполняющихся, когда текстовый файл с количеством оценок и средней оценкой был уже создан;

*echo " Благодарим вас за оценку
";*

echo "<button>Вернуться</button>"; – выводим в браузер надпись и кнопку для возврата (аналогично, как и в прошлом листинге 6.4);

3} – окончание блока команд, выполняющихся, если кнопка «*Оценить*» была нажата и данные из формы отправлены.

?>

Поле того, как все правильно будет сделано, запустите программу *statji.php*. Вверху у вас должна быть двухколонная таблица с указанием названия статей и их авторов. На рис. 6.7 показана верхняя часть окна браузера, как у меня.

Название статьи	Автор статьи
Укрощение вечности	Альберт Стельвер
Стратегический план Вселенной	Альберт Стельвер
Солнечная Система - творение разума?	Алик Войцеховский
Планеты других звёзд	Л. КСАНФОМАЛИТИ
Судьба планетных систем	Г. М. Рудницкий
Параллельные Вселенные	Макс Тегмарк

Рис. 6.7

Данные шесть статей я нашел в Интернете и расположил их по папкам с 1 по 6 и в HTML-страницу каждой статьи вставил meta-теги (см. разд. 6.3, рис. 6.4). При нажатии на ссылку какой-либо из статей она открывается в отдельном окне. Для вывода в таблице ссылки на новую статью вам надо будет только создать папку (в данном случае под номером 7), вставить туда HTML-файл со статьей и не забыть в этом HTML-файле написать пару строк (meta-теги) с указанием названия статьи и его автора.

После двухколонной таблицы в браузер выведется одноколонная, с отзывами посетителей, а также текстовые поля для ввода данных посетителя и его нового отзыва (рис. 6.8).

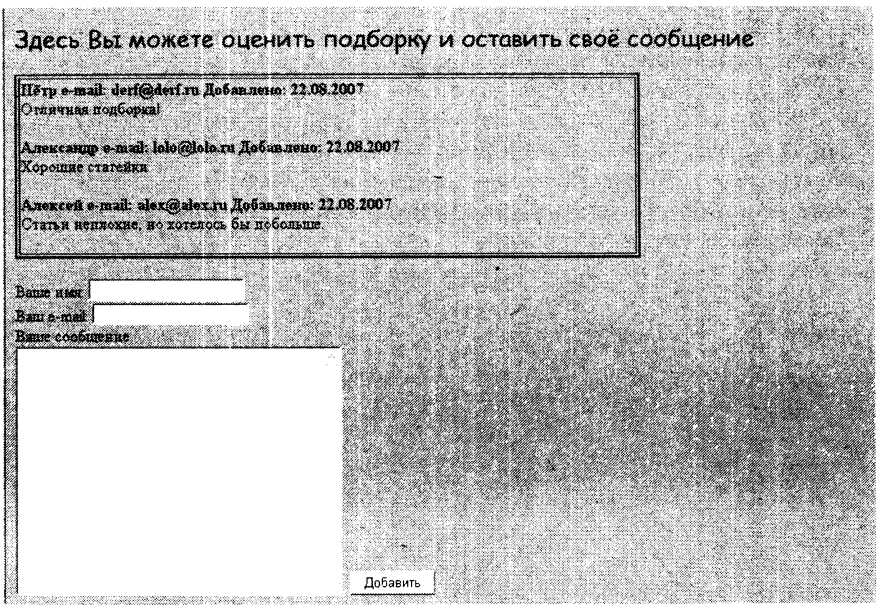


Рис. 6.8

Введите вымышленные данные и оставьте отзыв. После нажатия кнопки «*Добавить*» вы со страницы *statji.php* попадете на страницу *soobshenie.php*. Вы можете увидеть сообщение, как на рис. 6.1. Жмите «*Отмена*». Далее на странице *soobshenie.php* вы увидите кнопку «*Вернуться*». После нажатия на нее вы опять должны попасть на страницу *statji.php*. Ваши введенные данные и отзыв должны появиться в таблице отзывов. Я, например, ввел 3 отзыва (см. рис. 6.8).

Ну, а дальше, внизу в браузере, должен появиться выпадающий список с оценками и кнопка «*Оценить*», как на рис. 6.6. Правда, сначала вы не уви-

дите там надписей о количестве оценок и среднем балле. После нажатия на кнопку вы попадете на страницу *ocenka.php* с кнопкой «*Вернуться*». Нажмите на нее, и вы вновь попадете на страницу *statji.php*, но там уже отобразится и количество оценок, и средний балл. Если у вас все получилось, то прекрасно, однако есть один нюанс. Один и тот же посетитель может оценивать одну и ту же страницу несколько раз и ради прикола наставить вам кучу колов и двоек. Как этого избежать? Предлагаю решить вам эту задачу самостоятельно. Я лишь могу подсказать один из способов. Вам нужно будет в папке *ocenka* создать (программно) еще один текстовый файл, где будут храниться IP-адреса посетителей, которые выставили оценку. Как только посетитель выставил оценку, его ip заносится в этот текстовый файл (если такого ip там еще нет). Если такое ip уже есть, то выводится сообщение типа «*Вы уже оценивали*», и выполнение программы оценки *ocenka.php* приостанавливается. Конечно, этот метод крайне неэффективен, поскольку на одном ip может находиться много пользователей. Получается, что другие пользователи с таким же ip уже не смогут поставить оценку. Можно так же просто регистрировать посетителей, сохранять его логин и пароль в каком-нибудь текстовом файле. И если посетитель с парой «логин-пароль» уже ставил оценку, ему будет запрещено снова оценивать страницу.

Глава 7. ПРОСТЕЙШАЯ ГОСТЕВАЯ КНИГА

7.1. CSV-ФАЙЛЫ

Гостевая книга является чуть ли не обязательным атрибутом современных сайтов. Посетители сайта могут оставлять в гостевой книге свои сообщения или пожелания. Если помните, в прошлой главе мы делали книгу отзывов. Гостевая книга представляет собой нечто похожее, но есть и различия. В книге отзывов вы не могли отвечать на отзывы посетителей, вернее, могли, но как посетитель, а не администратор. Мы сделаем гостевую книгу, которая вам позволяла бы отвечать каждому посетителю, оставившему свою запись в этой книге. Помимо имени и e-mail, предоставим посетителю указывать URL своего сайта. Программа выведет данные посетителя, ссылки на его сайт и e-mail, дату и время появления сообщения, и, естественно, сам текст сообщения. Прежде чем приступить к созданию гостевой книги, мы, как обычно, продолжим изучение других возможностей php-языка.

До сих пор мы использовали простые текстовые файлы для хранения и считывания каких-либо данных. Для работоспособности, например, счетчика, этого вполне достаточно. Мы обходились несколькими строками. Аналогично в текстовом файле для книги отзывов мы записывали в одну строку и данные посетителя, и его отзыв. В текстовом файле для гостевой книги мы должны, помимо данных посетителя и его записи, добавлять при необходимости свой ответ этому посетителю. Записывать все это в одну строку в текстовом файле нежелательно, да и не удобно при ручном редактировании записей. В данном случае нужно применять текстовые файлы с табличной структурой, называемые CSV-файлами. Данные в таком файле записываются в виде двумерной таблицы, отделяясь друг от друга в строке некоторым разделительным символом. Для чтения данных, разделенных каким-либо символом, применяется функция:

fgetcsv($descr, line, razd, ogran). Первый аргумент является дескриптором или идентификатором открытого файла. Его, как вы помните, возвращает функция *fopen*. Второй необязательный аргумент задает длину строки в байтах. Длину можно и не указывать, но тогда снизится быстродействие функции. Вообще желательно этот параметр указывать всегда для корректной работы функции. Третий необязательный аргумент задает тип разделителя между данными в строке. По умолчанию это запятая, но можно задать свой разделитель, например «*» или «|». Четвертый необязательный параметр является символом, влияющим на объединение строк в файле. По умолчанию это двойные кавычки. Если вы в CSV-файле заключите в двойные кавычки какую-либо фразу, то данная фраза полностью станет отдельным элементом

массива, возвращаемого рассматриваемой функцией. Функция *fgetcsv* возвращает, как вы уже, наверное, догадались, массив данных строки, отделенных друг от друга указанным разделителем. При достижении конца файла данная функция возвратит *false*.

Посмотрим, как работает рассмотренная выше функция. Создайте в рабочей папке *htdocs* подпапку *gostevaja*, а в этой подпапке простой текстовый файл *CSVfile.txt*. Запишите туда две строки. Слова каждой строки разделите символом «|». Я, например, записал следующее:

один|два|три|четыре|пять
кол|пара|трояк|хорошо|отлично

После этого в PHP-редакторе создайте новый файл. Назовите его, например, *csv.php* и сохраните его тоже в подпапке *gostevaja*. Наберите между тегами *<?php* и *?>* следующие строчки:

```
$open=fopen("CSVfile.txt", "r");
$dan=fgetcsv($open, 2048, "|");
print_r($dan);
```

Сначала мы открываем файл и присваиваем ему дескриптор *$open*. Далее переменной *$dan* мы присваиваем результат действия функции *fgetcsv*. Размер строки мы установили 2 кб, хотя в данном случае хватило бы и 100 байт. Разделитель указываем тот, который использовали в текстовом файле. И, наконец, при помощи php-функции *print_r* выводим полученный массив в браузер. Сохраните и запустите программу. В браузере вы увидите следующее, если, конечно, вы записали в текстовый файл то же самое, что и я:

Array ([0]=>один [1]=>два [2]=>три [3]=>четыре [4]=>пять)

А увидите вы массив элементов первой строки (нумерация элементов в массиве, напомню, начинается с нуля). Если вы хотите, чтобы вывелись все строки, т. е. массив строк, каждая из которых представляет массив элементов (слов), удалите прошлый код и наберите немного другой:

```
$open=fopen("CSVfile.txt", "r");
while(($dan=fgetcsv($open, 2048, "|")) !==false)
{
$rec[]=$dan;
}
print_r($rec);
```

В отличие от прошлой программы, функция *fgetcsv* выполняется, пока не будет достигнут конец файла, т. е. цикл *while* будет выполняться (в нашем случае два раза), пока функция *fgetcsv* не возвратит значение *false*. При каж-

дом выполнении цикла, переменной *$dan* будет присваиваться массив слов очередной строки. Этот массив строки мы записываем в массив *$rec*. Таким образом, у нас получается двумерный массив *$rec*, элементом которого является строка, которая, в свою очередь, тоже является массивом слов, разделенных символом «|». Запустив данную программу, вы увидите следующее:

Array ([0]=>Array ([0]=>один [1]=>два [2]=>три [3]=>четыре [4]=>пять) [1]=> Array ([0]=>кол [1]=>пара [2]=>трояк [3]=>хорошо [4]=>отлично)).

Это уже двумерный массив, 2 строки по 5 элементов в каждой.

Вообще для чтения файлов с данными в виде двумерных массивов, удобно было бы использовать отдельную функцию, которую мы создадим в разд. 7.2.

7.2. СОЗДАНИЕ ФУНКЦИЙ В PHP

В PHP можно создавать функции-подпрограммы, которые можно вызывать по своим именам, при необходимости передавая им определенную информацию. Необходимы они в том случае, когда один и тот же код нужно выполнять несколько раз для разных данных, особенно если требуемое количество выполнений заранее неизвестно. Создать функцию на PHP можно, вставив в программу инструкцию, начинающуюся с ключевого слова *function*:

function имя (переменные, в которые записываются передаваемые параметры, и их тип) {...команды функции... },

а вызвать – простым указанием имени этой функции и параметров.

Помните, что переменные, созданные в функции, по умолчанию имеют установленное значение только внутри функции. Кроме того, также по умолчанию переменные, объявленные вне функции, в ней самой никакого значения не имеют.

Давайте для примера создадим функцию для считывания CSV-файлов, причем функция будет в виде отдельной программы. Создайте в PHP-редакторе новый файл, назовите его *func.php* и сохраните его в подпапке *gostevaja*, там же, где вы сохранили прошлый файл *csv.php*. Наберите код, как в листинге 7.1.

Листинг 7.1 (файл func.php)

```
<?php
//чтение массива из файла
function read($cfile, $razd="|")
{
```

```
$open=fopen($cfile, "a+");
while(($dan=fgetcsv($open, 2048, $razd)) !==false)
 {
  $rec[]=$dan;
 }
return $rec;
}
?>
```

Код похож на тот, который мы рассматривали в п. 7.1 с той разницей, что этот код мы делаем в виде отдельной функции. Разберем строчки кода:

function read($cfile, $razd="|") – после ключевого слова *function* пишем имя функции *read*. Имя может быть любым. В качестве параметров функции вводим две произвольные переменные. Первая переменная пока не задана, а вторая задана по умолчанию «|». Это значит, что при использовании данной функции, если мы не укажем второй ее аргумент, по умолчанию он примет значение «|»;

1{ – далее обязательно следует открывающаяся фигурная скобка. Весь код функции заключается в фигурные скобки;

$open=fopen($cfile, "a+"); – открываем файл, указанный в переменной *$cfile*, т. е. в переменной, которую мы указали в качестве первого аргумента нашей функции *read*. Этот аргумент будет определен после;

while(($dan=fgetcsv($open, 2048, $razd)) !==false) – считываем текстовый файл при помощи встроенной функции *fgetcsv* и создаем двумерный массив *$rec* (см. программу в п. 7.1);

2{

$rec[]=$dan;

2}

return $rec; – важная команда. Перед тем как поставить закрывающуюся фигурную скобку, означающую конец действия функции, при помощи ключевого слова *return* мы даем указание, что конкретно должна возвратить функция *read* при ее вызове. В нашем случае эта функция возвратит массив *$rec*;

1} – конец действия функции.

Теперь сохраните и запустите эту программу. Что-нибудь видно? Правильно, ничего. Мы просто написали функцию *read*, но еще не вызвали ее и не дали ей аргументов. Переменная *$cfile* еще не определена.

Откройте созданный нами предыдущий файл *csv.php*. Удалите между тегами *<?php* и *?>* весь код и вставьте новый:

```
include "func.php";
$m=read("CSVfile.txt", "|");
print_r($m);
```

Первым делом, при помощи оператора *include* вставляем на эту страницу код созданной нами функции, который у нас в файле *func.php*. Далее вызываем эту функцию, которую мы тогда назвали *read* и передаем ей два аргумента. Таким образом, в коде функции *func.php* первому аргументу, переменной *$cfile*, присваивается значение *CSVfile.txt*, а второму аргументу, переменной *$razd*, значение «|». Второй аргумент при вызове функции *read* можно было и не указывать, так как его значение «|» может быть использовано по умолчанию. Функция *read* возвращает нам двумерный массив. Его-то мы и сохраняем в переменной *$m*. Затем выводим этот массив в окно браузера. Если бы мы в последней строке написали: *print_r($rec)*, т. е. пытались бы вывести значение переменной, заданной нами непосредственно в функции на странице *func.php*, то у нас бы ничего не получилось. Как уже было отмечено выше, переменные, созданные в функции, имеют установленное значение только внутри функции, т. е. вне функции *read* переменная *$rec* не определена.

7.3. ФУНКЦИЯ БЛОКИРОВКИ ПРИ ЗАПИСИ В ФАЙЛ

В гл. 4 я объяснил, почему надо использовать блокировку текстового файла при записи туда что-либо пользователями. При одновременном использовании программы для записи в файл несколькими посетителями могут возникнуть конфликтные ситуации, которые приведут к нарушению или полному разрушению информации в файле для записи. Поэтому при записи одним посетителем что-либо в файл данный файл временно блокируется для другого посетителя. При составлении кода для счетчика для блокировки мы использовали функции *flock($descr, LOCK_EX)* и *flock($descr, LOCK_UN)*, где *$descr* – дескриптор (идентификатор) файла. Мы ставили первую функцию до записи в файл, а вторую после записи в файл (см. листинг для счетчика в гл. 4). Для счетчика такая блокировка вполне приемлема, а вот для гостевой книги – нет. Если, например, второй посетитель отправляет свои данные и свою запись в гостевую книгу на сервер для записи в текстовый файл, но этот файл был временно заблокирован первым посетителем, который отправил свои данные на мгновение раньше, то данные второго посетителя пропадут и ему придется заполнять текстовые поля заново. Разумно было бы задержать отправку данных второго посетителя, пока текстовый файл, куда пока записываются данные от первого посетителя, не разблокируется. Составим такую программу, вернее, функцию, которая бы выполняла такую задачу. Создайте в редакторе новый файл и сохраните его в папке *gostevaja* под названием *blok.php*. Наберите код листинга 7.2.

Листинг 7.2 (файл blok.php)

```
<?php
function blokir($myfile, $zap)
{
 if($fl=fopen($myfile, "a"))
 {
  for($j=0; $j<10; ++$j)
  {
   if(flock($fl, LOCK_EX)) break;
   else sleep(1);
  }
  fwrite($fl, $zap);
  flush($fl);
  flock($fl, LOCK_UN);
  fclose($fl);
  return true;
 }
 else
 {
  return false;
 }
}
?>
```

Код несложный, но мы его разберем:

function blokir($myfile, $zap) – задаем функцию с именем *blokir* и двумя произвольными переменными. Они пока еще не определены, но при вызове данной функции переменной *$myfile* мы присвоим имя файла, а переменной *$zap* – данные, которые нужно записать в файл;

1{ – начало функции;

if($fl=fopen($myfile, "a")) – проверяется условие, можем ли мы открыть файл и присвоить ему идентификатор *$fl*, т. е. доступен ли этот файл. Если текстовый файл, куда будут записываться данные, доступен, то попытаемся осуществить его блокировку. Если файл уже был заблокирован, то мы опять заблокировать тот же файл пока не сможем. Будем ждать, пока файл не освободится;

2{

for($j=0; $j<10; $++j) – делаем, так называемый цикл ожидания. Цикл будет повторяться не более десяти раз. Ждем, когда мы сможем заблокировать файл. Функция *sleep(1)*, которая на две строки ниже, каждый раз задерживает

выполнение цикла на 1 секунду (именно такая цифра указана в аргументе). Таким образом, ожидание может длиться до 10 секунд;

3{ – начало цикла ожидания;

if(flock($fl, LOCK_EX)) break; – если в течение 10 секунд нам все-таки удалось заблокировать файл, то выходим из данного цикла ожидания, при помощи инструкции *break*;

else sleep(1); – если же заблокировать файл не удалось, ждем еще 1 секунду;

3} – конец цикла ожидания;

fwrite($fl, $zap);* – теперь, когда файл заблокирован, записываем туда то, что задано нам в аргументе *$zap. Этот аргумент задается при вызове функции *blokir* в переменную *$zip*;

flush($fl); – данная функция очищает буфер записи;

flock($fl, LOCK_UN);* – снимаем блокировку с текстового файла с идентификатором *$fl;

fclose($fl); – закрываем файл;

return true; – функция *blokir* возвратит *true* при удачном выполнении. Это иногда нужно для того, чтобы узнать в программе, куда будет вставлена данная функция, удачно ли эта функция выполнилась;

2}

else – если же файл так и остался недоступным в течение 10 секунд, например, произошла какая-нибудь ошибка на сервере, то функция *blokir* не выполнится;

4{

return false; – если текстовый файл останется недоступным в течение 10 секунд, функция *blokir* возвратит *false*;

4}

1} – конец функции.

Сохраните этот код, мы его будем использовать для создания гостевой книги.

7.4. СОЗДАНИЕ ГОСТЕВОЙ КНИГИ

Гостевая книга будет состоять из трех файлов, а именно, текстового файла *CSVfile.txt*. В нем в табличном виде будут храниться данные от посетителей и их сообщения, а также ваши ответы на эти сообщения. Если файл *CSVfile.txt* у вас уже есть в папке *gostevaja*, то удалите его, так как он потом создастся сам. Потом мы создадим файл *gostev1.php*. Это будет «лицевая» страница нашей гостевой книги (рис. 7.1). На ней будут находиться последние 3 записи посетителей (можно сделать и больше) и ваши ответы на них. На данный момент на рис. 7.1 находятся пока только одна запись и ответ

на него. Чуть выше записи выводятся данные от посетителя: имя, ссылки на E-mail и сайт посетителя. Указываются также дата и время поступления записи. Слева вверху находится кнопка «*Добавить новую запись*», при нажатии на которую посетитель попадает на страницу *gostev.php*.

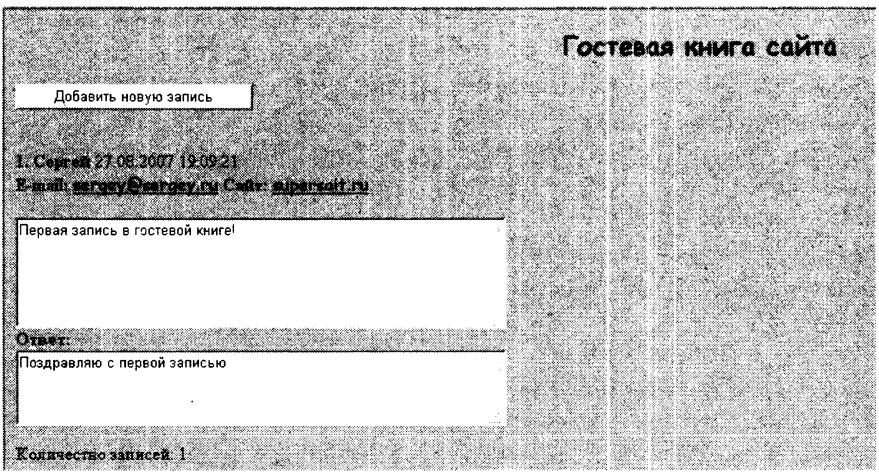


Рис. 7.1

Третий файл *gostev.php* будет представлять собой программу с формой для ввода данных от посетителей, обработки и записи этих данных в текстовый файл *CSVfile.txt* (рис. 7.2).

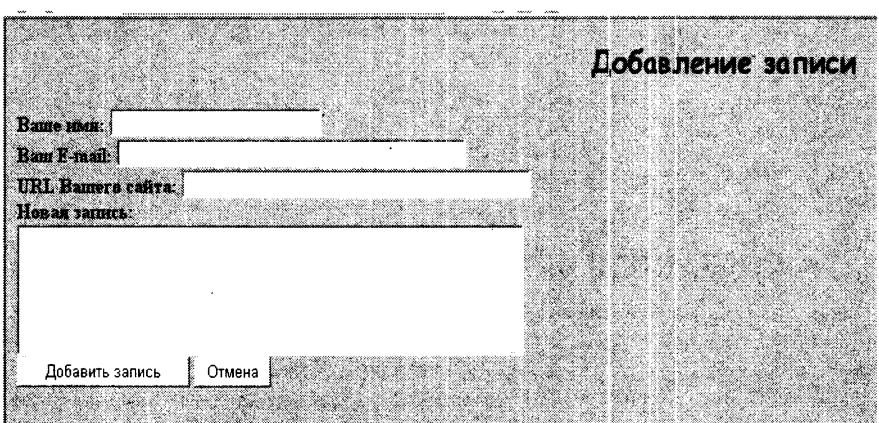


Рис. 7.2

Посетитель заполняет текстовые поля и нажимает на кнопку «*Добавить запись*». Если текстовая область для сообщения не будет пустой, данные от посетителя обработаются и запишутся в текстовый файл, а сам посетитель сможет опять возвратиться на главную страницу гостевой книги *gostev1.php* и увидеть свою запись.

Приступим к написанию кода. Сначала мы напишем код для страницы ввода новых записей *gostev.php* (см. рис. 7.2). В PHP-редакторе создайте новый файл, сохраните его в папке *gostevaja* под названием *gostev.php*. Код представлен в листинге 7.3. Желательно сначала разобраться в этом коде (разбор строк кода после листинга), а потом самостоятельно набрать его.

Листинг 7.3 (файл gostev.php)

```
<html>
<head>
<title>Новая запись</title>
<link type="text/css" rel="stylesheet" href="stil.css">
</head>
<body>
<center><b id="lolo3">Добавление записи</b></center>
<?php
$csv="CSVfile.txt";
if(!$_POST['add'])
{
 echo "<form action=gostev.php method=POST>
 <b>Ваше имя: </b><input type=text size=25 name=imja><br>
 <b>Ваш E-mail: </b><input type=text size=45 name=email><br>
 <b>URL Вашего сайта: </b><input type=text size=45 name=url><br>
 <b>Новая запись:</b><br>
 <textarea name=text rows=6 cols=70></textarea><br>
 <input type=submit name=add value='Добавить запись'>
 <input type=reset name=reset value='Отмена'>
 </form>";
}
else
{
 if(!$_POST['text'])
 {
  exit;
  }
```

```
$imja=$_POST['imja'];
$email=$_POST['email'];
$url=$_POST['url'];
$text=$_POST['text'];
$data=date("d.m.Y H:i:s");
$strok="|".strip_tags($imja)."|".$data."|".strip_tags($email)."|".strip_tags($url)."|".
strip_tags($text);
$strok=$strok.chr(13).chr(10);

$open=fopen($csv, "a");
fwrite($open,$strok);
fclose($open);

echo "<a href='gostev1.php'>Вернуться</a>";
}
?>
</body>
</html>
```

Разберем строчки этого кода:

<html>

<head>

<title>Новая запись</title> – оглавление страницы;

<link type="text/css" rel="stylesheet" href="stil.css"> – делаем указание об использовании таблицы стилей *stil.css* в данной папке. Кстати, этого файла, в котором задаются типы и цвета шрифтов, в папке *gostevaja* еще нет. Поэтому скопируйте его из главной папки *htdocs*, где мы его создавали раньше, в папку, где будут находиться все файлы гостевой книги, т. е. в папку *gostevaja*. Правда, можно и не копировать таблицу стилей, а просто указать на нее, изменив ссылку *href="stil.css"* на *href="../stil.css"*. То есть указать на каталог, который выше каталога *gostevaja*, в нашем случае каталог *htdocs*, где и находится файл стилей *stil.css*.

</head>

<body>

* Добавление записи * – вводим надпись вверху этой страницы. Тип и цвет шрифта задается идентификатором *lolo3*, который имеется у меня в таблице стилей *stil.css*. Например, у меня там есть такая запись: *#lolo3{font-size:18pt; color:#000066; font-family:Comic Sans MS}*;

<?php – начало PHP-кода;

$csv="CSVfile.txt";* – присваиваем переменной *$csv название текстового файла для записи данных от посетителей;

if(!$_POST['add']) – в элементе суперглобального массива *$_POST['add']* должны находиться переданные из формы данные от элемента этой формы с именем *add*. Форма выводится чуть ниже в следующих строках кода. Имя *name=add* имеет кнопка «*Добавить запись*». Элемент массива *$_POST['add']* будет определен только тогда, когда кнопка будет нажата. Поскольку в условии стоит восклицательный знак (символ отрицания), то оно будет истинным, если кнопка «*Добавить запись*» еще не нажата. Как только мы зайдем на данную страницу, данное условие будет пока истинно и в браузер выведется форма для ввода данных, как на рис. 7.2.

1{

echo "<form action=gostev.php method=POST> – делаем форму для ввода данных, которые после нажатия кнопки «*Добавить запись*» передадутся методом *POST* на эту же страницу *gostev.php*;

*Ваше имя: <input type=text size=25 name=imja>
*

*Ваш E-mail: <input type=text size=45 name=email>
*

*URL Вашего сайта: <input type=text size=45 name=url>
* – далее следуют 3 текстовых поля, разделенные строкой, для ввода имени посетителя, его электронного адреса и названия его сайта. По программе эти поля заполнять необязательно;

*Новая запись:
*

*<textarea name=text rows=6 cols=70></textarea>
* – далее следует текстовая область для ввода сообщения от посетителя. Эта область не должна быть пустой, иначе, как будет показано ниже в этой программе, данные в текстовый файл записываться не будут;

<input type=submit name=add value='Добавить запись'> – кнопка для отправки введенных посетителем данных и его сообщения программе-обработчику (в данном случае этой же программе *gostev.php*);

<input type=reset name=reset value='Отмена'> – кнопка для отмены введенных данных. Все текстовые поля и области очищаются;

</form>";

1} – конец блока команд, выполняющегося, когда кнопка «*Добавить запись*» еще не нажата;

else – если кнопка уже нажата, т. е. посетитель отправил свои данные и сообщение, то выполнится весь блок команд в фигурных скобках под номером 2;

{2

if(!$_POST['text']) – далее проверяется условие, переданы ли данные из текстовой области формы с именем *name=text*. Если данных нет, т. е. посетитель не ввел никакого сообщения, то выполнится единственная команда в фигурных скобках под номером 3;

{3

exit; – это команда выхода из программы. Посетитель может не указывать свои данные в первых трех текстовых полях, но если он оставил пустой текстовую область для ввода сообщения, программа дальше выполняться не будет и запись данных в текстовый файл производиться не будет;

3}

$imja=$_POST['imja'];

$email=$_POST['email'];

$url=$_POST['url'];

$text=$_POST['text']; – присваиваем четырем произвольным переменным значения элементов суперглобального массива *$_POST*. В этих элементах, как вы знаете, содержатся введенные пользователем данные – соответственно имя, E-mail, URL сайта, а в элементе *$_POST['text']* содержится сообщение посетителя;

$data=date("d.m.Y H:i:s"); – переменной $data присваиваем текущую дату и время в формате: число.месяц.год час.минуты.секунды;

$strok="|".strip_tags($imja)."|".$data."|".strip_tags($email)."|".strip_tags($url)."|".

strip_tags($text); – записываем все полученные из формы данные в строку, разделяя их символом «|». Введенные данные обрабатываем функцией *strip_tags*, которая удалит все теги и слэши, если вдруг они были введены посетителем. Зачем это нужно, объяснено в разд. 6.1. Полученную строку сохраняем в переменной *$strok*;

$strok=$strok.chr(13).chr(10); – добавляем к строке два символа *chr(13)* и *chr(10)*, означающие перевод строки, чтобы данные от следующего посетителя записывались бы с новой строки;

$open=fopen($csv, "a"); – открываем текстовый файл для записи;

fwrite($open,$strok); – записываем туда нашу строку *$strok*. После выполнения данных команд в текстовом файле *CSVfile.txt* появится строка. Сначала после пробела будет стоять символ «|». Это потом вместо пробела вы будете писать ответы на сообщения посетителей. Далее в строке будут идти имя посетителя, дата отсылки сообщения, E-mail посетителя, URL сайта посетителя и, наконец, после последнего символа «|» будет записано сообщение посетителя. Итого, получится 6 ячеек (в первой ячейке будет пока пробел);

fclose($open); – закрываем текстовый файл;

echo "Вернуться"; – после выполнения всей этой программы выводится ссылка на главную страницу гостевой книги;

2} – завершение блока команд, который выполнится только после нажатия в форме кнопки «*Добавить запись*»;

?> – завершение PHP-кода.

</body>

</html>

Итак, код разобран, но это еще не все. Вы, наверное, заметили, что я не использовал функции блокировки текстового файла при записи. Я не стал это делать сразу, чтобы вы не запутались в этом коде. Блокировку можно и не использовать, но тогда есть риск разушения информации в текстовом файле.

Еще раз посмотрите на листинг файла *blok.php* (листинг 7.2), который мы составили в разд. 7.3. Эта функция, которая открывает файл, переданный ей в качестве первого аргумента, блокирует его, делает в него запись того, что передано ей в качестве второго аргумента, и, наконец, закрывает файл. Если мы данную функцию вставим в наш разобранный выше код, то 3 строки в конце кода (в листинге 7.3 файла *gostev.php* они выделены курсивом) будут лишними.

Замените в листинге 7.3 файла *gostev.php* три строки в конце кода:

```
$open=fopen($csv, "a");
fwrite($open,$strok);
fclose($open);
```

следующими строками:

```
include "blok.php";
blokir($csv, $strok);
```

Первая строка вставляет код функции на разбираемую нами страницу *gostev.php*. Далее мы вызываем эту функцию (в коде *blok.php* она была названа *blokir*) и передаем ей в качестве аргументов название текстового файла для записи, находящегося в переменной *$csv* (именно само название файла, а не его идентификатор!) и составленную нами строку с данными *$strok*. Все, далее вызванная функция делает свою работу.

Теперь рассмотрим и разберем следующий листинг 7.4 для главной страницы нашей гостевой книги *gostev1.php* (см. рис. 7.1). Данный код считывает данные из текстового файла *CSVfile.txt* и выводит 3 сообщения в окно браузера в зависимости от значения переменной *$i*. Для считывания данных из файла *CSVfile.txt*, а они у нас там будут в виде двумерной таблицы, вос-

пользуемся ранее составленной нами функцией для считывания *func.php* (листинг 7.1).

Листинг 7.4 (файл gostev1.php)

```
<html>
<head>
<title>Гостевая книга</title>
<link type="text/css" rel="stylesheet" href="stil.css">
</head>
<body>
<center><b id="lolo3"> Гостевая книга </b></center>
<?php
if (!$_GET['i'])
{
 $i=1;
}
else
{
 $i=$_GET['i'];
}
$chislo=3;
include "func.php";
$csv="CSVfile.txt";
$m=read($csv, "|");
echo "<form action=gostev.php method=POST>
<input type=submit name=new value='Добавить новую запись'>
</form><br>";
if($m!=NULL)
{
 krsort($m);
 foreach($m as $stroka)
 {
  $n++;
  if(($n>$i*$chislo-$chislo)&&($n<=$i*$chislo))
  {
   echo "<b>$n. $stroka[1]</b> $stroka[2]<br>
   <b>E-mail: </b><a href=$stroka[3]>$stroka[3]</a>
   <b> Сайт: </b><a href=$stroka[4]>$stroka[4]</a><br><br>
   <textarea rows=6 cols=70>$stroka[5]</textarea><br>";
   if($stroka[0])
```

```
  {
   echo "<b>Ответ:</b><br>
   <textarea rows=4 cols=70>$stroka[0]</textarea><br>";
  }
  echo "<hr id=lolo3>";
 }
}
 echo "Количество записей: $n<br>";
}
echo "<center><p>";
for($k=1; $k<=$n/$chislo+1; $k++)
{
 if ($k==$i)
 {
  echo "<b>$k</b>";
 }
 else
 {
  echo "<a href='gostev1.php?i=$k'><b>$k</b></a>";
 }
}
echo "</p></center>";
?>
</body>
</html>
```

Разберем некоторые строчки кода:

<title>Гостевая книга</title> – оглавление страницы;

<link type="text/css" rel="stylesheet" href="stil.css"> – указываем браузеру на применение листа стилей stil.css;

</head>

<body>

* Гостевая книга сайта *

<?php

if (!$_GET['i']) – при загрузке данной страницы, мы должны сначала определиться со значением переменной *$i*. От нее зависит, какие 3 записи гостевой книги будут выводиться на странице. Смысл условия следующий: если переменная *$i* не получена (или мы ее не указали), то мы присваиваем ей значение единица;

```
{
$i=1;
}
```

else – иначе, если мы задали значение *i* в строке браузера, например, *gostev1.php?i=2*, то присваиваем переменной *$i* указанное значение (это значение будет храниться в элементе суперглобального массива *$_GET['i']*);

```
{
$i=$_GET['i'];
}
```

$chislo=3; – вводим переменную, которой присваиваем число 3. Именно столько записей будет на одной странице гостевой книги. По желанию можете увеличить данное число;

include "func.php"; – вставляем в данную страницу программный код файла *func.php*. Если помните, там мы создали функцию для считывания данных из csv-файлов, т. е. файлов с табличными данными (листинг 7.1);

$csv="CSVfile.txt";* – переменной *$csv присваиваем название текстового файла для записи;

$m=read($csv, "|"); – вызываем функцию для считывания содержимого текстового файла *read*. Именно так мы назвали эту функцию в *func.php*. В качестве аргументов передаем название файла и символ разделителя данных в строке текстового файла. Функция *read* возвращает содержимое текстового файла в виде двумерного массива строк и ячеек в каждой строке. Массив присваиваем переменной *$m*;

echo "<form action=gostev.php method=POST> – делаем небольшую форму с одной кнопкой «*Добавить новую запись*», при нажатии на которую посетитель перейдет на страницу для добавления записи *gostev.php*, программный код которой в прошлом листинге 7.3;

```
<input type=submit name=new value='Добавить новую запись'>
</form><br>";
```

if($m!=NULL) – теперь нам надо вывести записи из текстового файла в браузер, но вначале проверяется условие, не пуст ли массив *$m*, т. е. существует ли хотя бы одна запись в гостевой книге. Если массив *$m* не пуст, то выполнится весь код между фигурными скобками под номером 1;

```
1{
```

krsort($m); – сортировка массива с данными. Функция *krsort* сортирует массив в порядке убывания индексов, т. е. свежие записи окажутся наверху;

foreach($m as $stroka) – эта функция, как вам уже известно, присваивает переменной *$stroka* элемент массива *$m* (строку с данными) и выполняет весь

программный код между фигурными скобками под номером 2. Причем в переменной *$stroka* будет не просто строка, а массив данных в этой строке (строчный массив). Данный цикл повторяется до тех пор, пока не закончатся все элементы массива, т. е. пока функция *foreach* не переберет все строки в текстовом файле;

2{

$n++; – вводим счетчик числа сообщений, увеличивая его на единицу при каждом выполнении цикла;

if(($n>$i$chislo-$chislo)&&($n<=$i*$chislo))* – условие, указывающее на интервал номеров выводимых сообщений (записей) гостевой книги на страницу. Например, при *$i=1* условие будет истинно, если номера сообщений (*$n*) будут соответствовать интервалу от 1 до 3;

3{

*echo "$n. $stroka[1] $stroka[2]
* – если помните прошлый листинг, данные в строке отделяются друг от друга символом «|». В строке у нас 6 ячеек. Содержимое первой ячейки будет являться элементом строчного массива *$stroka* с индексом 0, т. е. *$stroka[0]*. Содержимое второй ячейки строки – первый элемент массива *$stroka[1]* и т. д. Вспомним прошлый листинг. Первая ячейка (*$stroka[0]*) содержала пробел, вторая (*$stroka[1]*) – имя посетителя, третья (*$stroka[2]*) – дату и время добавления записи. Итак, сначала в браузер (см. рис. 7.1) выводится номер сообщения *$n*, затем имя посетителя жирным текстом (используются теги ** и **), потом дата и время добавления сообщения. Далее стоит тег пропуска строки *
*;

E-mail: $stroka[3] – во второй строке жирным шрифтом выводится надпись «*E-mail:*», а затем выводится содержимое четвертой ячейки строки (элемент массива *$stroka[3]*). Это E-mail посетителя. При помощи тега *<a>* указываем ссылку на указанный посетителем электронный адрес;

* Сайт: $stroka[4]

* – далее выводим содержимое пятой ячейки строки (элемент массива *$stroka[4]*), делая ссылку на сайт посетителя. Потом пропускаем пару строк;

*<textarea rows=6 cols=70>$stroka[5]</textarea>
";* – далее создаем текстовую область и выводим туда содержимое шестой ячейки строки (элемент массива *$stroka[5]*). Там у нас текст сообщения от посетителя;

if($stroka[0]) – далее проверяется условие, есть ли что-нибудь в первой ячейке строки. В первую ячейку, как я уже отмечал, вы будете вписывать ответ на сообщение. Если вы еще ничего не вписали, то код в фигурных скобках под номером 4 выполняться не будет;

4{

*echo "Ответ:
* – если же вы вписали ответ вместо пробела в первую ячейку строки перед первым символом «|», то в браузер выведется текстовая область с вашим ответом;

*<textarea rows=4 cols=70>$stroka[0]</textarea>
";* – вывод текстовой области с вашим ответом на сообщение посетителя. Ответ, как нетрудно догадаться, находится в нулевом элементе строчного массива *$stroka[0]* (в первой ячейке строки текстового файла);

4}

echo "<hr id=lolo3>"; – выводим горизонтальную линию, отделяя сообщения друг от друга. Цвет линии задается параметром *lolo3*, который у меня присутствует в листе стилей *stil.css*;

3} – завершение вывода записей посетителя. Выведутся 3 записи;

2} – окончание выполнения цикла *foreach*. Он переберет все строки, но согласно условию *if(($n>$i*$chislo-$chislo)&&($n<=$i*$chislo))* в браузер выведутся данные только из трех строк. Какие именно строки – зависит от значения переменной *$i*;

*echo "Количество записей: $n
";* – выводится общее число записей. Поскольку функция *foreach* перебирает все строки, увеличивая переменную *$n* на единицу, после окончания цикла в переменной *$n* будет общее число этих строк, т. е. общее число сообщений;

1} – конец действия условия *if($m!=NULL)* (см. выше);

echo " <p>"; – после вывода трех сообщений гостевой книги мы должны разместить ссылки на другие сообщения. Ссылки выводим в центре, в отдельном блоке *<p>*;

for($k=1; $k<=$n/$chislo+1; $k++) – при помощи цикла *for* выводим ссылки на остальные страницы с сообщениями гостевой книги. Страниц с сообщениями у нас будет несколько, но все они как бы будут объединены в одну. Будет меняться только параметр *$i*, и в зависимости от этого параметра на страницу будет выводиться та или иная порция из трех сообщений. В переменной *$n* у нас хранится общее число всех сообщений в массиве *$m*. Число *$n/$chislo+1* означает, сколько всего должно быть страниц с записями гостевой книги. Отсюда счетчик *$k* означает у нас как бы номер страницы с очередными тремя сообщениями гостевой книги;

5{

if ($k==$i) – прежде чем выводить ссылки – проверяется условие. Нечто подобное я уже объяснял при выводе ссылок новостей сайта в листинге 5.8

(файл *new1.php*). Итак, на странице, например *gostev1.php?i=2*, будут ссылки на другие 3 записи гостевой книги (на страницы с другим параметром *$i*). Нам нужно сделать так, чтобы не срабатывала ссылка на ту страницу, на которой находится посетитель. Например, если посетитель находится на второй странице (в этом случае *$i=$k=2*), то зачем ему ссылка на вторую страницу? Выведется только числовой номер страницы (выполнится команда между фигурными скобками под номером 6);

6{

echo "$k";

6}

else

7{

echo "$k";* – выводим ссылки на страницы с другими параметрами *$i, на которых посетитель еще не находится;

7}

5} – окончание цикла вывода ссылок;

echo "</p> "; – закрываем блок ссылок;

?> – окончание PHP-кода;

</body>

</html> – конец кода.

Итак, гостевая книга готова. Запустите программу *gostev1.php*. Введите десять записей. Данные десяти посетителей можете просто выдумать. После ввода десятой записи и возврата опять на страницу *gostev1.php* вы увидите там последние 3 записи, а также ссылки на страницы гостевой книги с другим параметром *i*: *gostev1.php?i=2*, *gostev1.php?i=3*, *gostev1.php?i=4*. Перейдите последнюю страницу *gostev1.php?i=4*. Вы должны увидеть там свою первую запись. Теперь откройте текстовый файл с записями *CSVfile.txt* и в самом начале последней десятой строки, до символа «|», введите текст, например «*Ответ на десятую запись*». Сохраните и закройте текстовый файл. Перезапустите опять программу *gostev1.php*. Вы опять увидите последние 3 записи, но самая верхняя запись будет уже с ответом (рис. 7.3).

Если вы хотите редактировать текстовый файл *CSVfile.txt* не заходя на сервер, где будет находиться ваш сайт, составьте программу наподобие *redaktNovost.php* (листинг 5.6). Эта программа создавала интерфейс для редактирования текстового файла с новостями. Создать подобную программу для редактирования текстового файла с сообщениями гостевой книги для вас уже не составит особого труда.

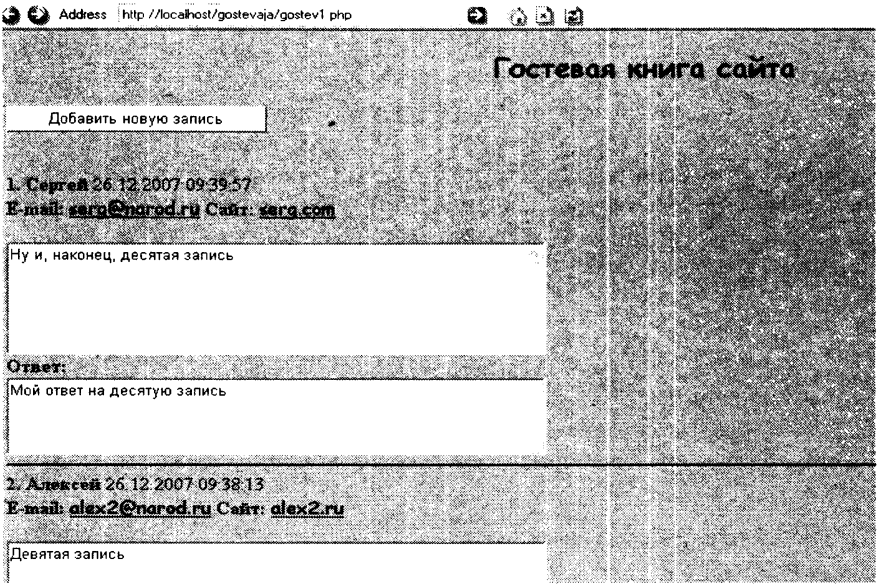

Address http //localhost/gostevaja/gostev1 php
Гостевая книга сайта
Добавить новую запись
1. Сергей 26.12.2007 09:39:57
E-mail: serg@narod.ru Сайт: serg.com
Ну и, наконец, десятая запись
Ответ:
Мой ответ на десятую запись
2. Алексей 26.12.2007 09:38:13
E-mail: alex2@narod.ru Сайт: alex2.ru
Девятая запись

Рис. 7.3

Глава 8. РАЗМЕЩЕНИЕ НА САЙТЕ ФОТОГРАФИЙ, КАРТИНОК

8.1. РАБОТА С ИЗОБРАЖЕНИЯМИ

Используя изображения на страницах своего сайта, вы создаете ему привлекательность. Сайт без картинок скучен. В этой главе мы создадим страничку для вывода каких-либо изображений на ваш сайт. Это могут быть ваши фотографии, рисунки и т. п. В нашем случае мы используем картинки, скачанные из Интернета. Мы создадим программу, которая будет выводить на страницу мини-изображения первых двенадцати картинок. Эти мини-изображения будут являться ссылками на полные изображения этих картинок, т. е. при клике мышью по какой-либо мини-картинке в новом окне появится полное изображение данной картинки. Как только число картинок превысит двенадцать, программно создастся новая PHP-страница, куда будут размещены следующие мини-картинки. По правде говоря, это будет не новая, а та же самая PHP-страница, но с другим параметром для вывода других картинок (об этом чуть позже). Делать маленькие копии размещаемых картинок (мини-картинки) вручную мы не будем, да и зачем? Все это за вас сделает php! Программа сама создаст мини-копию картинок и выведет их в окне браузера. Более того, мы предоставим посетителям вашего сайта самим загружать на ваш сайт какие-либо картинки, естественно, обезопасив себя от недоброжелателей, которые вместо картинки могут загрузить вредный код.

Как и вначале почти каждой главы, рассмотрим некоторые теоретические основы php, в данном случае для работы с изображениями. В php существует библиотека GDLib, которая и позволяет использовать функции для работы с различными изображениями. По умолчанию она отключена. Чтобы ее подключить, нужно снять комментарий со строки *extension=php_cpdf.dll* в конфигурационном файле *php.ini*. Этот файл мы настраивали в гл. 2. А находится он у меня на диске C в папке WINDOWS. У вас диск может быть другим. Итак, откройте файл php.ini и уберите точку с запятой в начале строки *extension=php_cpdf.dll*. То же самое сделайте и в строках *php_mbstring.dll, extension=php_gd2.dll* и *php_exif.dll*. Причем библиотека *php_mbstring.dll* должна быть в списке подключаемых библиотек первой. После всех сделанных изменений перезапустите свой домашний сервер на компьютере.

Рассмотрим теперь некоторые встроеннные функции в php для работы с изображениями.

Функция **getimagesize($file, $info)** возвращает массив с информацией о размере изображения в пикселах и другую информацию о файле. Первый аргумент функции *$file* – имя файла с изображением. Во второй необязатель-

ный аргумент функции заносится другая информация об изображении. Функция возвратит массив, например *$razmer* (естественно, назвать можно как угодно), состоящий из четырех элементов:

$razmer [0] – содержит ширину изображения в пикселах;

$razmer [1] – содержит высоту изображения в пикселах;

$razmer [2] – содержит число (константа), определяющее тип файла. Например, если *$razmer [2]=1*, то мы имеем дело с графическим файлом с расширением *gif*. Некоторые значения констант элемента *$razmer [2]* и соответствующие им форматы графических файлов приведены в табл. 8.1.

Таблица 8.1

Значение элемента $razmer [2]	1	2	3	6	15
Соответствующий формат	*gif*	*jpeg (jpg)*	*png*	*bmp*	*wbmp*

$razmer [3] – содержит строку с указанием ширины и высоты изображения, если они указаны в теге изображения *<img>*.

Второй необязательный аргумент *$info* представляет собой массив, в котором можно хранить различную информацию к изображению и, при необходимости, выводить ее в окно браузера, например под картинкой или фотографией. Если это не совсем понятно, то не расстраивайтесь. Чуть позже я приведу небольшой и легкий пример, показывающий работу данной (и других) функции.

Функция **imagecreatetruecolor(x, y)** создает в памяти пустое цветное изображение размерами x на y и создает дескриптор (идентификатор) созданного изображения. Все это аналогично, как и в операциях с файлами. Вспомните гл. 4. Прежде чем что-то записать в новый файл, его нужно было сначала создать и присвоить ему дескриптор (идентификатор).

Функция **imagecreatefromjpeg($file)** создает в памяти изображение из файла *$file* формата *jpeg* и возвращает дескриптор созданного изображения. Переменная *$file* содержит, как нетрудно догадаться, название файла с изображением или путь к нему, если этот файл находится в какой-нибудь другой папке.

Функция **imagecreatefromgif($file)** аналогична предыдущей, но только создает в памяти изображение из файла *$file* формата *gif*.

Функция **imagecreatefrompng($file)** аналогична двум предыдущим, но только создает в памяти изображение из файла *$file* формата *png*.

Функция **imagecopyresampled($descr_nazn, $descr_istoch, X_nazn, Y_nazn, X_istoch, Y_istoch, W_nazn, H_nazn, W_istoch, H_istoch)** копирует прямоугольные области с одного изображения на другое. Разберем аргументы этой функции:

$descr_nazn – изображение назначения, точнее, его дескриптор. Это может быть, например, дескриптор пустого изображения, возвращаемого функцией *imagecreatetruecolor(x, y)*;

$descr_istoch – дескриптор изображения источника, создаваемого в памяти, например, функцией *imagecreatefromjpeg($file)*;

X_nazn, Y_nazn – координаты точки на изображении назначения, определяющие левый верхний угол прямоугольника, в который будет вставляться копируемая область;

X_istoch, Y_istoch – координаты точки на изображении-источнике, определяющие левый верхний угол прямоугольника, содержащего копируемую точку;

W_nazn, H_nazn – ширина и высота прямоугольника, в который будет вписана копируемая область. К примеру, это могут быть размеры созданного пустого изображения;

W_istoch, H_istoch – ширина и высота копируемой области на изображении-источнике.

Позже, в примере, мы будем копировать изображение-источник с его левой верхней точки в левую верхнюю точку прямоугольной области изображения назначения, т. е. в нашем случае координаты *X_nazn, Y_nazn, X_istoch, Y_istoch* будут нулевыми.

Функция **imagejpeg($descr, name, koef)** сохраняет изображение, имеющее дескриптор *$descr*, на диск под именем *name* в формате *jpeg*. Третий необязательный аргумент-коэффициент *koef* (по умолчанию равен 75) определяет качество сжатия изображения (от 0 до 100). Чем больше коэффициент, тем меньше сжатие, но качественнее изображение.

Функция **imagegif($descr, name)** сохраняет изображение, имеющее дескриптор *$descr*, на диск под именем *name* в формате *gif*.

Функция **imagepng($descr, name)** сохраняет изображение, имеющее дескриптор *$descr*, на диск под именем *name* в формате *png*.

Функция **imagedestroy($descr)** очищает память от изображения с дескриптором *$descr*. После того как мы сохранили изображение, созданное в памяти, на диске, память нужно очистить.

Это еще далеко не полный перечень функций. Здесь мы рассмотрели только те функции, которые будут нам нужны в этой главе.

Приведу небольшой пример, чтобы показать, как работают приведенные выше функции. Но сначала создайте в нашей рабочей папке *htdocs* подпапку *risynki*, с которой мы и будем работать в этой главе. В папке *risynki* создайте еще 2 подпапки:

kart – для хранения картинок;

kartmin – для хранения уменьшенных копий этих картинок.

В подпапку *kart* скопируйте какую-нибудь картинку формата *jpg* с вашего компьютера (у меня такой файл называется *Fog.jpg*).

В PHP-редакторе создайте новый файл и сохраните его в папке *risynki* под названием *obychenie.php* (можете выбрать другое название). Наберите код листинга 8.1.

Листинг 8.1 (файл obychenie.php)

```
<?php
$f="kart/Fog.jpg";
$min="kartmin/Fog.jpg";
$razmer=getimagesize($f, $picture);
$picture[1]="Космическая туманность";
$pust=imagecreatetruecolor(130, 100);
$img=imagecreatefromjpeg($f);
imagecopyresampled($pust, $img, 0, 0, 0, 0, 130, 100, $razmer[0], $razmer[1]);
imagejpeg($pust, $min);
imagedestroy($pust);
imagedestroy($img);
$mm=basename($f, ".jpg");
print_r ("<a href='$f' target='_blank'><img src='$min' border=0
alt='$mm'></a><br>$mm<br>$razmer[3]<br>$picture[1]<br><br>");
?>
```

Данный код выводит в браузер уменьшенную копию картинки и информацию о ней. При клике мыши по этой уменьшенной картинке в отдельном окне выводится полная картинка.

Разберем листинг 8.1:

$f="kart/Fog.jpg";* – присваиваем переменной *$f путь к графическому файлу, который у нас находится в подпапке *kart*;

$min="kartmin/Fog.jpg";* – переменной *$min присваиваем путь к мини-копии графического файла. Эта мини-картинка, которой пока еще нет, будет находися в подпапке *kartmin*;

$razmer=getimagesize($f, $picture);* – получаем массив *$razmer, содержащий элементы размера картинки. В элементе массива *$razmer[0]* содержится ширина полного изображения, а в элементе *$razmer[1]* содержится высота полного изображения, а в элементе *$razmer[3]* – строка с шириной и высотой

полного изображения. Также получаем массив *$picture*, куда можно вводить другую информацию;

$picture[1]="Космическая туманность";* – первому элементу массива *$picture присваиваем описание картинки, которое мы потом выведем под изображением;

$pust=imagecreatetruecolor(130, 100);* – создаем в памяти пустое полноцветное изображение, размером 130×100 пикселей и присваиваем ему дескриптор *$pust;

$img=imagecreatefromjpeg($f); – создаем в памяти изображение из файла *$f* формата *jpeg* и присваиваем дескриптор *$img* созданному изображению;

imagecopyresampled($pust, $img, 0, 0, 0, 0, 130, 100, $razmer[0], $razmer[1]);* – копируем изображение, имеющее размеры *$razmer[0]×$razmer[1], с дескриптором *$img* в созданную пустую область *$pust*, имеющую размер 130×100;

imagejpeg($pust, $min);* – сохраняем изображение с дескриптором *$pust (теперь оно уже не пустое) под именем, указанным в переменной *$min* (в папке *kartmin*);

imagedestroy($pust);* – очищаем память от созданного изображения с дескриптором *$pust;

imagedestroy($img);* – очищаем память от созданного изображения с дескриптором *$img;

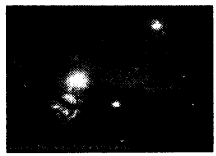
$mm=basename($f, ".jpg"); – функция *basename*, если помните гл. 4, выводит название файла без расширения, если оно указано во втором аргументе. Переменной *$mm* присвоим название нашего графического файла без расширения, т. е. в моем случае *Fog*;

*print_r ("<img src='$min' border=0>
 $mm
$razmer[3]
$picture[1]

");* – выводим графическую ссылку на полную картинку, путь к которой в переменной $f. Параметр target='_blank' указывает на то, что ссылка откроется в новом окне. При помощи тега <img> выводим созданную мини-картинку, путь к которой в переменной $min. Мини-картинка будет выведена без рамки (border=0). Далее под мини-картинкой выводим название файла из переменной $mm, строку с указанием ширины и высоты полного изображения из элемента массива $razmer[3] и, наконец, нашу надпись в элементе массива $picture[1].

Все, код разобран. Запустите теперь программу *obychenie.php*. Вы можете увидеть созданную программой мини-картинку, примерно как на рис. 8.1.

Кликнув мышью по этой мини-картинке, в отдельном окне откроется ее полноразмерная копия. Зайдите теперь в созданную вами подпапку *kartmin*. Там вы и обнаружите созданную программно мини-копию вашей картинки.



Fog
width="373" height="295"
Космическая туманность

Рис. 8.1

8.2. ФУНКЦИЯ ДЛЯ СОЗДАНИЯ МИНИ-КАРТИНОК

В прошлом примере мы уменьшили картинку и в виде графической ссылки вывели ее в браузер. Однако код в листинге 8.1 далеко не совершенен. Во-первых, при создании мини-копии фиксированного размера, изображение в этой мини-картинке может быть искаженным, поскольку полные размеры самих картинок могут быть различны. Во-вторых, программа работает только с картинками формата jpeg, а хотелось бы, чтобы она обрабатывала и другие форматы, например gif и png. В-третьих, а если картинок 10 или 100? Писать код для обработки каждой картинки просто неразумно. Давайте просто составим отдельную функцию для создания мини-копии картинки и будем вызывать ее по мере надобности. Если мини-картинка, по каким-либо причинам не будет создана, то функция возвратит значение *false*. Если вы разобрались с прошлым кодом, то код функции для создания мини-картинок (листинг 8.2) не вызовет у вас особых затруднений.

В PHP-редакторе создайте новый файл и сохраните его в папке *risynki* под названием *minkartinki.php*. Наберите код, приведенный внизу.

Листинг 8.2 (файл minkartinki.php)

```
<?php
function resizeimg($f, $min, $w, $h)
{
$koaf=$w/$h;
$razmer=getimagesize($f);
$src_koaf=$razmer[0]/$razmer[1];
if (($razmer[0]<$w) && ($razmer[1]<$h)) return true;
if ($koaf<$src_koaf) $h=$w/$src_koaf;
else $w=$h*$src_koaf;
$pust=imagecreatetruecolor($w, $h);
if ($razmer[2]==2)  $img=imagecreatefromjpeg($f);
else if ($razmer[2]==1)  $img=imagecreatefromgif($f);
else if ($razmer[2]==3)  $img=imagecreatefrompng($f);
```

```
if (!imagecopyresampled($pust, $img, 0, 0, 0, 0, $w, $h, $razmer[0], $razmer[1]))
return false;
$path=pathinfo($min);
if(($path["extension"] == "jpg") || ($path["extension"] == "JPG")) imagejpeg($pust,
$min);
else if(($path["extension"] == "gif") || ($path["extension"] == "GIF"))
imagegif($pust, $min);
else if(($path["extension"] == "png") || ($path["extension"] == "PNG"))
imagepng($pust, $min);
imagedestroy($pust);
imagedestroy($img);
return true;
}
?>
```

Разберем строчки данного кода:

function resizeimg($f, $min, $w, $h) – создаваемая нами функция resizeimg будет иметь 4 аргумента: *$f* – имя файла с исходным изображением, *$min* – имя файла с уменьшенной копией изображения, *$w* и *$h* – соответственно максимальные ширина и высота содаваемой уменьшенной копии. Хочу отметить, что это именно максимальные ширина и высота мини-копии. Реальные размеры будут вычисляться программно и будут соответствовать значениям, зависящим от размеров исходной картинки, чтобы не вызвать искажений;

{ – начало функции;

$koaf=$w/$h; – определяем коэффициент сжатия уменьшенной копии картинки, как соотношение ее ширины и высоты;

$razmer=getimagesize($f); – получаем массив *$razmer* с данными о графическом файле $f. В элементе массива *$razmer[0]* содержится ширина исходного изображения, а в элементе *$razmer[1]* – высота исходного изображения. И, наконец, в элементе *$razmer[2]* содержится информация о расширении (формате) исходного графического файла (см. табл. 8.1);

$src_koaf=$razmer[0]/$razmer[1]; – определяем коэффициент сжатия исходного изображения как соотношение ее ширины и высоты;

if (($razmer[0]<$w) && ($razmer[1]<$h)) return true; – если размеры исходного изображения меньше размеров создаваемой мини-копии, то никакой мини-копии делаться не будет и функция завершит работу;

if ($koaf<$src_koaf) $h=$w/$src_koaf; – если коэффициент сжатия мини-картинки меньше коэффицента сжатия исходной картинки, то высота создаваемой мини-картинки будет вычисляться отношением ее ширины и коэффициента сжатия исходной картинки. Это делается для того, чтобы сохранить

пропорции изображения при уменьшении. Действительно, допустим, реальная картинка имеет размеры 300×100 (коэффициент сжатия равен трем), а нам нужно создать ее мини-копию размером 100×100 (коэффициент сжатия равен единице). Ширина картинки уменьшится в 3 раза, но высота-то останется прежней. В итоге изображение исказится. Чтобы этого не произошло, мы должны уменьшить высоту мини-картинки тоже в 3 раза, что мы и сделаем в этой программной строке. В итоге мы получим мини-картинку с размерами примерно 100×33. Это меньше, чем 100×100, но зато без искажений;

else $w=$h$src_koaf;* – иначе, если коэффициент сжатия мини-картинки больше коэффициента сжатия исходного изображения, то уменьшаем ширину мини-картинки (коэффициент *$src_koaf* будет меньше единицы);

$pust=imagecreatetruecolor($w, $h); – создаем в памяти пустое полноцветное изображение для нашей мини-картинки, размером $w×$h пикселей и присваиваем ему дескриптор *$pust*;

if ($razmer[2]==2) $img=imagecreatefromjpeg($f); – проверяем, какое расширение имеет исходный графический файл, ведь от этого зависит, какие встроенные php-функции для работы с изображением мы будем использовать. Информация о расширении, как вы помните, находится в элементе массива *$razmer[2]* (см. таблицу 8.1). Итак, если графический файл имеет формат *jpeg*, то при помощи встроенной функции *imagecreatefromjpeg* создаем в памяти исходное изображение формата *jpeg* и присваиваем его дескриптор *$img*;

else if ($razmer[2]==1) $img=imagecreatefromgif($f); – иначе, если исходный графический файл имеет расширение gif, то создаем в памяти исходное изображение формата *gif*;

else if ($razmer[2]==3) $img=imagecreatefrompng($f); – иначе, если исходный графический файл имеет расширение *png*, то создаем в памяти исходное изображение формата *png*;

if (! imagecopyresampled($pust, $img, 0, 0, 0, 0, $w, $h, $razmer[0], $razmer[1])) return false; – далее функция *imagecopyresampled* копирует изображение, имеющее размеры *$razmer[0]* x *$razmer[1]*, с дескриптором *$img* в созданную пустую область *$pust*, имеющую размер $w×$h. Если же мини-картинка по каким либо причинам не будет создана, то функция возвратит значение *false*. Вообще условие *if* здесь необязательно, и данную строку можно записать так: *imagecopyresampled($pust, $img, 0, 0, 0, 0, $w, $h, $razmer[0], $razmer[1])*, т. е. без проверки условия, но если мини-копия картинки не будет создана, например, обрабатывается картинка другого формата, не указанного в данной программе, то при выполнении этой программы произойдет ошибка;

$path=pathinfo($min); – функция *pathinfo*, если помните (см. гл. 4, п. «работа со строками»), возвращает массив (присвоим ему имя *$path*), хранящий

в своих элементах: путь к директории, по которому расположен файл *($path["dirname"])*, имя файла *($path["basename"])*, расширение файла *($path["extension"])*. Нам нужно именно расширение графического файла мини-картинки, путь к которому находится в переменной *$min*;

if(($path["extension"] == "jpg") || ($path["extension"] == "JPG")) imagejpeg($pust, $min);* – если расширение файла с мини-картинкой *jpg ($path["extension"] == "jpg") или *JPG ($path["extension"] == "JPG")*, то используем функцию сохранения *jpeg* изображения *imagejpeg*. Эта функция сохраняет изображение с дескриптором *$pust* в файле, указанном в переменной *$min*. А переменную *$min* мы зададим при вызове в других программах создаваемой сейчас нами этой функции *minkartinki.php*;

else if(($path["extension"] == "gif") || ($path["extension"] == "GIF")) imagegif($pust, $min);* – иначе, если расширение файла с мини-картинкой *gif ($path["extension"] == "gif") или *GIF ($path["extension"] == "GIF")*, то используем функцию сохранения *gif* изображения *imagegif*;

else if(($path["extension"] == "png") || ($path["extension"] == "PNG")) imagepng($pust, $min);* – иначе, если расширение файла с мини-картинкой *png ($path["extension"] == "png") или *PNG ($path["extension"] == "PNG")*, то используем функцию сохранения png изображения *imagepng*;

imagedestroy($pust);* – очищаем память от созданного изображения с дескриптором *$pust;

imagedestroy($img);* – очищаем память от созданного изображения с дескриптором *$img;

return true; – в случае успешного выполнения данной функции (*minkartinki.php*), она возвратит true, хотя в программах, где мы будем использовать эту функцию, возвращаемое значение нам не понадобится. Данная функция выполнит свою задачу (создаст и сохранит мини-картинку) и закончит работу;

} – окончание функции.

Итак, функция для создания мини-копий графических изображений создана. Теперь мы ее будем использовать в дальнейшем.

8.3. ЗАКАЧКА ГРАФИЧЕСКИХ ФАЙЛОВ НА СЕРВЕР

Чтобы посетитель или вы могли загрузить файл на сервер с какой-либо web-страницы, на этой странице должна присутствовать форма с параметром заголовка *enctype*, равном '*multipart/form-data*', например:

<form enctype='multipart/form-data' action=zagryzkaObych.php method=POST>.

В форме также должно присутствовать специальное поле типа *file* (выглядит как поле ввода имени файла с кнопкой «Обзор», нажав на которую

можно отобразить окно выбора файла) и кнопка *submit* (отправить). Мы потом (в разд. 8.4) создадим форму для загрузки графических файлов, которая будет выглядеть примерно как на рис. 8.2.

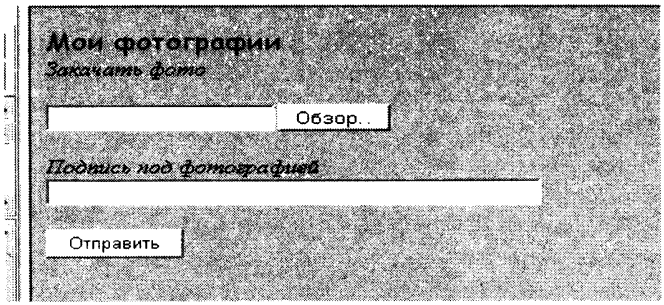


Рис. 8.2

Как только кнопка «Отправить» будет нажата, браузер начнет передавать файл, указанный в поле типа *file* на сервер. В заголовке формы *<form>* также следует указать параметр action, значением которого должно быть имя страницы с обрабатывающим загруженный файл сценарием (у нас это будет страница *zagryzkaObych.php*).

Загрузку файла на сервер умеют осуществлять практически все браузеры (только самые старые модели Microsoft Internet Explorer и Netscape Navigator этого не могут), а воспринять ее могут практически все web-серверы, в том числе и самый распространенный – Apache.

После того как файл полностью загружен на сервер, он помещается в его временную папку и находится там до тех пор, пока web-сервер не закончит обрабатывать и отдавать браузеру пользователя ту страницу, имя которой было указано в параметре *action* в теге формы загрузки файла (*<form>*). После полной выдачи страницы пользователю временная папка удаляется. Отсюда следует, что на этой странице должны обязательно присутствовать команды, перемещающие этот файл в какую-либо папку.

Странице, указанной в параметре *action* заголовка формы, передаются несколько переменных, содержащих информацию о загруженном файле. Именно на их основе сценарий на ней сможет работать с загруженным файлом. Кроме того, эти же самые переменные помещаются в массив *$_FILES*.

Вот эти переменные (*filen* – имя текстового поля для указания загружаемого файла *name='filen'*):

$_FILES ['filen '] ['tmp_name ']. В эту переменную записывается то имя (временное, создающееся автоматически), которое загруженный файл имеет в папке временных файлов. Именно с ним будут работать команда копирования файла *copy*.

Если в исходной форме присутствовало несколько полей типа *file* с разными именами, то для каждого из них создается своя переменная со значением, относящимся к соответствующему файлу:

переменная, имеющая имя *$_FILES ['filen'] ['name ']*. Ее значением является исходное имя файла в системе отправителя;

переменная, имеющая имя *$_FILES[' filen'][‘size']*. Ее значение – размер загруженного файла в байтах;

переменная, имеющая имя *$_FILES[' filen'][‘type']*. Ее значение – тип загруженного файла согласно например, «image/gif».

Все эти переменные можно использовать в PHP-сценарии на странице, указанной в параметре *action* заголовка формы. Для копирования файла используется команда *copy($_FILES['filen']['tmp_name'], filename)*, где *filename* – путь к папке, куда файл должен быть помещен. Путь к файлу во временной папке можно не указывать (она используется по умолчанию), а путь к папке, куда файл должен быть помещен (*filename*), должен указываться относительный от того каталога, в котором находится страница с обрабатывающим загруженный файл сценарием.

Об удалении файла из временной папки после копирования его в нужный каталог можно не думать – это произойдет автоматически.

8.4. СОЗДАНИЕ ФОРМЫ ДЛЯ ЗАГРУЗКИ ГРАФИЧЕСКИХ ФАЙЛОВ

Итак, сначала мы создадим страницу с формой, при помощи которой можно будет закачивать фотографии на сервер в отдельную папку. Также можно будет создать надпись под каждой фотографией. Все надписи, естественно, должны быть сохранены в отдельном текстовом файле. На данной странице будет не только форма, но и программа со сценарием для закачки файлов, т. е. в теге *<form>* в параметре *action* мы укажем ту же страницу, где и будет наша форма. Такой прием мы уже неоднократно использовали в приложениях из прошлых глав. Затем мы напишем программу для вывода мини-копий этих фотографий и надписей к ним на web-страницу.

Для начала в папке *risynki* создайте две подпапки под названием: 1. *foto* – для закачки туда фотографий; 2. *fotomin* – для хранения там мини-копий фотографий, которые будут создаваться там программно. Затем также в папке *risynki* создайте пустой текстовый файл *foto.txt*, где у нас будут храниться надписи под фотографиями.

В PHP-редакторе создайте новый файл и сохраните его в папке *risynki* под именем *zagryzkaObych.php*. Желательно сначала разобраться с листингом 8.3 (разборка кода после листинга), а потом набрать эту программу в редакторе самостоятельно.

Листинг 8.3 (файл zagryzkaObych.php)

```
<html>
<head>
 <title>Загрузка фотографий</title>
<link type="text/css" rel="stylesheet" href="stil.css">
</head>
<body bgcolor="#C0C0C0">
<div id="lplp">Мои фотографии</div>
<i>Закачать фото</i><br>
<?php
echo "<form enctype='multipart/form-data' action=zagryzkaObych.php
method=POST>
<input type=FILE name=zak><br><br>
<i>Подпись под фотографией</i><br>
<input type=TEXT name=text size=50><br><br>
<input type=SUBMIT name=otpr value='Отправить'>
</form>";
if ($_POST['otpr']==true)
{
 $zak=$_POST['zak'];
 $file=$_FILES['zak']['name'];
 $size=$_FILES['zak']['size'];
 $nadp=$_POST['text'];
$ext=array(".jpg",".gif",".png",".JPG",".GIF",".PNG");
 if(in_array(strrchr($file, "."), $ext))
 {
  if(copy($_FILES["zak"]["tmp_name"], "foto/".$file))
  {
   echo "Файл ".$file." объемом: ".$size." байт успешно закачан!<br>";
   $mass=file("foto.txt");
   if(!in_array($file."|".$nadp.chr(13).chr(10), $mass))
   {
    $foto=fopen("foto.txt", "a");
    fwrite($foto, $file."|".$nadp.chr(13).chr(10));
    fclose($foto);
    exit();
   }
  }
  else
```

```
 {
  echo "Ошибка при закачке файла! Повторите попытку.<br>";
  exit();
 }
}
else
{
 echo "Расширение закачиваемых файлов должно быть jpg, gif, png, JPG, GIF,
PNG! Повторите попытку.<br>";
 exit();
}
}
?>
</body>
</html>
```

Результат запуска этого скрипта вы уже видели на рис. 8.2. Теперь разберем строчки кода, начиная с тега *<?php*:

<?php – начало PHP-кода;

echo "<form enctype='multipart/form-data' action=zagryzkaObych.php method=POST> – создаем форму для отправки файла методом *post*. Данные из этой формы после нажатия кнопки «Отправить» передадутся вновь на эту же страницу (*action=zagryzkaObych.php*);

*<input type=FILE name=zak>

* – создаем поле для ввода имени загружаемого на сервер файла (точнее, пути к нему);

*Подпись под фотографией
*

*<input type=TEXT name=text size=50>

* – создаем текстовое поле для ввода надписи под фотографией;

<input type=SUBMIT name=otpr value='Отправить'> – создаем кнопку для отправки данных из формы;

</form>"; – закрываем тег формы;

if ($_POST['otpr']==true) – если элемент суперглобального массива *$_POST['otpr']* определен, т. е. мы ввели данные в форму и нажали кнопку «Отправить» (а кнопка имеет имя *name=otpr*), то выполнится весь код между фигурными скобками под номером 1;

1{

$zak=$_POST['zak']; – в переменной *$zak* сохраняем путь к закачиваемому файлу, указанному в форме в поле с именем *name=zak*;

$file=$_FILES['zak']['name']; – переменной *$file* присваиваем имя закачиваемого файла. Оно хранится в элементе *$_FILES['zak']['name']* массива *$_FILES*, созданного при передаче данных из формы (см. разд. 8.1–8.3);

$size=$_FILES['zak']['size']; – переменной *$size* присваиваем размер закачиваемого файла в байтах;

$nadp=$_POST['text']; – переменной *$nadp* присваиваем содержимое текстового поля с именем *name=text* из формы. А в это поле мы записали надпись под фотографией;

$ext=array(".jpg",".gif",".png",".JPG",".GIF",".PNG");* – создаем массив *$ext с шестью элементами, указывающими на расширение графических файлов. Мы предполагаем, что будут закачиваться только 3 типа графических файлов с расширениями *jpg, gif* и *png* (или с такими же расширениями, но из заглавных букв). Вы можете добавить сюда и другие расширения для графических файлов;

if(in_array(strrchr($file, "."), $ext)) – сначала вспомним, что функция *strrchr* (см. гл. 4) возвращает часть строки, начиная с символа, указанного во втором аргументе (в нашем случае это точка), и до конца строки, т. е. данная функция возвратит нам расширение файла. Затем функция *in_array* ищет в массиве *$ext* элемент, указанный в первом аргументе, а первый аргумент у нас *strrchr($file, ".")*, т. е. расширение файла. Короче, условие *if* проверяет, имеет ли закачиваемый файл расширение, соответствующее хотя бы одному из элементов массива *$ext*. Если закачиваемый файл графический и имеет расширение *jpg, gif* или *png* (или с такими же расширениями, но из заглавных букв), то выполнится блок команд между фигурными скобками под номером 2;

2{

if(copy($_FILES["zak"]["tmp_name"], "foto/".$file)) – функция *copy* копирует файл из переменной *$_FILES["zak"]["tmp_name"]* в папку *foto* (мы ее создали заранее) и сохраняет ее там под именем, которое у нас в переменной *$file* (т. е. под тем же именем). В переменную *($_FILES["zak"]["tmp_name"]*, как было сказано выше, записывается то имя, которое загруженный файл имеет в папке временных файлов. Если файл удачно скопирован и сохранен, то выполнится блок команд между фигурными скобками под номером 3;

3{

*echo "Файл ".$file." объемом: ".$size." байт успешно закачан!
";* – при удачном выполнении прошлой команды в браузер выведется надпись, что файл под названием *$file* и объемом *$size* (а в переменной *$size* у нас размер закачиваемого файла в байтах) успешно закачан;

$mass=file("foto.txt"); – в текстовом файле *foto.txt*, который мы создали заранее, будут храниться надписи под фотографиями. Причем, как вы увидите

позже, записываться и храниться они будут в виде таблицы с двумя столбцами. Первый столбец будет содержать название графического файла с расширением, второй – надпись под этим графическим файлом. Столбцы будут разделены символом «|». Функция *file* считывает текстовый файл *foto.txt* и возвращает массив строк (см. гл. 4) из этого файла. Весь массив сохраняем в переменной *$mass*;

if(!in_array($file."|".$nadp.chr(13).chr(10), $mass)) – ищем в массиве *$mass* строку с названием загружаемого файла и надписью под ним, т. е. проверяем, загружали ли мы уже этот файл. Это мы делаем для того, чтобы: во-первых, вы не загружали один и тот же файл несколько раз; во-вторых, если при отправке данных из формы сервер потребует повторную отправку (см. рис. 6.1), то в текстовый файл уже не будут записываться вторично название и надпись для одного и того же файла. Итак, если такой строки с именем и надписью для загружаемого файла еще не существует, то выполнится код между фигурными скобками под номером 4;

4{

$foto=fopen("foto.txt", "a"); – открываем текстовый файл *foto.txt* для записи и присваиваем открытому файлу идентификатор *$foto*;

fwrite($foto, $file."|".$nadp.chr(13).chr(10)); – делаем запись очередной строки. Сначала запишем название файла *$file*, затем при помощи точки (методом конкатенации) присоединяем символ-разделитель «|», после идет надпись под фотографией, хранящейся в переменной *$nadp*. В конец строки ставим известные уже вам два знака *$chr(13)* и *chr(10)*, обозначающие переход на новую строку;

fclose($foto); – закрываем текстовый файл;

exit(); – после удачной записи в файл программа должна завершиться. Если мы не поставим здесь оператор *exit()*, программа выполнит код ниже и выведет в браузер надписи, которые должны выводиться только в случае неудачной загрузки файла;

4} – завершение действия условия, проверяющего, был ли загружаемый файл уже загружен ранее;

3} – завершение действия условия, выполняющегося при удачной загрузке файла, т. е. при удачном копировании и сохранении загружаемого файла;

else – если же загружаемый файл по каким-либо причинам не был загружен, то выполнится код в фигурных скобках под номером 5 (выведется сообщение об ошибке, и программа завершит работу);

5{

*echo "Ошибка при закачке файла! Повторите попытку.
";*

exit();

5}

2} – завершение действия условия, выполняющегося, если закачиваемый файл графический и имеет расширение *jpg, gif* или *png* (или с такими же расширениями, но из заглавных букв);

else – если файл не графический или имеет другое расширение, то выполнится код в фигурных скобках под номером 6 (выведется сообщение, что расширение файла должно быть *jpg, gif* или *png*, и программа завершит работу);

6{

*echo "Расширение закачиваемых файлов должно быть jpg, gif, png, JPG, GIF, PNG! Повторите попытку.
";*

exit();

6}

1} – окончание действия условия, выполняющегося, если элемент суперглобального массива *$_POST['otpr']* определен, т. е. мы ввели данные в форму и нажали кнопку «*Отправить*»;

?> – окончание PHP-кода.

Пусть на диске вашего компьютера есть фото с названием *img001.jpg*. Запустите программу и попробуйте закачать этот графический файл. Нажмите кнопку «*Обзор*» (см. рис. 8.2) и в появившемся диалоговом окне выберите данный файл, нажав потом на кнопку «*Открыть*». Далее в текстовом поле для надписи (подпись под фотографией) введите фразу «первое фото» и нажмите на кнопку «*Отправить*». Если файл будет удачно закачан, вы увидите об этом надпись с указанием размера закачанного файла (рис. 8.3).

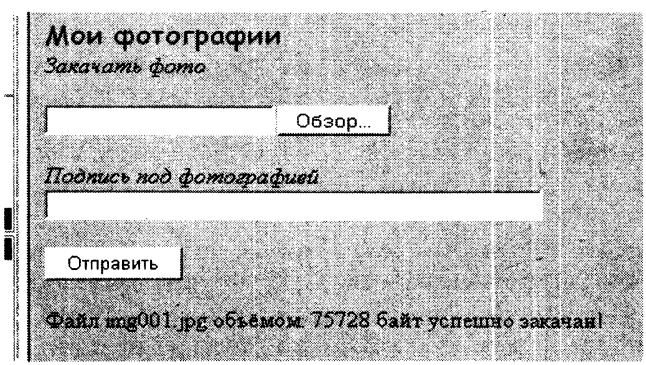


Рис. 8.3

Зайдите теперь в папку *foto* (она должна была быть создана в рабочей папке *htdocs* еще до написания этой программы). Там вы должны увидеть закачанный графический файл. Откройте теперь текстовый файл *foto.txt*. Там

вы увидите пока единственную строку с названием закачанного файла и надписью. Закройте текстовый файл, ничего в нем не меняя!

А теперь одно важное замечание! Естественно, вы будете закачивать в папку *foto* не одну фотографию. Название загружаемых файлов не должно состоять из русских букв, иначе они не распознаются php-интерпретатором. Например, файл *мое первое фото.jpg* не загрузится в папку *foto*. Все загружаемые вами графические файлы должны иметь имена либо из латинских букв, либо из цифр (можно вперемежку с латинскими буквами). Поэтому, прежде чем загружать файлы, переименуйте их. Назовите их так: *img001, img002* и т. д. Обязательно сохраняйте цифровой порядок при закачке файлов, чтобы потом они выводились на вашей странице (ее мы составим в следующем листинге) в порядке возрастания цифр. Зачем? А чтобы при выводе ваших фото на страницу, не перепутались бы выведенные в браузер фотографии и надписи к ним! Итак, переименуйте свои фото так, как указано выше, и закачайте их в папку *foto* при помощи созданной выше программы *zagryzkaObych.php*.

8.5. ПРОГРАММЫ ДЛЯ ВЫВОДА МИНИ-КОПИЙ ГРАФИЧЕСКИХ ФАЙЛОВ НА СТРАНИЦУ

Теперь создадим страницу для вывода мини-копий закачанных фотографий с надписями. При клике мышью по этой мини-копии в отдельном окне откроется полная фотография.

Программа для создания мини-копии картинки *minkartinki.php* уже нами составлена (листинг 8.2). Нам еще понадобится программа для чтения файлов с табличными данными. Эту программу мы тоже уже составили (листинг 7.1 файла *func.php*). Файл *func.php*, если вы помните, находится у нас в папке для гостевой книги *gostevaja*. Скопируйте этот файл в папку *risynki*, с которой мы и работаем в данной главе.

В PHP-редакторе создайте новый файл и сохраните его в папке *risynki* под названием *kartinkiObych.php*. Наберите код (листинг 8.4).

Листинг 8.4 (файл kartinkiObych.php)

```
<html>
<head>
 <title>Фото</title>
 <link rel="stylesheet" type="text/css" href="stil.css">
</head>
<body bgcolor="silver">
<center><I id=lolo>Мои фотографии</I></center><br>
```

```
<?php
echo "<table align=CENTER border=2><tr>";
$z=0;
$kart=opendir("./foto");
$foto="foto.txt";
include "minkartinki.php";
include "func.php";
$mass=read($foto);
while (($file=readdir($kart)) !==false)
{
 if($file !="." &&  $file !=".." && $file !="Thumbs.db")
 {
  if($mass[$z][0]==$file)   $zap=$mass[$z][1];
  if (!file_exists("./fotomin/$file"))
  {
   resizeimg("foto/$file", "fotomin/$file", 130, 100);
  }
  $size_r=getimagesize("foto/$file");
  print_r ("<td align='center'><a href='foto/$file' target='_blank'><img
src='fotomin/$file' border=0 alt='$file'></a><br><I id=lolo5>$file<br>$size_r[0] x
$size_r[1]<br>$zap</I></td>");
  $zap="";
  $z++;
  if (($z/4) == is_integer($z/4)) echo "</tr> <tr> ";
 }
}
closedir($kart);
?>
</body>
</html>
```

Код для вас уже должен быть несложным, но мы его все равно разберем, начиная с тега <?php:

<?php – начало PHP-кода;

echo "<table align=CENTER border=2><tr>"; – создаем таблицу, в ячейках которой и будут выводиться ваши фотографии. Тег *<tr>*, если помните, создает новую строку в таблице;

$z=0;* – вводим переменную *$z, которая потом будет счетчиком числа фотографий в папке *foto*;

$kart=opendir("./foto"); – функция *opendir* открывает директорию, указанную в скобках в качестве аргумента. У нас это директория *foto*, где хранятся загруженные нами фотографии. Открытой директории присваивается дескриптор *$kart*, который и используется в дальнейшем;

$foto="foto.txt";* – переменной *$foto присваиваем имя текстового файла, где будут храниться надписи к фотографиям;

include "minkartinki.php"; – вставляем в этот код другой код из файла *minkartinki.php*. Там у нас функция *resizeimg* для создания мини-копий;

include "func.php"; – вставляем код из страницы *func.php*. Там у нас функция *read* для чтения табличных данных из файла;

$mass=read($foto); – при помощи созданной нами функции *read* считываем данные из текстового файла *foto.txt*. Функция возвратит двумерный массив *$mass* из строк и ячеек, из которых эти строки состоят;

while (($file=readdir($kart)) !==false) – цикл с условием, который перебирает все файлы (фотографии) из директории *foto* (дескриптор *$kart*). Этот цикл выполнится столько раз, сколько файлов в директории, присваивая переменной *$file* очередное название файла с фотографией. Как только условие цикла *($file=readdir($kart)) !==false* станет истинным, т. е. все картинки будут перебраны (переменной *$file* ничего уже не будет присвоено, и она примет значение *false*), цикл прекратится;

1{ – начало цикла;

if($file !="." && $file !=".." && $file !="Thumbs.db") – среди выданных функцией *readdir* имен файлов будут и ссылки на текущий и родительский (т. е. включающий в себя текущий) каталог, обозначаемые соответственно одной и двумя точками (так уж работает web-сервер). Поскольку нас будут интересовать только файлы каталога, то данные ссылки из списка файлов следует исключить, добавив проверку состава имени файла. Также следует исключить файл *Thumbs.db*, который может создаться на сервере автоматически. Итак, если выбранный функцией *readdir* файл (*$file*) не является текущим или родительским каталогом и это не файл *Thumbs.db*, то выполнится код между фигурными скобками под номером 2;

2{

if($mass[$z][0]==$file) $zap=$mass[$z][1]; – проверяется условие. Если помните, данные в текстовом файле *foto.txt* представлены в виде таблицы с двумя столбцами. В первом столбце – имя графического файла с фотографией, во втором – надпись под этой фотографией. Второй индекс в двумерном массиве *$mass* указывает на номер столбца (у нас их два – с индексами соответственно 0 и 1). Нулевой индекс указывает на первый столбец с именами файлов. Первый индекс массива *$mass[$z]* указывает на строку, нумерация которой тоже начинается с нуля. При каждом выполнении цикла про-

веряется каждая строка (переменная *$z* каждый раз увеличивается на единицу). Смысл проверки условия следующий: если название очередного файла *$file*, совпадает с названием файла из первого столбца проверяемой строки таблицы, то переменной *$zap* присваивается содержимое второго столбца этой же проверяемой строки, т. е. надпись под фотографией для фото с именем *$file*;

if (!file_exists("./fotomin/$file")) – снова проверяется условие, существует ли уже мини-копия фотографии с именем *$file*. Мини-копии у нас будут храниться в заранее созданной папке *fotomin*. Если мини-копии еще нет, то выполнится команда между фигурными скобками под номером 3;

3{

resizeimg("foto/$file", "fotomin/$file", 130, 100); – при помощи созданной нами функции *resizeimg* (эта функция в файле *minkartinki.php*, а файл мы уже вставили в этот код при помощи оператора *include*) создаем мини-копию из фотографии с именем *$file*, размером 130×100, и сохраняем ее в папке *fotomin* под таким же именем;

3}

$size_r=getimagesize("foto/$file"); – функция *getimagesize* создает массив *$size_r* с данными о файле (фотографии) *$file* в папке *foto* (см. разд. 8.1–8.3). В элементе массива *$size_r[0]* будет содержаться ширина полной фотографии в пикселах, а в элементе *$size_r[1]* – высота полной фотографии;

*print_r ("<td align='center'><img src='fotomin/$file' border=0 alt='$file'>
<I id=lolo5>$file
 $size_r[0] x $size_r[1]
$zap</I></td>");* – в этой длинной строке при помощи оператора *print_r* выводим мини-копию фотографии в браузер. Копия будет выводится в ячейку (тег *<td>*) таблицы, созданной в самом начале этого кода. Мини-копия фотографии в папке *fotomin* является еще и графической ссылкой на полную фотографию в папке *foto* (тег *<a>*). Далее выводятся курсивом название файла *$file*, его ширина и длина и надпись под фотографией (она у нас в переменной *$zap*). Идентификатор *lolo5* указывает на тип и размер шрифта для вывода названия файла, его размера и надписи под фото. Этот идентификатор задан у меня в таблице стилей *stil.css*, а сама таблица стилей была задана в начале этого кода в строке *<link rel="stylesheet" type="text/css" href="stil.css">*. Кстати, файл стилей *stil.css* должен обязательно быть в папке *risynki*. Как создавать файл стилей, рассказано в гл. 4;

$zap="";* – очищаем содержимое переменной *$zap. При следующем выполнении цикла *while* этой переменной будет присвоена следующая надпись для следующей фотографии;

$z++;* – при каждом выполнении цикла будем увеличивать счетчик файлов *$z на единицу;

if (($z/4) == is_integer($z/4)) echo "</tr> <tr> "; – как только число загруженных фотографий станет кратно четырем, завершаем строку таблицы с фотографиями и начинаем новую. Например, пятая фотография будет выведена уже во втором ряду. Если вы хотите, чтобы в одном ряду выводилось, например, 5 фотографий, измените в этой строке число 3 на 5;

2} – окончание блока команд для условия, выполняющегося, когда выбранный функцией *readdir* файл не является текущим или родительским каталогом и это не файл *Thumbs.db*;

1} – окончание блока команд для цикла *while*, который перебирает все файлы (фотографии) из директории *foto*;

closedir($kart); – закрываем директорию *foto* с дескриптором *$kart*;

?> – конец PHP-кода.

Итак, программа разобрана. А теперь при помощи формы для загрузки файлов (программа *zagryzkaObych.php* листинга 8.3, рис. 8.3) загрузите в папку *foto* семь любых фотографий, предварительно переименовав их на *img001.jpg, img002.jpg* и т. д. У меня, например, это 7 космических фото. Надписи к ним (в текстовом поле формы в программе *zagryzkaObych.php*) я написал просто: первое фото, второе фото и т. д. Теперь запустите последнюю разобранную программу *kartinkiObych.php*. Результат работы данной программы на рис. 8.4.

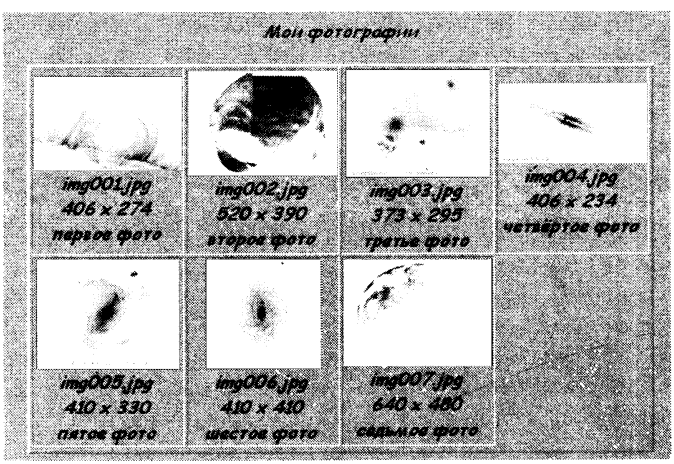


Рис. 8.4

Как видите, мини-копии фотографий выводятся по 4 штуки в ряд. Кликните мышью по какой-нибудь мини-копии – в отдельном окне откроется полная фотография.

Теперь допустим, что вам надо удалить какую-либо фотографию. Зайдите в папку *foto* и удалите оттуда, например, фото с названием *img005.jpg*. Перезапустите заново программу *kartinkiObych.php*. Вы увидите, что пятая картинка *img005.jpg* будет удалена, но начиная с картинки *img006.jpg* изчезнут надписи внизу (изчезнут надписи «шестое фото» и «седьмое фото»). А произошло это потому, что мы не удалили в текстовом файле *foto.txt* строку, принадлежащую картинке с названием *img005.jpg*. Если вы внимательно разобрали программу *kartinkiObych.php* листинга 8.4, то поймете, что надпись под какую-либо картинку будет выводиться в браузер, если она действительно принадлежит именно этой картинке. Мы удалили картинку, но программа все равно будет искать название файла с этой картинкой в папке *foto* и в текстовом файле *foto.txt* строго по порядку. Картинок у нас стало на одну меньше, но число записей в файле *foto.txt* не изменилось. Начиная с картинки *img006.jpg* возникнет несоответствие картинок и надписей к ним, поэтому они (надписи) и перестанут выводиться в браузер. Под картинкой будет только выведено название файла и размеры полной картинки. Откройте теперь текстовый файл *foto.txt* и полностью удалите там строку, начинающуюся с названия удаленного файла *img005.jpg*. У нас это пятая по счету строка. Удалите так, чтобы между строками 4 (*img004.jpg*) и 6 (*img006*.jpg) не было бы даже пустой строки. Сохраните изменения в файле и вновь перезапустите программу *kartinkiObych.php*. Теперь все в порядке, каждая надпись соответствует своей картинке. Если же вы потом захотите вновь разместить удаленное фото на странице *kartinkiObych.php*, загрузив его опять при помощи формы (программы *zagryzkaObych.php*), то вам придется сначала изменить номерное название фото с *img005.jpg* на название с более бóльшим номером (следующим после номера последней закачанной картинки). В нашем случае (см. рис. 8.4) название удаленного файла *img005.jpg*, прежде чем его вновь загрузить, нужно будет заменить, например на *img008.jpg*. Делается это, чтобы не нарушать цифровой порядок закачанных фотографий (чтобы не спутались файлы с фотографиями и надписи к ним).

Вы, конечно, можете задать вопрос, а почему мы вообще делаем надписи, дополнительно создаем для их хранения текстовый файл? Почему бы просто не назвать свои файлы по-русски, например, не *img001.jpg*, а *мое первое фото.jpg*? Увы, так делать нельзя. Как я уже отметил выше, на многих серверах (почти на всех) интерпретатор php не распознает файлы с русскими названиями. Поэтому надписи на русском языке мы сохраняем отдельно в текстовом файле.

Итак, мы составили нашу web-страничку для вывода фотографий. Теперь давайте создадим еще одну страничку. Допустим, вы хотите вывести много картинок – 1000 и более. Надписи к ним выводить, допустим, не нужно. Названия файлов переименовывать не будем, если, конечно, они не на русском языке. Закачивать картинки на сервер при помощи нашей формы (программа

zagryzkaObych.php) мы не будем (если картинок очень много, то это будет утомительно), а просто закачаем их в отдельную папку вручную. На первой странице будет выводиться только 12 картинок (точнее, их мини-копий), остальные (тоже по 12) будут выводиться на последующих страницах. Мини-копии картинок тоже будут создаваться программно. На всех страницах, естественно, будут ссылки и к другим страницам. На всех страницах будет находиться форма для загрузки графических файлов, чтобы любой посетитель вашей web-странички с картинками мог добавить свою картинку. Программу для загрузки файлов посетителями мы создадим чуть позже. Она немного будет отличаться от ранне созданной программы *zagryzkaObych.php*. Там уже не будет никакой формы (она будет уже на страницах с картинками), и мы введем ограничение, чтобы объем закачиваемой посетителем картинки не превышал бы 3Мб.

Итак, если на каждой странице у нас будет по 12 картинок, а всего картинок будет, предположим, 1000, то число страниц с картинками будет 84. Составлять каждую страницу мы не будем, да и зачем? Мы создадим одну-единственную страницу для вывода картинок. Она будет называться *kartinki.php*. Введем некую переменную *$j*, и в зависимости от ее значения, на страницу будет выводиться тот или иной блок из 12 картинок. Например, при *$j=1* выведутся картинки с 1 по 12, при *$j=2* выведутся картинки с 13 по 24 и т. д. Значение переменной *$j* можно указать после названия страницы с картинками. Например, если мы хотим, чтобы переменная *$j* была равна 2, нужно записать *kartinki.php?j=2* (без значка $). Такой способ передачи на страницу переменных был рассмотрен в разд. 5.1.

Итак, в папке *risynki* вы должны были создать две подпапки: подпапку *kart*, куда мы будем загружать полные картинки, и подпапку *kartmin*, где будут храниться созданные программно мини-копии картинок. В папку *kart* я, к примеру, скопировал 27 космических картинок. В PHP-редакторе создайте новый файл. Сохраните его в папке *risynki* под названием *kartinki.php*. Наберите код:

Листинг 8.5 (файл kartinki.php)

```
<html>
<head>
 <title>Космические рисунки</title>
 <meta http-equiv="content-type" content="text/html; charset=windows-1251">
 <link rel="stylesheet" type="text/css" href="stil.css">
</head>
<body bgcolor="silver">
<center><I id=lolo3>Космические рисунки</I></center><br>
<?php
$z=0;
```

```
if (!$_GET['j'])
{
 $j=1;
}
else
{
 $j=$_GET['j'];
}
echo "<table align=CENTER border=2 bgcolor=black><tr>";
include "minkartinki.php";
$kart=opendir("./kart");
while (($file=readdir($kart)) !==false)
{
 if($file !="." && $file !=".." && $file !="Thumbs.db")
 {
  $z++;
  if (!file_exists("./kartmin/$file"))
  {
   resizeimg("kart/$file", "kartmin/$file", 100, 70);
  }
  $size_r=getimagesize("kart/$file");
  if (($z>12*$j-12) && ($z<=12*$j))
  {
   print_r ("<td align=center id=lolo2><a href=\"kart/$file\" target=_blank><img
src=\"kartmin/$file\" border=0 alt=\"$file\"></a><br><I
id=lolo6>$file<br>$size_r[3]</I></td>");
   if (($z/4) == is_integer($z/4)) echo "</tr>";
  }
 }
}
echo"</td></tr></table>";
echo "<center><p>";
for($i=1; $i<=$z/12+1; $i++)
{
 $k=(12*$i-11)."...".($i*12);
 if ($i==$j)
 {
  echo "<b id=lolo4>$k</b>";
 }
```

```
else
{
 echo "<a href='kartinki.php?j=$i'><b id=lolo4>$k</b></a>";
}
}
echo "</p></center>";
?>
<br><br>
<div>Можете закачать на сайт свою картинку!</div>
<form enctype='multipart/form-data' action=zagryzka.php method=POST>
<input type=FILE name=zak><br><br>
<input type=SUBMIT name=otpr value='Отправить'>
</form>
</body>
</html>
```

Пока не запускайте эту программу, поскольку она еще работать не будет. Давайте разберем строчки кода, начиная с тега <?php:

<?php

$z=0;* – вводим счетчик картинок. В переменной *$z будет храниться порядковый номер картинки;

if (!$_GET['j']) – при загрузке данной страницы мы должны сначала определиться со значением переменной *$j*. Ведь от нее зависит, какие картинки будут выводиться. Смысл условия следующий: если переменная *$j* не получена (или мы ее не указали), то мы присваиваем ей значение единица, т. е. делаем по умолчанию *$j=1*;

{

$j=1;

}

else – иначе, если мы задали значение j в строке браузера, например, *kartinki.php?j=2*, то присваиваем переменной *$j* указанное значение (это значение будет храниться в элементе суперглобального массива *$_GET['j']*);

{

$j=$_GET['j'];

}

echo "<table align=CENTER border=2 bgcolor=black><tr>"; – выводим начало таблицы, в ячейках которой и будут вставлены мини-копии картинок. Толщина линий таблицы равна 2 пиксела (*border=2*), фоновый цвет таблицы задан черный (*bgcolor=black*), хотя можете задать и свой. Открываем первую

строку таблицы, при помощи тега <tr>. Все, пока строительство таблицы прерываем;

include "minkartinki.php"; – вставляем код из файла *minkartinki.php*. В том коде есть написанная нами функция *resizeimg* для создания мини-копий картинок;

$kart=opendir("./kart"); – открываем директрию с полными картинками;

while (($file=readdir($kart)) !==false) – цикл с условием, который будет выполняться до тех пор, пока функция *readdir* не переберет все файлы в директории *kart*. Этот цикл выполнится столько раз, сколько файлов в директории, присваивая переменной *$file* очередное название файла с картинкой;

1{ – начало цикла;

if($file !="." && $file !=".." && $file !="Thumbs.db") – если выбранный функцией *readdir* файл (*$file*) не является текущим или родительским каталогом и это не файл *Thumbs.db*, то выполнится код между фигурными скобками под номером 2;

2{

$z++;* – при каждом выполнении цикла будем увеличивать счетчик файлов *$z на единицу. Это и будет как бы порядковый номер картинки, выбранный функцией *readdir*;

if (!file_exists("./kartmin/$file")) – если мини-копии выбранной функцией *readdir* картинки еще не существует в папке *kartmin*, то выполнится команда между фигурными скобками под номером 3 для создания данной мини-копии;

3{

resizeimg("kart/$file", "kartmin/$file", 100, 70); – при помощи созданной нами функции *resizeimg* (эта функция в файле *minkartinki.php*, а файл мы уже вставили в этот код при помощи оператора *include*) создаем мини-копию из картинки с именем *$file*, размером 100×70, и сохраняем ее в папке *kartmin* под таким же именем;

3}

$size_r=getimagesize("kart/$file"); – функция *getimagesize* создает массив *$size_r* с данными о файле (картинке) *$file* в папке *kart* (см. разд. 8.1–8.3). В элементе массива *$size_r[0]* будет содержаться ширина полной картинки в пикселах, а в элементе *$size_r[1]* – высота полной картинки. Здесь же мы воспользуемся другим элементом массива – *$size_r[3]*, в котором содержится строка с указанием ширины и высоты изображения;

if (($z>12*$j-12) && ($z<=12*$j)) – условие, указывающее на интервал номеров выводимых картинок на страницу. Например, при *$j=1* условие будет истинно, если номера картинок (*$z*) будут соответствовать интервалу от 1 до 12;

4{

*print_r ("<td align=center id=lolo2><img src=\"kartmin/$file\" border=0 alt=\"$file\">
*

*<I id=lolo6>$file
$size_r[3]</I></td>");* – если условие *if* истинно, то 12 мини-копий картинок при помощи функции *print_r* будут выводиться в окно браузера на страницу. Подобная строка уже разобрана в прошлом листинге;

if (($z/4) == is_integer($z/4)) echo "</tr>"; – как только число загруженных картинок на странице станет кратно четырем, завершаем строку таблицы с картинками и начинаем новую. Например, пятая картинка будет выведена уже во втором ряду;

4} – окончание цикла вывода картинок на страницу;

2} – окончание действия условия *if*, проверяющего, не является ли выбранный функцией *readdir* файл (*$file*) текущим или родительским каталогом или файлом *Thumbs.db*;

1} – окончание цикла *while* перебора всех файлов в директории *kart*;

echo"</td></tr></table>"; – заканчиваем (закрываем) таблицу с картинками на странице;

echo " <p>"; – после таблицы с 12 картинками мы должны разместить ссылки на другие таблицы с другими 12 картинками. Ссылки выводим в центре, в отдельном блоке *<p>*;

for($i=1; $i<=$z/12+1; $i++) – при помощи цикла *for* выводим ссылки на остальные страницы с картинками, причем, эти ссылки будут в виде интервала чисел – порядковых номеров картинок. Страниц с картинками у нас будет много, но все они как бы будут объединены в одну. Будет меняться только параметр *$j*, и в зависимости от этого параметра на страницу будет выводиться та или иная порция из 12 картинок. К этому моменту в переменной *$z* у нас будет храниться общее число всех картинок в папке *kart*. Число *$z/12+1* означает, сколько всего должно быть страниц с картинками. Отсюда счетчик *$i* означает у нас как бы номер страницы с картинками;

5{

$k=(12*$i-11)."...".($i*12);* – формируем интервалы чисел, означающие номера картинок на странице. Например, при *$i=1 выведется интервал *1...12*, а при *$i=2* выведется интервал *13...24* и т. д.;

if ($i==$j) – проверяется условие, которое я сейчас попытаюсь объяснить. Итак, на странице, например *kartinki.php?j=1*, будут ссылки на другие таблицы с картинками (на страницы с другим параметром *$j*). Нам нужно сделать так, чтобы не срабатывала ссылка на ту страницу, на которой находится посетитель. Например, если посетитель находится на второй странице (в этом случае *$i=$j=2*), то зачем ему ссылка на вторую страницу? Выведется только

числовой интервал картинок (выполнится команда между фигурными скобками под номером 6). Параметр *id=lolo4* задает стиль шрифта из таблицы стилей *stil.css*;

6{

echo "$k";

6}

else

7{

echo "$k";* – выводим ссылки на страницы с другими параметрами *$j, на которых посетитель еще не находится;

7}

5} – окончание цикла вывода ссылок;

echo "</p> "; – закрываем блок ссылок;

?> – конец PHP-кода;

*

*

* Можете закачать на сайт свою картинку! *

<form enctype='multipart/form-data' action=zagryzka.php method=POST> – делаем форму для закачки картинок на страницу посетителями сайта. После нажатия кнопки «*Отправить*» запустится сценарий загрузки графического файла на сервер *zagryzka.php*. Его мы создадим в следующем листинге;

*<input type=FILE name=zak>

* – создаем поле с именем *name=zak* для ввода загружаемого на сервер файла;

<input type=SUBMIT name=otpr value='Отправить'> – кнопка для отправки данных;

</form> – закрываем форму.

Теперь напишем код для загрузки файлов на сервер *zagryzka.php*. Этот код идентичен коду сценария *zagryzkaObych.php* (листинг 8.3) с той лишь разницей, что в коде *zagryzka.php* не строится форма (она уже есть на странице *kartinki.php*), мини-картинки создаются другого размера и вводится ограничение на объем закачиваемого файла. В PHP-редакторе создайте новый файл и сохраните его в папке *risynki* под названием *zagryzka.php*. Наберите код:

Листинг 8.6 (файл zagryzka.php)

```
<?php
if ($_POST['otpr']==true)
{
 $zak=$_POST['zak'];
 $file=$_FILES['zak']['name'];
```

```
$size=$_FILES['zak']['size'];
if($size > 1024*1024*2)
{
 echo "Размер файла не должен превышать 2 МБ! Повторите попытку.<br>";
 exit();
}
$ext=array(".jpg",".gif",".png",".JPG",".GIF",".PNG");
if(in_array(strrchr($file, "."), $ext))
{
 if(copy($_FILES["zak"]["tmp_name"], "kart/".$file))
 {
  echo "Файл ".$file." объемом: ".$size." байт успешно закачан!<br>";
  echo "<a href=# onClick='history.back()'><button>Вернуться</button></a>";
 }
 else
 {
  echo "Ошибка при закачке файла! Повторите попытку.<br>";
  exit();
 }
}
else
{
 echo "Расширение закачиваемых файлов должно быть jpg, png или gif!
Повторите попытку.<br>";
 exit();
}
}
?>
```

Сохраните код. Этот код разбирать не буду, поскольку похожий код разобран выше (листинг 8.3). Объясню лишь пару строк:

*if($size > 1024*1024*2)* – условие, вводящее ограничение на объем закачиваемого файла (до 2 Мб). Если посетители начнут закачивать файлы, например, в 20 Мб, то они быстро заполнят все дисковое пространство, выделенное вам на сервере;

echo "<button>Вернуться</button>"; – после закачки файла нам надо вернуться со страницы *zagryzka.php* на первую страницу с картинами *kartinki.php*, т. е. назад. Для этого и существует ссылка *<a href=# onClick='history.back()'>*, которая возвращает посетителя на прошлую страницу. Ссылка сделана в виде кнопки «Вернуться».

Запустите теперь код страницы с мини-картинками *kartinki.php* (листинг 8.5). У меня, например, в папке *kart* уже имеется 48 космических картинок и результат работы данной программы изображен на рис. 8.5.

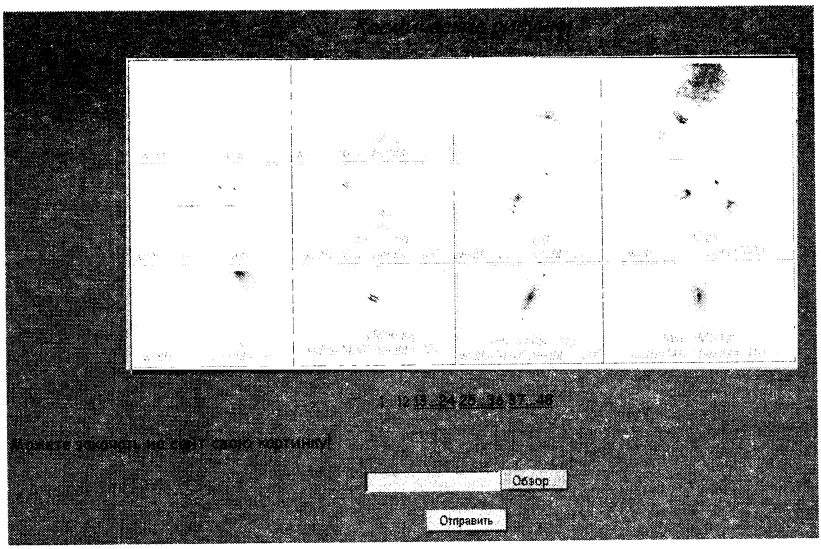


Рис. 8.5

При нажатии кнопкой мыши на любую мини-копию в отдельном окне должна появиться полная картинка. Теперь щелкните мышью на числовом интервале *13...24*. Вы должны попасть на эту же страницу с мини-картинками *kartinki.php* (рис. 8.6), но с параметром *j=2*. Посмотрите вверху на строку браузера. Вы увидите, что после названия страницы стоит параметр *j*: *kartinki.php?j=2*.

Все мини копии-картинок созданы программно и вы должны их увидеть в подпапке *kartmin*, которая, в свою очередь, была раньше создана вами в папке *risynki*.

Вернемся теперь к странице *kartinki.php* (рис. 8.5). Выберите при помощи кнопки «Обзор» какой-нибудь графический файл на вашем компьютере (файл должен быть с расширением *jpg*, *gif* или *png*) и нажмите кнопку «*Отправить*». В случае успешной закачки файла, вы, попав на страницу *zagryzka.php*, увидите в браузере надпись, примерно как на рис. 8.7.

Нажав на кнопку «*Вернуться*», вы снова вернетесь на страницу *kartinki.php* (по умолчанию мы сделали параметр *j=1*). Обновите страницу. Вы увидите вашу закачанную картинку (ее мини-копию) на какой-либо из страниц (она, например, может оказаться последней).

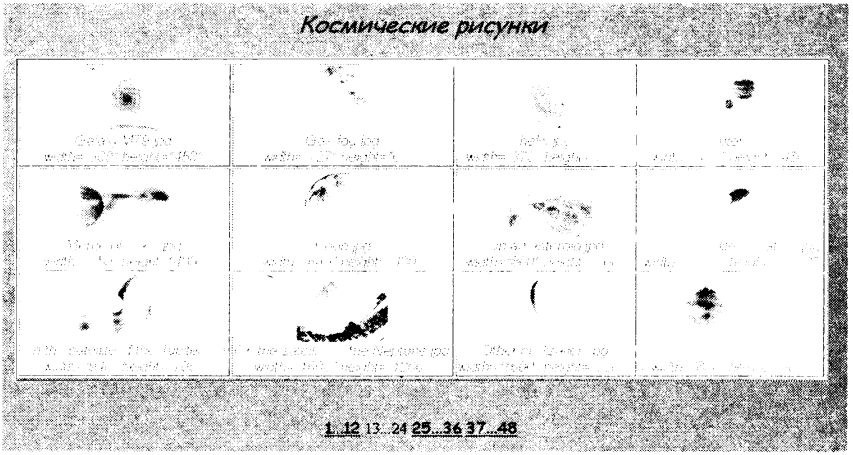


Рис. 8.6

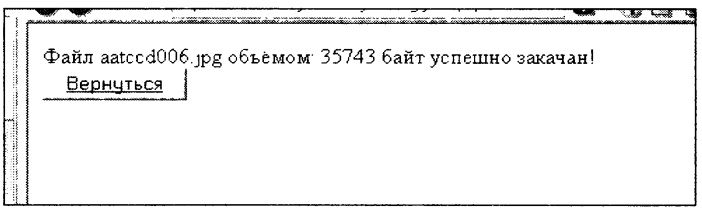


Рис. 8.7

Теперь вы можете создавать на сайте свой фотоальбом или картинную галерею. Вам только остается загружать фотографии или картинки в отдельно созданную папку. Все остальное php сделает за вас! Что касается оформления и дизайна ваших страниц, то при помощи html-тегов и таблицы стилей вы это сможете сделать самостоятельно на свой вкус.

Глава 9. ИНТЕРНЕТ-МАГАЗИН

9.1. ПРОСТЕЙШИЙ КНИЖНЫЙ ИНТЕРНЕТ-МАГАЗИН

Вот мы и дошли до Интернет-магазина. Сейчас покупка по Интернету становится таким же обычным делом, как и покупка в реальном магазине. Пока данный вид торговли в нашей стране составляет малую долю от общего товарооборота, но в недалеком будущем, я уверен, большинство покупателей будут приобретать товары сидя за компьютером, а не бегать по магазинам за поиском того или иного товара. Вы заказываете товар, оставляя свои контактные данные и адрес на сайте магазина, и получаете через несколько дней свой заказ. Расплачиваться можно либо наложным платежом, при получении заказа, либо сразу электронными деньгами (об этом чуть позже). Мы создадим на сайте простой Интернет-магазин. С помощью PHP можно легко сделать мини-Интернет-магазин, т. е. установить на сайт форму заказа, которая будет отправляться вам по электронной почте. Посетитель выбирает товар, оставляет в текстовых полях формы свой электронный и домашний адреса и отправляет заказ вам по электронной почте. Причем отправка осуществляется автоматически, при нажатии в форме кнопки «Отправить». Наш электронный магазин будет выставлять на продажу книги, хотя вы сами можете выставить все, что угодно. Продаваемые книги будут выведены в ячейки четырехколонной таблицы. У меня, например, там выведено 6 книг (рис. 9.1). Посетитель, придя на главную страницу магазина (она у нас будет называться *magazin.php*), выберет себе книгу, количество экземпляров. Внизу будет показана общая стоимость выбранных книг. Если он захочет узнать какую-либо информацию о книге, то ему будет достаточно нажать кнопку мыши на ее названии, и тогда в отдельном окне выведется интересующая нас информация, а также фотография товара (книги) (рис. 9.2).

После заполнения текстовых полей «Ваше имя», «Ваш E-mail», «Адрес доставки», посетитель нажимает кнопку «Заказать». Все, заказ направляется на ваш электронный адрес и выводится в удобочитаемом виде, а посетитель направляется на страницу, где видит свой заказ и благодарность за него (рис. 9.3).

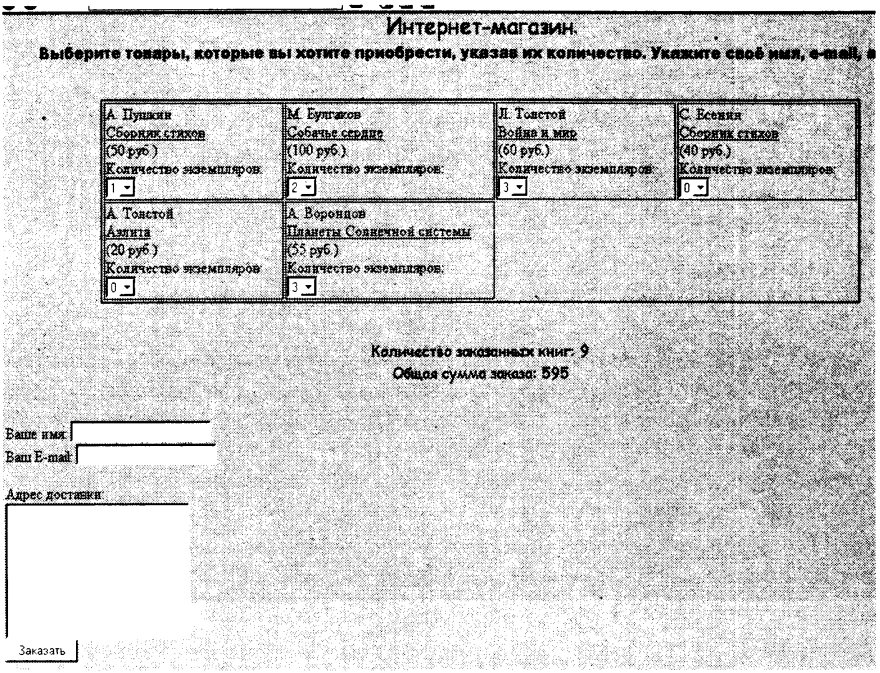

Интернет-магазин
Выберите товары, которые вы хотите приобрести, указав их количество. Укажите своё имя, e-mail, а
А. Пушкин
Сборник стихов
(50 руб.)
Количество экземпляров:
1
М. Булгаков
Собачье сердце
(100 руб.)
Количество экземпляров:
2
Л. Толстой
Война и мир
(60 руб.)
Количество экземпляров:
3
С. Есенин
Сборник стихов
(40 руб.)
Количество экземпляров:
0
А. Толстой
Аэлита
(20 руб.)
Количество экземпляров:
0
А. Воронцов
Планеты Солнечной системы
(55 руб.)
Количество экземпляров:
3
Количество заказанных книг: 9
Общая сумма заказа: 595
Ваше имя:
Ваш E-mail:
Адрес доставки:
Заказать

Рис. 9.1

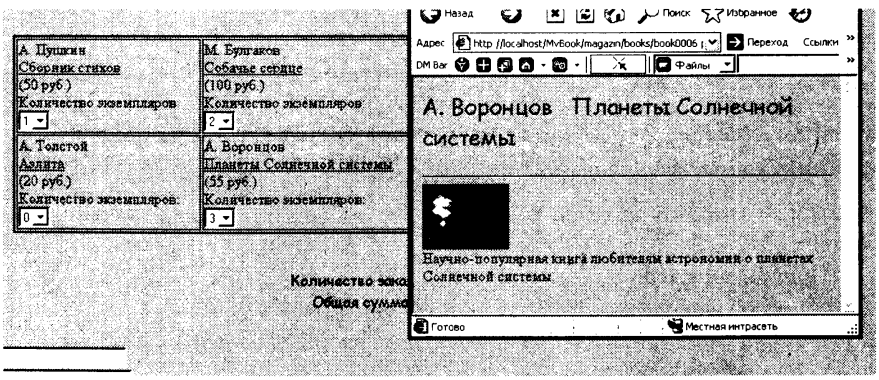

А. Пушкин
Сборник стихов
(50 руб.)
Количество экземпляров:
1
М. Булгаков
Собачье сердце
(100 руб.)
Количество экземпляров:
2
А. Толстой
Аэлита
(20 руб.)
Количество экземпляров:
0
А. Воронцов
Планеты Солнечной системы
(55 руб.)
Количество экземпляров:
3
Назад
Поиск
Избранное
Адрес
http://localhost/MyBook/magazin/books/book0006.
Переход
Ссылки
DM Bar
Файлы
А. Воронцов Планеты Солнечной системы
Научно-популярная книга любителям астрономии о планетах Солнечной системы
Готово
Местная интрасеть

Рис. 9.2

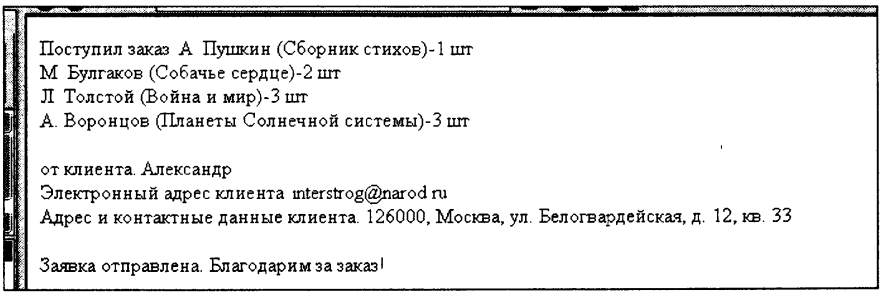


Рис. 9.3

Для отправки заказа на ваш электронный адрес используется встроенная php-функция *mail(email, tema, soob, zagolovki)*. Первый аргумент *email* – электронный адрес получателя, т. е. ваш электронный адрес. Второй аргумент *tema* – тема сообщения, третий аргумент *soob* – само сообщение. Четвертый, необязательный аргумент *zagolovki* – заголовки сообщения. Заголовки бывают следующих видов:

From – содержит электронный адрес отправителя, т. е. в нашем случае – адрес посетителя, сделавшего заказ в магазине;

Reply-To – позволяет указывать адрес для ответа на сообщение;

Content-Type – указывает тип содержимого сообщения (text/plain или text/html). Например, для нормальной отправки сообщения на русском языке необходимо передать заголовок:

Content-Type: text/plain; charset=Windows-1251;

Content-Transfer-Encoding – указывает количество бит для передачи символа (для сообщений на русском языке необходимо 8 бит). Для нормальной отправки сообщения на русском языке необходимо передать заголовок:

Content-Transfer-Encoding: 8bit.

После каждого заголовка обязательно указывается символ новой строки *\n*.

Приведу пример:

mail("interstrog@narod.ru", "Тест", "Всем привет!", "From: noname@narod.ru\nReply-To: noname@narod.ru\nContent-Type: text/plain; charset=windows-1251").

Функция *mail* отправит на адрес *interstrog@narod.ru* сообщение от посетителя с адресом *noname@narod.ru*. В сообщении с темой «Тест» будет фраза «Всем привет!».

Новый товар в магазин будет добавляться вами через административную панель (файл *admin.php*), где вы будете указывать автора и название посту-

пившей в продажу книги, опишите эту книгу и загрузите на сервер фотографию товара (рис. 9.4). После нажатия вами кнопки «Отправить» будет создана новая PHP-страница для данного товара и фотография. Все данные о товаре, введенные вами в административной панели, и его фотография и будут выведены на созданной странице, которая будет вызываться кликом мыши на названии книги на главной странице магазина (см. рис. 9.2).

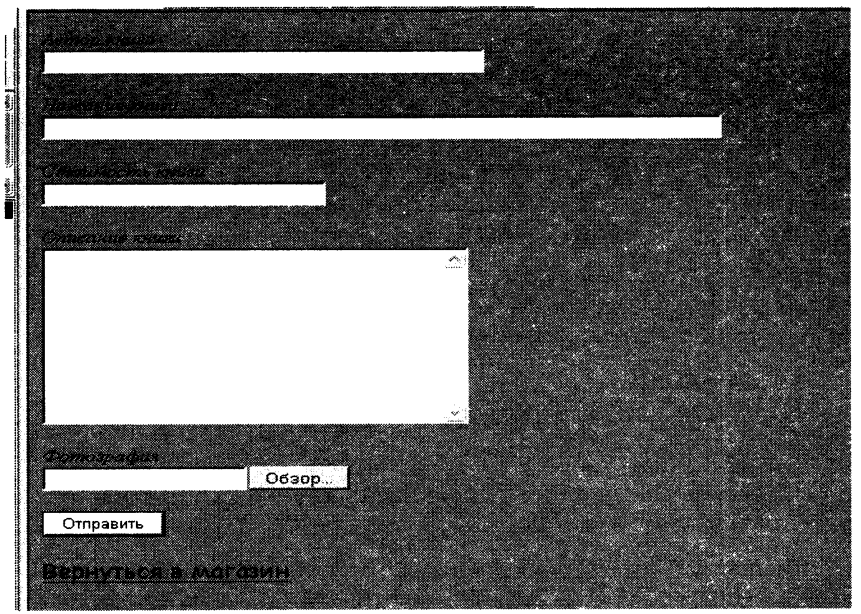


Рис. 9.4

Вход на страницу *admin.php* мы, естественно, закроем паролем, чтобы заходить на нее смогли бы только вы. Для этого надо вспомнить, как мы создавали административную панель для редактирования и написания новостей для сайта (файл *redaktNovost.php* в гл. 5).

Если вы разобрались с предыдущими главами, то код нашего электронного магазина не должен вызвать у вас затруднений. Если что-то подзабыли, повторите разбор кода для файла *redaktNovost.php*, а также разбор кода для файла *statji.php* (листинг 6.3). В файле *statji.php* мы использовали теги *<meta>* для вывода названий и авторов статей. Нечто похожее мы будем использовать и в этой главе. В теги *<meta>* мы будем записывать автора книги, название книги, цену за книгу и выводить содержимое этих тегов на главную страницу магазина.

Итак, в рабочей папке *htdocs* создайте новую подпапку *magazin*. Именно в ней мы и будем хранить все файлы нашего электронного магазина. Чтобы

не «изобретать велосипед» в созданную подпапку *magazin* скопируем созданные раньше программы, которые нам понадобятся. Из папки *risynki*, которую мы создали в прошлой главе, скопируйте файл *zagryzka.php* (листинг 8.6) и вставьте в папку *magazin*. Эта программа, если помните, загружает графические файлы на сервер. Мы ее немного изменим. Откройте файл *zagryzka.php* в PHP-редакторе (открывать нужно из папки *magazin*!). Уберите в начале кода строку вместе с открывающейся фигурной скобкой:

if ($_POST['otpr']==true)

{.

В конце кода нужно, естественно, убрать и одну закрывающуюся скобку:

}.

Измените строку:

if(copy($_FILES["zak"]["tmp_name"], "kart/".$file)) на строку:

if(copy($_FILES["zak"]["tmp_name"], "books/fotoshop/".$file)). У нас фотографии товаров будут храниться в папке *fotoshop*, которая, в свою очередь, будет находиться в папке *books* (эти папки создадим позже).

Еще удалите строку:

echo "<button>Вернуться</button>", поскольку она здесь не будет нужна.

Если вы все сделаете правильно, то файл *zagryzka.php* из папки *magazin* будет выглядеть, как в листинге 9.1.

Листинг 9.1 (файл zagryzka.php из папки magazin)

```
<?php
 $zak=$_POST['zak'];
 $file=$_FILES['zak']['name'];
 $size=$_FILES['zak']['size'];
 if($size > 1024*1024*2)
 {
  echo "Размер файла не должен превышать 2 МБ! Повторите попытку.<br>";
  exit();
 }
 $ext=array(".jpg",".gif",".png",".JPG",".GIF",".PNG");
 if(in_array(strrchr($file, "."), $ext))
 {
  if(copy($_FILES["zak"]["tmp_name"], "books/fotoshop/".$file))
  {
   echo "Файл ".$file." объемом: ".$size." байт успешно закачан!<br>";
  }
```

```
 else
 {
  echo "Ошибка при закачке файла! Повторите попытку.<br>";
  exit();
 }
}
 else
 {
  echo "Расширение закачиваемых файлов должно быть jpg, png или gif!
Повторите попытку.<br>";
  exit();
}
?>
```

Это почти копия листинга 8.6 из гл. 8, поэтому разбирать я его не буду.

Далее из папки *novosti* скопируйте файл для создания пароля *newparol.php* (листинг 5.5 из гл. 5) и тоже вставьте его в папку *magazin*. В эту же папку скопируйте и листы стилей *stil.css, stili2.css* (я их создавал с такими названиями) либо создайте новый файл стилей, который вы будете использовать для оформления текста.

В папке *magazin* создайте теперь новую подпапку *books*, где будут находиться программно созданные страницы и картинки для каждого товара. Теперь в подпапку *books* скопируйте опять листы стилей, чтобы текст на программно созданных страницах был тоже красиво оформлен. И, наконец, если вы хотите хранить фотографии товаров в отдельной папке, то ее тоже нужно создать. Создайте в папке *books* подпапку *fotoshop*, где и будут храниться эти фотографии (названия всех подпапок можете задать свои, но при этом вы должны указать эти названия в кодах программ). Итак, в главной рабочей папке *htdocs* мы создали папку *magazin*. В папке *magazin* создали папку *books*, а в папке *books* создали папку *fotoshop*. Теперь можно создавать основные страницы для нашего магазина. Их будет 3, и все они должны быть сохранены в папке *magazin*. Первый файл *magazin.php* будет являться главной страницей или витриной вашего магазина, на которой будут выставлены товар и форма для отправки заказа от посетителя (рис. 9.1). Там же будет находиться и калькулятор для подсчета общей стоимости выбранных книг. Второй файл *magazin2.php* будет обрабатывать данные посетителя, показывать текст с заказом в браузере (рис. 9.3) и отправлять этот текст к вам на E-mail. И, наконец, третий файл *admin.php* будет являться вашей административной панелью с формой, при помощи которой вы сможете добавлять новый товар (книгу) на витрину магазина и закачать фотографию для данного товара. Эта же программа создает новую PHP-страницу для каждого товара с описанием и фотографией этого товара (рис.9.2), которую посетитель при желании может посмотреть.

9.2. АДМИНИСТРАТИВНАЯ ПАНЕЛЬ ДЛЯ МАГАЗИНА

Начнем мы с написания файла *admin.php* (рис. 9.4) и потом создадим при его помощи страницы с описанием и фотографиями наших товаров. Вы просто будете добавлять новый товар на сервер, все остальное php сделает за вас (разместит товар на витрине и создаст отдельную для него страницу). При входе на страницу *admin.php* нужно будет ввести пароль. Создать зашифрованный пароль можно при помощи программы *newparol.php*, как мы это делали в гл. 5. Этот файл вы должны были скопировать в папку *magazin*. Запустите в PHP-редакторе файл *newparol.php* из папки *magazin*. Введите пароль (запомните его!) и нажмите кнопку «Ввести пароль» (см. рис. 9.5).

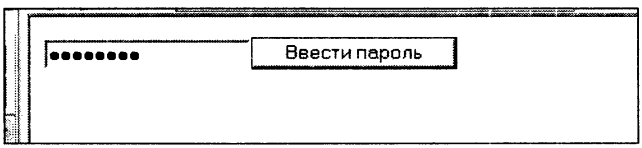


Рис. 9.5

В папке *magazin* у вас должен появиться текстовый файл *parol.txt* с зашифрованным паролем. Пароль зашифрован при помощи встроенной функции необратимого шифрования *md5* (если забыли, что это такое, см. гл. 5).

Создайте в PHP-редакторе новый файл и сохраните его в папке *magazin* под названием *admin.php*. Наберите код из листинга 9.2 (желательно, сначала разберитесь в нем, а потом наберите самостоятельно).

Листинг 9.2 (файл admin.php)

```
<html>
<head>
<title>Новый товар</title>
<link rel="stylesheet" type="text/css" href="stil.css">
</head>
<body>
<?php
if(!$_POST['parol'])
{
 echo "<form action=admin.php method=POST>";
 echo "<input type=PASSWORD name=parol>";
 echo "<input type=SUBMIT name=knopka value='Ввести пароль'>";
 echo "</form><br><br>";
}
```

```
$parol=$_POST['parol'];
$file="parol.txt";
$open=fopen($file, "r");
$read=fread($open, filesize($file));
fclose($open);
if(md5($parol)==$read)
{
 $knigi="books";
 $dir=opendir($knigi);
 while($file=readdir($dir))
 {
  if (($file!=".")&&($file!=".."))   $j=$j+1;
 }
 closedir($dir);
 echo "<form enctype='multipart/form-data' action='admin.php' method='POST'
name='forma'>
 <i>Автор книги</i><br>
 <input type=TEXT name='avtor' size=50><br><br>
 <i>Название книги</i><br>
 <input type=TEXT name='nazvanie' size=80><br><br>
 <i>Стоимость книги</i><br>
 <input type=TEXT name='cena' size=30><br><br>
 <i>Описание книги</i><br>
 <textarea name='opisanie' rows=10 cols=50></textarea><br><br>
 <i>Фотография</i><br>
 <input type=FILE name='zak'><br><br>
 <input type=hidden name=parol value=$parol>
 <input type=SUBMIT name='otpr' value='Отправить'><br><br>";
 if (($_POST['otpr']==true) && ($_POST['nazvanie']<>""))
 {
  $avtor=$_POST['avtor'];
  $nazvanie=$_POST['nazvanie'];
  $cena=$_POST['cena'];
  $opisanie=$_POST['opisanie'];
  $next=$j+1;
  $next="$knigi/book000$next.php";
  $prev="$knigi/book000$j.php";
  include "zagryzka.php";
  $new=fopen($next, "w");
```

```
 $list='<html><head>
 <title>Книга</title>
 <meta name="avtor" content="'."$avtor".'">
 <meta name="nazvanie" content="'."$nazvanie".'">
 <meta name="stoimost" content="'."$cena".'">
 <link rel="stylesheet" type="text/css" href="stili2.css">
 </head>
 <body bgcolor="cyan">
 <?php
 $teg=get_meta_tags(basename($_SERVER[\'PHP_SELF\']));
 echo "<div id=lolo3>$teg[avtor]   $teg[nazvanie]</div><br><hr>";
 $file="'.$file.'";
 if (file_exists("fotoshop/$file")) echo "<img src=\"fotoshop/$file\" border=0><br>";
 $opisanie="'.$opisanie.'";
 echo $opisanie;
 ?>
 </body>
 </html>';
 fwrite($new, $list);
 fclose($new);
 $new=fopen($next, "r");
 $old=fopen ($prev, "r");
 $oldread=fread($old, 1024);
 $newread=fread($new, 1024);
 if ($oldread===$newread)
 {
  fclose($new);
  unlink($next);
  fclose($old);
 }
 else
 {
  fclose($new);
  fclose($old);
 }
}
 echo "<a href='magazin.php'>Вернуться в магазин</a>";
}
else if($parol)
```

```
{
 echo "Пароль неверен!";
}
?>
</body>
</html>
```

Разберем некоторые строчки кода:

<html>

<head>

<title>Новый товар</title>

<link rel="stylesheet" type="text/css" href="stil.css"> – указывам на таблицу стилей *stil.css*, которая должна находиться в той же папке (*magazin*), что и данная программа (*admin.php*);

</head>

<body>

<?php

if(!$_POST['parol']) – если пароль еще не был получен, а при запуске данной программы это будет действительно так, элемент суперглобального массива *$_POST['parol']* еще не будет определен и выполнится программный код между фигурными скобками под номером 1, т. е. выведется форма для ввода пароля. После ввода пароля в текстовое поле с именем *name=parol* и нажатии кнопки «Ввести пароль», мы опять попадем на эту же страницу *admin.php* (в теге <form> указан параметр *action=admin.php*), но элемент *$_POST['parol']* уже будет определен (он будет содержать пароль, введенный вами в текстовое поле с именем *name=parol*). В этом случае код между фигурными скобками под номером 1 уже не будет выполняться;

1{

echo "<form action=admin.php method=POST>";

echo "<input type=PASSWORD name=parol>";

echo "<input type=SUBMIT name=knopka value='Ввести пароль'>";

*echo "</form>

";*

1} – окончание кода, выполняющегося, если еще не нажата кнопка «Ввести пароль». Если кнопка уже нажата, то выполнится только код после этой закрывающейся скобки;

$parol=$_POST['parol']; – присваиваем переменной *$parol* пароль, который был введен ранее в текстовое поле для ввода пароля;

$file="parol.txt";* – переменной *$file присваиваем название файла с нашим зашифрованным паролем, который мы создали раньше при помощи программы *newparol.php*;

$open=fopen($file, "r"); – открываем файл с паролем для чтения;

$read=fread($open, filesize($file));* – считываем содержимое файла и сохраняем его в переменной *$read;

fclose($open); – закрываем файл с паролем;

if(md5($parol)==$read) – далее мы сравниваем введенный пароль (сначала зашифровав его при помощи функции *md5*) с зашифрованным паролем в файле *parol.txt*. Если пароли совпадают, то выполнится весь код между фигурными скобками под номером 2;

2{

$knigi="books";* – присваиваем переменной *$knigi название каталога, где у нас будут содержаться страницы для каждого товара. Этот каталог (*books*) вы должны были создать в папке *magazin*;

$dir=opendir($knigi); – открываем каталог *books* и присваиваем ему идентификатор *$dir*;

while($file=readdir($dir)) – знакомый нам цикл, который перебирает все файлы в каталоге, присваивая переменой *$file* название очередного файла из каталога при каждом выполнении цикла. Чуть раньше мы присваивали переменной $file название файла с нашим зашифрованным паролем *parol.txt*. Здесь мы снова используем эту переменную, но для других целей. Старое значение переменной *$file="parol.txt"* можно «затереть», так как оно уже нам не понадобится. После цикла *while* данную переменную можно использовать вновь для других целей, что мы и сделаем позже;

3{

if (($file!=".")&&($file!=".")) $j=$j+1; – данный цикл *while*, как вы уже поняли, просто подсчитывает общее количество файлов в каталоге *books*, т. е. количество страниц для товаров (или количество товаров нашего магазина). Зачем нам это нужно, поймете позже;

3}

closedir($dir); – закрываем каталог;

echo "<form enctype='multipart/form-data' action='admin.php' method='POST' name='forma'> – выводим форму с полями для ввода данных нового товара: автора и названия книги, цены книги, описания, фотографии данного товара. После нажатия кнопки «Отправить» все эти данные отправятся на эту же страницу (*action='admin.php'*). Все это вам должно быть знакомо. Я лишь объясню смысл некоторых полей;

*Автор книги
*

*<input type=TEXT name='avtor' size=50>

*

*Название книги
*

*<input type=TEXT name='nazvanie' size=80>

*

*Стоимость книги
*

*<input type=TEXT name='cena' size=30>

*

*Описание книги
*

*<textarea name='opisanie' rows=10 cols=50></textarea>

*

*Фотография
*

*<input type=FILE name='zak'>

* – поле для закачки с вашего компьютера фотографии товара;

<input type=hidden name=parol value=$parol>* – передаем в скрытом поле переменную *$parol. Дело в том, что после нажатия кнопки «Отправить», элемент суперглобального массива *$_POST['parol']* вновь будет не определен и опять начнет выполняться код между фигурными скобками под номером 1, т. е. нам опять придется вводить пароль. Поэтому мы вводим скрытый элемент с именем *name=parol* и со значением *value=$parol*. В результате после нажатия кнопки «Отправить» элемент *$_POST['parol']* будет уже вновь определен и ему будет присвоено значение *$parol*, т. е. введенный нами пароль. Если помните, мы такой прием уже делали в файле для редактирования новостей сайта *redaktNovost.php* в гл. 5;

*<input type=SUBMIT name='otpr' value='Отправить'>

";* – заполнив все данные для нового товара, нажимаем на кнопку «Отправить», которая имеет имя *name='otpr'*. Все данные передадутся, как было отмечено выше, на эту же страницу;

if ($_POST['otpr']==true) – после нажатия кнопки элемент суперглобального массива *$_POST['otpr']* будет иметь значение *true*. Код между фигурными скобками под номером 4 начнет выполняться только после нажатия кнопки «Отправить»;

4{

$avtor=$_POST['avtor']; – сохраняем в переменных содержание заполненных текстовых полей: автора книги, названия книги, цену книги, описание книги. Все эти данные, как вам должно быть известно, были переданы на эту страницу методом *POST*, после нажатия в форме кнопки «Отправить», и сохранены в массиве *$_POST*;

$nazvanie=$_POST['nazvanie'];

$cena=$_POST['cena'];

$opisanie=$_POST['opisanie'];

$next=$j+1; – переменной *$next* присваиваем значение на единицу больше, чем общее количество листов для описания товаров в папке *books* (общего количества товаров). Дело в том, как вы увидите тремя строчками кода ниже, страницы для описания товара и размещения там его фотографии будут создаваться автоматически и называться так: *book000X.php*, где X – порядковый номер страницы в папке *books* (название этой папки, если помните, находится в переменной *$knigi*). Если, к примеру, число товаров у нас равно 5, то последняя созданная для пятого товара страница будет иметь название *book0005.php*. Если мы отправим данные для нового (шестого) товара, то должна быть автоматически сформирована новая страница *book0006.php*;

$next="$knigi/book000$next.php";* – название (путь) для новой страницы, которая еще только сформируется, сохраняем в переменной *$next;

$prev="$knigi/book000$j.php";* – в переменной *$prev сохраняем путь к последней, ранее сформированной странице. Эта переменная нам понадобится позже;

include "zagryzka.php"; – включаем в эту страницу код для загрузки графических файлов на сервер. Файл *zagryzka.php* вы должны были скопировать из папки *risynki* в папку *magazin* и немного изменить этот файл (см. разд. 9.1 и листинг 9.1);

$new=fopen($next, "w"); – создаем новый файл (новую PHP-страницу) для нового товара. Путь к новому файлу указан в переменной *$next*. Новой странице присваиваем идентификатор *$new*;

$list='<html><head>* – далее в переменную *$list мы будем писать код для созданной страницы;

<title>Книга</title>

<meta name="avtor" content="'."$avtor".'">* – на новой странице создаем мета-теги. Зачем нужны эти теги и как их использовать, рассказано в гл. 6, где мы при помощи мета-тегов выводили в браузер название и автора той или иной статьи. Здесь мы переносим переменные *$avtor, *$nazvanie* и *$cena* со страницы, где находится весь этот код (*admin.php*), на новую создающуюся страницу в значение мета-тега *content*. Этот метод (переноса переменных на новые формирующиеся страницы) мы использовали в прошлых главах, например при формировании новых PHP-страниц для вывода мини-картинок в прошлой главе. Содержимое в мета-тегах, а именно, значения *name* и *content* заключаются в двойные кавычки;

<meta name="nazvanie" content="'."$nazvanie".'">

<meta name="stoimost" content="'."$cena".'">

<link rel="stylesheet" type="text/css" href="stili2.css"> – на новой странице указываем на лист стилей, который должен находиться в папке *book*. У меня, например, он называется *stili2.css*;

</head>

<body bgcolor="cyan">

<?php

$teg=get_meta_tags(basename($_SERVER[\'PHP_SELF\'])); – элемент суперглобального массива *$_SERVER['PHP_SELF']* возвращает путь к странице, на которой находится посетитель. В данном случае он возвратит путь к этой новой странице, содержимое которой мы и записываем сейчас в переменную *$list*. Здесь мы используем символ экранирования «\» для одинарных кавычек, поскольку эти кавычки находятся внутри тоже одинарных кавычек (все содержание переменной *$list* заключено в одинарные кавычки). Само понятие экранирования объяснено в гл. 4. Функция *basename*, если помните, оставляет только название страницы с расширением. Функция *get_meta_tags* считывает содержимое мета-тегов из страницы, в данном случае из этой новой страницы, т. е. считывает содержимое мета-тегов, которые мы создали несколькими строчками раньше. Эта функция возвратит массив *$teg* (мы так его назовем) с тремя элементами: *$teg[avtor]* (туда запишется автор книги из переменной *$avtor*, которая была указана в параметре *content* мета-тега *<meta>*), *$teg[nazvanie]* (туда запишется название книги из переменной *$nazvanie)*, *$teg[cena]* (туда запишется цена книги из переменной *$cena*);

*echo " $teg[avtor] $teg[nazvanie]
<hr>";* – выводим на новой странице содержимое мета-тегов, а именно автора книги и название книги через пробел между ними (оператор * * обозначает пробел);

$file="".$file.""; – передаем на новую страницу переменную *$file*. Как вы думаете, что она будет содержать? Перед тем как начать записывать код для новой страницы в переменную *$list*, мы вставили на эту (*admin.php*) страницу код файла для загрузки графических изображений при помощи команды *include "zagryzka.php"*. Так вот, в этом файле (*zagryzka.php*) в переменную *$file* записывается название графического файла с фотографией, которую мы и закачали на сервер для данного товара. Теперь эту переменную мы переносим на новую страницу;

*if (file_exists("fotoshop/$file")) echo "<img src=\"fotoshop/$file\" border=0>
";* – если фотография для нового товара была закачана, т. е. файл *$file* в папке *fotoshop* существует, то это фото выводится на новой странице. Здесь мы тоже применяем символ экранирования для двойных кавычек, которые расположены внутри двойных кавычек;

$opisanie="".$opisanie.""; – также на новую страницу переносим переменную *$opisanie*, в которой находится введенное вами описание для данного товара;

echo $opisanie; – выводим это описание на создаваемую страницу. Итак, на образованной новой странице у нас будет написано: автор и название но-

вой книги, выведена фотография товара, а под фотографией – описание данного товара (см. рис. 9.2);

?>

</body>

</html>'; – окончание кода для новой страницы, заканчиваем писать переменную *$list*;

fwrite($new, $list);* – вставляем теперь код, записанный в переменной *$list, на новую страницу, которой мы присвоили идентификатор *$new*;

fclose($new); – закрываем созданную страницу;

$new=fopen($next, "r"); – далее следуют строчки кода, которые опять открывают только что созданную страницу, открывают предыдущую созданную страницу, считывается и потом сравнивается содержимое этих страниц. Зачем это нужно? Попробую объяснить. Если вы запустите файл с формой (*admin.php*) у себя на компьютере, введете данные для нового товара, загрузите фото для товара и нажмете на кнопку «Отправить», то в браузере появится форма (рис. 9.6).

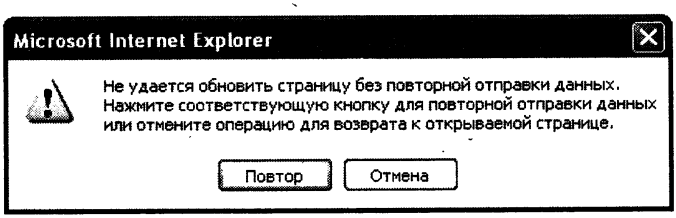


Рис. 9.6

При нажатии кнопки «Отмена» ничего не произойдет. Вам останется нажать только кнопку «Повтор» и введенные вами данные повторно отправятся на сервер, а программа *admin.php* снова начнет формировать новую страницу с теми же данными. В результате в папке *books* появяться две страницы с абсолютно одинаковым содержимым, т. е. одна и та же книга будет потом добавлена на витрину магазина дважды. Позже, ради эксперимента, попробуйте удалить строки кода, начиная со строки: *$new=fopen($next, "r")* до закрывающейся фигурной скобки под номером 6 включительно (то, что отмечено курсивом в листинге 9.2), и посмотрите потом, как будет работать программа. Она будет дважды выводить на витрину магазина один и тот же товар. Поэтому, чтобы такого не было, мы сравниваем только что созданную страницу с предыдущей, и если они идентичны, то новая страница удаляется;

$old=fopen ($prev, "r");

$oldread=fread($old, 1024);

$newread=fread($new, 1024);

if ($oldread===$newread) – если содержимое страниц одинаково (новой и предыдущей), то выполнится код в фигурных скобках под номером 5;

5{

fclose($new);

unlink($next); – удаляем созданную новую страницу при помощи встроенной php-функции *unlink* (такая функция упоминалась в гл. 4);

fclose($old);

5}

else – иначе, если содержимое страниц разное, то ничего не удаляем, а просто закрываем эти страницы при помощи функции *fclose*;

6{

fclose($new);

fclose($old);

6}

4} – окончание кода, который выполнится после нажатия кнопки «Отправить» в форме;

echo "Вернуться в магазин"; – делаем ссылку для возврата на главную страницу (витрину) магазина. Эту страницу (*magazin.php*) мы пока не создали;

2} – окончание кода, который выполнится, если введенный вами пароль для входа на страницу *admin.php* был верен;

else if($parol) – иначе, если пароль был введен (переменная *$parol* существует), но введен неверно, то выведется об этом надпись;

7{

echo "Пароль неверен!";

7}

?>

</body>

</html> – окончание кода.

В PHP-редакторе запустите эту программу. Сначала вам предложат ввести пароль. Введите тот пароль, который вы создали при помощи программы *newparol.php*. Далее заполните текстовые поля, хотя бы как на рис. 9.7.

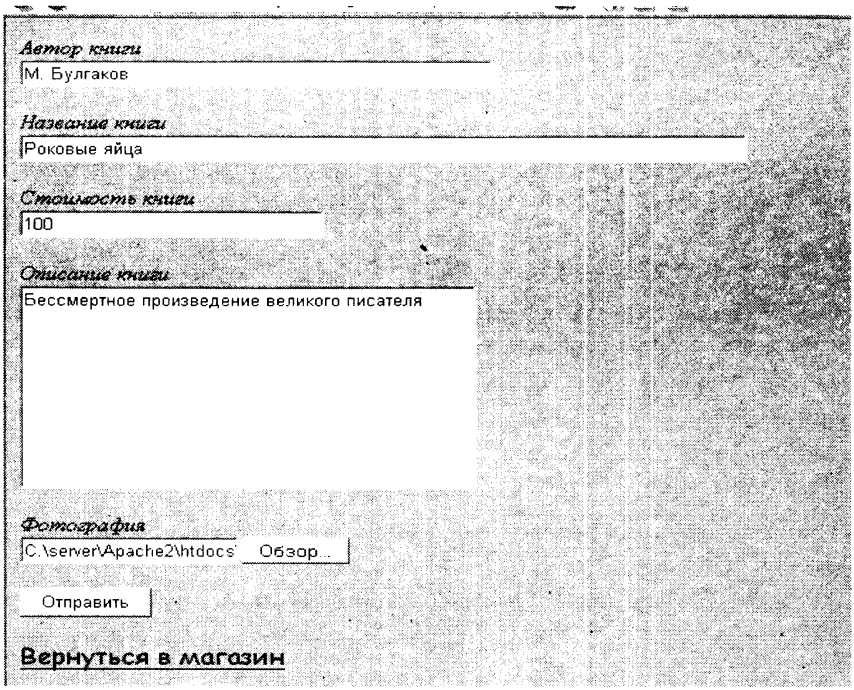


Рис. 9.7

В поле «фотография» выберите какой-нибудь рисунок (формата *gif*, *jpg* или *png*) на вашем компьютере (можно любой, поскольку фотографии данного товара (книги) у вас, скорее всего, нет). Нажмите кнопку отправить. Если вы все сделали правильно, то в папке *books* у вас должна появиться страничка *book0001.php* (можете тоже запустить ее в редакторе), содержащая название и описание книги, а также закачанную вами фотографию. Код программно созданной страницы может быть таким, как в листинге 9.2, а.

Листинг 9.2, а (файл book0001.php. Его создавать не нужно! Он создастся автоматически, т. е. программно)

```
<html>
<head>
 <title>Книга</title>
 <meta name="avtor" content="М. Булгаков">
 <meta name="nazvanie" content="Роковые яйца">
 <meta name="stoimost" content="100">
 <link rel="stylesheet" type="text/css" href="stili2.css">
 </head>
```

```
<body bgcolor="cyan">
<?php
$teg=get_meta_tags(basename($_SERVER['PHP_SELF']));
echo "<div id=lolo3>$teg[avtor]   $teg[nazvanie]</div><br><hr>";
$file="Bulgakov.jpg";
if (file_exists("fotoshop/$file")) echo "<img src=\"fotoshop/$file\" border=0><br>";
$opisanie="Бессмертное произведение великого писателя";
echo $opisanie;
?>
</body>
</html>
```

Как видно из кода, введенные нами данные о книге (автор, название, цена) будут храниться в мета-тегах *<meta>*. Данные об авторе книги и ее названии чуть ниже выведутся в браузер. Затем выведется фотография книги *Bulgakov.jpg*, а под ней описание книги. Сама фотография должна быть в подпапке *fotoshop*. Итак, первый товар на витрине магазина у вас есть.

9.3. ИСПОЛЬЗОВАНИЕ JAVASCRIPT В PHP-СЦЕНАРИЯХ

Составим теперь код главной страницы или витрины нашего магазина (см. рис. 9.1 и листинг 9.3). Сначала разберите этот код (разбор кода сразу после листинга 9.3 в п. 9.4), а потом постарайтесь набрать его самостоятельно. Часть кода (выделено курсивом в листинге) написана на **JavaScript** (между тегами *<script>* и *</script>*). Это калькулятор для подсчета общей стоимости книг, выбранных посетителем. Объясню вкратце, что такое JavaScript и как этот язык используют в сценариях.

JavaScript – это мощный и универсальный язык подготовки сценариев. С его помощью вы можете переделать простейшие web-страницы в динамичные страницы с мгновенным откликом на действия пользователя (например, реагирование на нажатие кнопки на web-странице). Главное достоинство JavaScript – простота в применении. Поскольку частично основой JavaScript послужил язык Java, в нем предусмотрен объектно-ориентированный подход к созданию сценариев, в соответствии с которым мы можем работать как с собственными функциями и объектами, так и со встроенными объектами. JavaScript обеспечивает оперативную реакцию web-страниц на действия пользователя. Вообще, JavaScript предназначен для создания «клиентских» сценариев, т. е. сценариев, выполняемых браузером читателя. Основной частью сценария (программы на языке JavaScript) является описание событий и обработчиков этих событий. События происходят, главным образом, теми или иными действиями пользователя. Например, щелчку мышью на некото-

ром элементе страницы соответствует событие *onclick*. Если указатель мыши оказывается над какой-либо областью страницы, имеет место событие *onMoitseOver*. Таким образом, суть выполнения сценария заключается в воспроизведении реакции на события, происходящие при работе с HTML-документом. Так, в результате щелчка на кнопке может, например, открываться новое окно браузера.

Для добавления сценария JavaScript на web-страницу используется пара тегов *<script>* и *</script>*. Все, что находится между этими двумя тегами, браузер рассматривает как сценарий на JavaScript. Тег *<script>* обязательно должен содержать параметр *language="JavaScript"*, который и определяет язык сценария. Сценарий JavaScript может находиться между тегами *head* (такие сценарии анализируются браузером в первую очередь), после тега *body* (такие сценарии реализуются во вторую очередь), непосредственно в обработчиках событий (сценарии запускаются при выполнении того или иного события), в отдельном файле (можно создать сценарий в отдельном файле с расширением *js*). Напишем простенький сценарий. Создайте в PHP-редакторе новый PHP-файл и сохраните его в нашей рабочей папке *htdocs* под названием *java.php*. Наберите следующий код:

```
<html>
<head>
 <title>Сценарии</title>
 <script language="JavaScript">
 alert("Это сценарий на JavaScript, первая запись");
 </script>
</head>
<body>
<script language="JavaScript">
document.write("Это сценарий на JavaScript, вторая запись");
</script>
<br>
<input type="BUTTON" value="Нажми меня" onclick="alert('Это сценарий на
JavaScript, третья запись')">
<?php
?>
</body>
</html>
```

Первый сценарий у нас расположен между тегами *<head>* и *</head>*. Команда *alert* выводит специальное окно с сообщением, которое записано у нас в скобках (рис. 9.8). Данный сценарий выполнится сразу после запуска скрипта.

Рис. 9.8

После нажатия кнопки *OK* или кнопки закрытия окна выполнится второй сценарий после тега *<body>*. Команда *document.write* выводит строку уже в окно браузера. Окно браузера представляет собой объект *window*. Этот объект содержит другие объекты, например в нашем случае это объект *document* (наша PHP-страничка). В этом объекте при помощи команды *write* мы и выводим вторую строку (рис. 9.9).

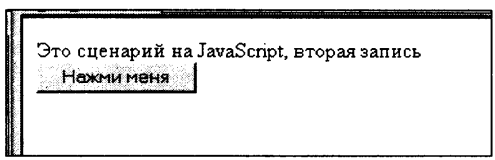


Рис. 9.9

Наконец, третий сценарий находится в обработчике события *onclick*. Данное событие наступает при нажатии кнопки. Кнопка создается при помощи тега *<input>* с параметром *type="BUTTON"*, который указывает, что мы создаем именно кнопку. При наступлении события *click* (нажатии кнопки), заработает обработчик события *onclick* и снова появится специальное окно с сообщением.

Можно изменить событие. Пусть, например, при нажатии кнопки изменится цвет фона документа с белого на красный. Измените строку кода: *<input type="BUTTON" value="Нажми меня" onclick="alert('Это сценарий на JavaScript, третья запись')">* на следующую строку: *<input type="BUTTON" value="Нажми меня" onclick="document.bgColor='red' ">*.

При нажатии на кнопку цвет фона изменится. Мы указали на объект *document* и приписали свойству объекта *bgColor* (заднему фону) красный цвет. Обратите внимание, что название цвета обрамляется в одинарные кавычки, а название *Color* обязательно должно писаться с большой буквы.

В JavaScript используются и другие обработчики событий. В табл. 9.1 указаны обработчики событий и какие события они обрабатывают.

Таблица 9.1. Типы событий и обработчиков

Событие	*Обработчик события*
Потеря фокуса ввода элементом формы	onBlur
Пользователь щелкает мышью по элементу формы	onClick
Пользователь изменяет значение элемента text, textarea, select или выделяет элемент	onChange
Установка фокуса ввода на элементе формы	onFocus
Пользователь загружает страницу в браузер	onLoad
Пользователь двигает мышью над связью или якорем	onMouseOver
Пользователь выбирает поле ввода элемента формы	onSelect
Пользователь посылает форму	onSubmit
Пользователь выходит со страницы	onUnload

Как вы уже поняли, первый и основной объект – это *document*. Через этот объект можно обращаться ко всем другим объектам на web-странице. Например, первая картинка на web-странице будет иметь имя *images[0]*, вторая *images[1]* и т. д. Обратиться, например, к первой картинке можно при помощи команды *document.images[0]*. У нас, например, на главной странице *index.php* есть одна картинка. Давайте изменим ее ширину (*width*). Откройте в PHP-редакторе нашу главную станицу и перед закрывающимся тегом *</body>* вставьте строку:

<input type="BUTTON" value="Нажми меня"
onclick="document.images[0].width=200 ">.

Сохраните и запустите скрипт. При нажатии на кнопку внизу страницы наша картинка уменьшится по ширине до 200 пикселей (пропорционально уменьшится и высота). Понятно, что мы обратились к первой картинке и задали ее ширину *width* в 200 пикселей.

После удалите эту строку из кода главной страницы.

Если у нас на странице содержится форма с именем, например *frm*, то обратиться к ней можно при помощи команды *document.frm*. Пусть у нас в форме находится текстовое поле с именем *pole* и это поле содержит некий текст, тогда, чтобы обратиться к этому тексту, нужно составить команду: *document.frm("pole").value*. Вывести же сам текст в нужном нам месте поможет команда *id.innerHTML*, где *id* – идентификатор места, куда вы хотите вывести текст, например * *. Текст выведется в блоке * *. Поясню я все это на простом примере. Удалите все в файле *java.php*. Наберите код:

```
<html>
<head>
<title> </title>
</head>
<body>
<script language="JavaScript">
function itog()
{
var txt=document.frm("pole").value;
mess.innerHTML=txt;
}
</script>
<br>
<input type="BUTTON" value="Нажми меня" onclick="alert('Это сценарий
на JavaScript')" onmouseover=itog()>
<br>
<form action="java.php" method="POST" name="frm">
<input type="TEXT" name="pole" value="Этот текст выведется в браузер">
</form>
<p id="mess"></p>
<?php
?>
</body>
</html>
```

После тега *<body>* пишем сценарий на JavaScript. Создаем некую функцию *itog()*. Эта функция присваивает переменной *txt* содержимое (*value*) текстового поля с именем *pole*, которое находится в форме с именем *frm*. Эта форма у нас создана позже между тегами *<form>* и *</form>*. При задании любой переменной в JavaScript перед ней ставится тип этой переменной. В нашем случае – это *var*. Далее эта функция выводит в браузер содержимое текстового поля там, где указан идентификатор *id=mess*.

Вот и весь сценарий, но выполнится он не сразу при запуске скрипта. Выполнится он только при вызове в коде функции *itog()*. Как видно из кода, мы создаем кнопку, при нажатии на которую (*onclick*) появится специальное окно с сообщением, а вот при событии *onmouseover* (при перемещении мыши по кнопке) вызывается функция *itog()*. Запускается первый сценарий JavaScript, где мы писали эту функцию *itog()*. В конце кода, после формы, в блоке *<p>* с идентификатором *id=mess* и выводится текст из текстового поля формы (рис. 9.10).

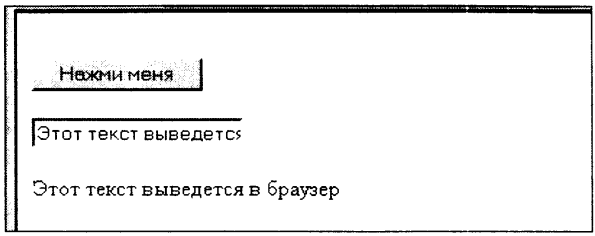


Рис. 9.10

Ну, вот и все. Существует масса литературы по JavaScript и при желании вы можете с ней ознакомиться. При помощи этого языка на web-страницах можно будет сделать много интересного. В этой книге, как я уже сказал, нам понадобится JavaScript только для калькулятора, для подсчета общей стоимости книг, выбранных посетителем.

9.4. ПРОГРАММА ДЛЯ ГЛАВНОЙ СТРАНИЦЫ МАГАЗИНА

Итак, в PHP-редакторе создайте новый файл и сохраните его в папке *magazin* под названием *magazin.php*.

Листинг 9.3 (файл magazin.php)

```
<html>
<head>
<title>Книжный магазин</title>
<link rel="stylesheet" type="text/css" href="stili2.css">
<script language="JavaScript">
function itog()
{
 var kolich=0;
 var summa=0;
 for(n=1; n<=(document.forma("kol").value); n++)
 {
  kolich=kolich+1*document.forma("kniga"+n).value;
  summa=summa+document.forma("cena"+n).value*document.forma("kniga"+n).value;
  mess.innerHTML= "Количество заказанных книг: "+kolich+"<br>"+"Общая
сумма заказа: "+summa;
 }
}
</script>
</head>
```

```
<body bgcolor="cyan">
<center><div id="lolo3">Интернет-магазин</div>
<div id="lolo">Выберите товары, которые вы хотите приобрести, указав их
количество. Укажите свое имя, e-mail, адрес.</div></center><br><br>
<form action="magazin2.php" method="POST" name="forma">
<center><table width="80%" border="2" bordercolor="green"><tr>
<?
$exten=array(".php");
$knigi="books";
$dir=opendir($knigi);
while($file=readdir($dir))
{
 if (in_array(strchr($file, "."), $exten))   $c[]=$file;
}
closedir($dir);
foreach($c as $lolo)
{
 $teg=get_meta_tags("$knigi/$lolo");
 $i++;
 $stoimost[$i]=$teg[stoimost];
 $avtor[$i]=$teg[avtor];
 $nazvanie[$i]=$teg[nazvanie];
 echo "<td>$teg[avtor]<br><a href=$knigi/book000$i.php target=_blank>
$teg[nazvanie]</a><br>($teg[stoimost] руб.)<br>
 Количество экземпляров:<br><select name='kniga$i' onchange=itog()>
 <option value=0>0</option>
 <option value=1>1</option>
 <option value=2>2</option>
 <option value=3>3</option>
 <option value=4>4</option>
 <option value=5>5</option>
 </select></td>
 <input type=HIDDEN name=cena$i value=$stoimost[$i]>
 <input type=HIDDEN name=avtor$i value='$avtor[$i]'>
 <input type=HIDDEN name=nazvanie$i value='$nazvanie[$i]'>";
 if (($i/4) == is_integer($i/4))  echo"</tr><tr>";
}
echo "<input type=HIDDEN name=kol value=$i>";
?>
```

```
</tr></table></center><br><br>
<p align="CENTER" id="mess" style="color:red; font-family:Comic Sans MS"></p><br>
Ваше имя: <input name="imja" type="TEXT"><br>
Ваш E-mail: <input name="email" type="TEXT"><br><br>
Адрес доставки: <br><textarea name="text" cols="30" rows="10" wrap="virtual"></textarea><br>
<input type="SUBMIT" value="Заказать">
</form>
</body>
</html>
```

Разберем строчки кода. При разборе кода было бы полезно периодически посматривать на рис. 9.1.

<html>

<head>

<title>Книжный магазин</title>

<link rel="stylesheet" type="text/css" href="stili2.css">

<script language="JavaScript"> – начало кода на JavaScript;

function itog() – создаем функцию *itog*, которая будет подсчитывать общую стоимость заказанных книг. Потом мы эту функцию вызовем в нужном месте PHP-кода;

1{

var kolich=0; – вводим переменную для подсчета общего количества заказанных книг, присваивая ей первоначальное значение – ноль. В *JavaScript* перед переменными знак *$* не ставится, а объявляется тип переменной (*var*);

var summa=0; – вводим переменную для подсчета общей стоимости заказанных книг, присваивая ей первоначальное значение – ноль.

for(n=1; n<=(document.forma("kol").value); n++) – это, как вы поняли, цикл со счетчиком *n*, начальное значение которого равно 1. Рассмотрим теперь второй аргумент цикла *for*. Параметр *document* в *JavaScript* указывает на текущий документ, т. е. на данную страницу с этим кодом *magazin.php*. В этом коде у нас имеется форма (см. ниже) с именем *name=forma*. В форме, как вы увидите дальше в коде, у нас будет элемент с именем *name=kol*, а параметр *value* в этом элементе будет иметь значение числа выбранных разных книг. Таким образом, второй аргумент цикла, указывающий конечное значение счетчика *n*, будет означать количество выбранных посетителем разных книг. Цикл будет выполняться, пока *n* будет пробегать значения от 1 до общего числа выбранных книг;

2{

*kolich=kolich+1*document.forma("kniga"+n).value;* – каждый раз при выполнении цикла переменной *kolich* будет добавляться количество экземпляров очередной выбранной книги (в элементах формы с именами *name=kniga1*, *name=kniga2* и т. д., параметр *value* будет указывать на количество экземпляров той или иной конкретной книги);

*summa=summa+document.forma("cena"+n).value*document.forma("kniga"+n).value;* – каждый раз при выполнении цикла переменной *summa* будет добавляться стоимость всех экземпляров очередной выбранной конкретной книги (в элементах формы с именами *name=cena1*, *name=cena2* и т. д., параметр *value* будет указывать на стоимость одного экземпляра той или иной конкретной книги). Короче говоря, в этой строке стоимость одного экземпляра умножается на число экземпляров конкретной выбранной книги, затем все это прибавляется общей сумме;

*mess.innerHTML="Количество заказанных книг: "+kolich+"
"+"Общая сумма заказа: "+summa;* – выводим в браузер общее число выбранных книг и их общую стоимость. Знак «+» в *JavaScript* является знаком соединения или конкатенации данных. Вывод в браузер в *JavaScript* осуществляется при помощи конструкции *mess.innerHTML*, где *mess* является своеобразной меткой или идентификатором для *id*. Вывод количества заказанных книг и общей их стоимости в окно браузера произойдет в том месте страницы, где будет указано: *id=mess*;

2} – окончание цикла *for*;

1} – окончание функции *itog()*;

</script> – окончание кода на *JavaScript*. Если вы не поняли приведенный выше код, то сначала разберите код ниже, а потом снова разберите этот код на *JavaScript*. Он станет вам уже более понятен;

</head>

<body bgcolor="cyan">

* Интернет-магазин *

* Выберите товары, которые вы хотите приобрести, указав их количество. Укажите свое имя, e-mail, адрес.

*

<form action="magazin2.php" method="POST" name="forma"> – выводим форму для заказа посетителем книг, количества экземпляров и для ввода его данных. После нажатия в форме кнопки «Заказать», заказ и данные посетителя будут обработаны программой *magazin2.php* (ее мы составим после) и отправлены вам на E-mail в удобочитаемом виде. Также заказ и данные посетителя будут выведены в окно браузера;

* <table width="80%" border="2" bordercolor="green"><tr>* – строим таблицу, в ячейки которой мы будем выводить автора, название книги, цену книги и выпадающий список для выбора числа экземпляров заказываемой книги;

<? – начало PHP-кода;

$exten=array(".php");* – создаем массив *$exten с одним элементом, который будет нужен для поиска файлов с расширением *php*;

$knigi="books";* – переменной *$knigi присваиваем название папки, где у нас находятся PHP-страницы с описанием товаров (*books*);

$dir=opendir($knigi); – открываем папку *books* и присваиваем ей идентификатор *$dir*;

while($file=readdir($dir)) – цикл *while* перебирает все файлы в папке *books*, каждый раз выполняя команду в фигурных скобках под номером 3;

3{

if (in_array(strchr($file, "."), $exten)) $c[]=$file; – в переменной *$file* при каждом выполнении цикла у нас будет очередное название файла в папке *books*. Если помните, функция *strchr* (см. гл. 4) возвращает часть строки, начиная с символа, указанного в качестве второго аргумента этой функции, т. е. в нашем случае она возвратит только расширение файла. Далее функция *in_array* сравнивает полученное расширение с содержимым массива *$exten* (а там у нас элемент *.php*). Если полученное расширение файла совпадает хотя бы с одним элементом массива *$exten*, то функция *in_array* возвратит *true*. Итак, если очередной файл, выбранный в цикле *while*, имеет расширение *php* (а страницы для описания того или иного товара будут именно с такими расширениями), то название этого файла заносится в новый массив *$c*. Таким образом, мы как бы отсекаем в папке *books* все остальные папки и файлы, не имеющие расширение *php*;

3}

closedir($dir); – закрываем папку *books*;

foreach($c as $lolo) – снова цикл, который будет выполняться столько раз, сколько файлов в массиве *$c*, присваивая очередной элемент массива *$c* переменной *$lolo*. Все тело цикла заключено в фигурные скобки под номером 4. Для наибольшего понимания кода цикла, рассмотрим его первое выполнение, когда переменной *$lolo* присваивается первый (точнее, нулевой) элемент массива *$c*. Этим элементом будет файл *book0001.php*, который уже у нас имеется в папке *books*;

4{

$teg=get_meta_tags("$knigi/$lolo"); – при помощи функции *get_meta_tags* (данная функция была рассмотрена в гл. 6) получаем массив *$teg*, в котором

будет содержание мета-тегов на странице *book0001.php*. Массив будет содержать 3 элемента (для понимания можете посмотреть код страницы *book0001.php* в листинге 9.2а): *$teg[avtor]* (автор книги который указан в параметре *content="М. Булгаков"*), *$teg[nazvanie]* (название книги, тоже указано в параметре *content="Роковые яйца"*) и *$teg[stoimost]* (цена книги в параметре *content="100"*);

$i++; – создаем счетчик для пробегания всех PHP-страниц в папке *books*. При первом выполнении цикла, а мы пока рассматриваем только первое выполнение, переменная *$i* будет равна единице (*$i=1*);

$stoimost[$i]=$teg[stoimost];* – создаем массив *$stoimost, первому элементу которого (*$stoimost[1]*) присваивается стоимость первой книги;

$avtor[$i]=$teg[avtor];* – аналогично создаем массив *$avtor, первому элементу которого (*$avtor [1]*) присваивается фамилия автора первой книги;

$nazvanie[$i]=$teg[nazvanie];* – аналогично создаем массив *$nazvanie, первому элементу которого (*$nazvanie [1]*) присваивается название первой книги;

*echo "<td>$teg[avtor]
 $teg[nazvanie]
($teg[stoimost] руб.)
* – в первую ячейку таблицы выводим фамилию автора книги, название книги (оно будет являться также ссылкой на страницу *book0001.php*) и цену книги;

*Количество экземпляров:
<select name='kniga$i' onchange=itog()>* – в первой же ячейке таблицы создаем раскрывающийся список с именем *name='kniga1'* (при первом выполнении цикла *foreach*, когда *$i=1*). Список будет состоять из 6 пунктов (тег <option>) с цифрами от 0 до 5 (*value*) для выбора посетителем количества экземпляров первой книги. Цифра «0» по умолчанию будет означать, что данная книга не выбрана. Как только посетитель выберет из списка какую-нибудь другую цифру, произойдет событие *onchange* (выбор) и вызовется функция *itog()*, которая начнет считать общую стоимость всех экземпляров первой книги. Эта функция была написана в начале этого кода на языке *JavaScript*. Вернитесь туда и посмотрите на строчку:

*kolich=kolich+1*document.forma("kniga"+n).value*

Теперь, надеюсь, она вам понятна. Переменной *kolich* (не забываем, только при первом выполнении цикла *foreach*) присваивается параметр *value* элемента формы с именем *name='kniga1'* (знак «+» в *JavaScript* обозначает конкатенацию или соединение данных). Параметр *n* для первой книги будет равен 1. А параметр *value* – это количество экземпляров, которое выбрал посетитель для первой книги (имя *name='kniga1'* у нас имеет элемент формы – раскрывающийся список для первой книги). Элементы формы с именами *name=cena$i* и *name=kol$i* будут переданы для *JavaScript* чуть позже;

<option value=0>0</option>

<option value=1>1</option>

<option value=2>2</option>

<option value=3>3</option>

<option value=4>4</option>

<option value=5>5</option>

</select></td>

<input type=HIDDEN name=cena$i value=$stoimost[$i]> – создаем скрытый элемент формы с именем (при первом выполнении цикла *foreach*) *name=cena1* и *value=$stoimost[1]*. Это нужно для функции *itog()* кода *JavaScript*. Данные этого элемента формы (как и раскрывающегося списка) будут нужны для подсчета общей стоимости выбранных книг;

<input type=HIDDEN name=avtor$i value='$avtor[$i]'> – создаем скрытое поле для передачи фамилии автора первой книги странице *magazin2.php*, которую мы создадим позже. Фамилия автора первой книги у нас в переменной *$avtor[1]*;

<input type=HIDDEN name=nazvanie$i value='$nazvanie[$i]'>"; – аналогично создаем скрытое поле для передачи названия первой книги странице *magazin2.php*;

if (($i/4) == is_integer($i/4)) echo"</tr><tr>"; – как видно из рис. 9.1, в строке таблицы находится по 4 ячейки. Пятая книга будет уже на новой строке. Короче говоря, если число книг кратно 4, то создаем новую строку для таблицы. Математическая функция *is_integer*, если помните, возвращает целую часть числа, отбрасывая дробную. Выражение в круглых скобках оператора *if* будет истинным только при *$i*=4, 8,12 и т. д.;

4} – окончание тела цикла *foreach*. Далее после первого выполнения этого цикла в папке *books* будет выбран второй файл *book0002.php* для второй книги (если, конечно, данные для второй книги были переданы на сервер при помощи панели администратора *admin.php*), и цикл повторится вновь, но со значением переменной *$i*, равным двум. Также будет создан раскрывающийся список с выбором количества экземпляров для второй книги. Элементы массивов *$avtor[2]*, *$nazvanie[2]*, *$stoimost[2]* будут содержать соответственно фамилию автора, названия и цену для второй книги. Функция *itog()* в коде на *JavaScript* высчитает стоимость всех экземпляров второй книги и прибавит эту сумму к стоимости всех экземпляров первой книги (если, конечно, выбранное число экземпляров первой и второй книги отлично от нуля). Как только функция *foreach* переберет все книги магазина, т. е. все php-страницы в папке *books*, цикл между фигурными скобками под номером 4 прекратит выполняться;

echo "<input type=HIDDEN name=kol value=$i>";* – после окончания выполнения прошлого цикла в переменной *$i будет общее число различных книг в магазине. Это число мы и передадим потом странице *magazin2.php* при помощи скрытого поля;

?> – окончание PHP-кода;

</tr></table>

 – заканчиваем таблицу;

*
* – выводим в этом месте страницы, в контейнере * *, общее количество заказанных книг и их общую стоимость, красным шрифтом типа *Comic Sans MS*. Все это посчитала нам функция *itog()*, которую мы написали в начале кода на *JavaScript*. Там показано, что вывод данных должен быть в HTML-теге с идентификатором *id=mess* (*mess.innerHTML=...*). Этот идентификатор мы и присвоили текстовому блоку (или контейнеру) * *.

*Ваше имя: <input name="imja" type="TEXT">
* – создаем текстовое поле для ввода имени посетителем, заказавшего книги;

*Ваш E-mail: <input name="email" type="TEXT">

* – создаем текстовое поле для ввода посетителем своего электронного адреса;

*Адрес доставки:
<textarea name="text" cols="30" rows="10" wrap="virtual"></textarea>
* – создаем текстовое поле для ввода посетителем своего домашнего адреса. Параметр *wrap* задает автоматический перенос строк в текстовом поле;

<input type="SUBMIT" value="Заказать"> – после того, как посетитель магазина выбрал книги, в текстовых полях записал свои данные и нажал на кнопку «Заказать», заказ и данные посетителя передадутся для обработки на страницу *magazin2.php*. Сам посетитель тоже переедет на ту же страницу;

</form> – завершаем вывод формы;

</body>

</html> – окончание кода.

9.5. ОБРАБОТКА ЗАКАЗА ОТ ПОСЕТИТЕЛЯ МАГАЗИНА

Теперь осталось нам написать код для страницы *magazin2.php*. Этот код, как я уже отмечал, обрабатывает заказ и данные от посетителя, выводит их в окно браузера (см. рис. 9.3) и отправляет их вам по указанному вами электронному адресу.

В PHP-редакторе создайте новый файл и сохраните его в папке *magazin* под названием *magazin2.php*. Наберите код, как в листинге 9.4.

Листинг 9.4 (файл magazin2.php)

```
<html>
<head>
 <title>Запрос</title>
</head>
<body>
<?php
$n=$_POST['kol'];
$imja=strip_tags($_POST['imja']);
$email=strip_tags($_POST['email']);
$text=strip_tags($_POST['text']);
$zakaz="";
for ($i=1; $i<=$n; $i++)
{
 if($_POST['kniga'.$i]>0)
 {
  $zakaz="$zakaz". $_POST['avtor'.$i]." (".$_POST['nazvanie'.$i].")-
".$_POST['kniga'.$i]." шт<br>";
 }
}
$zakaz="Поступил заказ: $zakaz<br> от клиента: $imja <br> Электронный адрес
клиента: $email <br>
Адрес и контактные данные клиента: $text";
echo "$zakaz<br>";
echo "<div id=lolo3>Заявка отправлена. Благодарим за заказ!</div>";
mail("interstrog@narod.ru", "Заказ на товар", $zakaz, "From: $email\nReply-To:
$email\nContent-Type: text/plain; charset=windows-1251");
?>
</body>
</html>
```

Разберем некоторые строчки кода:

<?php

$n=$_POST['kol']; – все данные из формы, переданные из программы *magazin.php*, будут храниться в суперглобальном массиве *$_POST*. Переменной *$n* присваиваем общее число различных книг, т. е. общее число товаров в магазине. Помните, в прошлой программе *magazin.php* мы создавали скрытое поле с именем *name=kol*. В этом поле мы и передали на эту страницу (*magazin2.php*) общее число товаров в магазине;

$imja=strip_tags($_POST['imja']); – аналогично переменной *$imja* присваиваем введенное посетителем имя в текстовом поле формы в программе *magazin.php*. Имя обрабатываем функцией *strip_tags*, которая удаляет все теги, если они были введены посетителем. Это, как вы уже догадались, делается для безопасности, чтобы какой-нибудь злоумышленник не ввел вместо имени вредоносный код;

$email=strip_tags($_POST['email']); – переменной *$email* присваиваем введенный посетителем E-mail;

$text=strip_tags($_POST['text']); – переменной *$text* присваиваем введенный посетителем домашний адрес;

$zakaz=""; – в эту переменную мы будем записывать строку с заказом и данными посетителя (покупателя книг). Пока она у нас пуста;

for ($i=1; $i<=$n; $i++) – создаем цикл со счетчиком *$i*, принимающий значения от 1 до *$n*, где *$n* – общее число товаров (различных книг) в магазине. Для лучшего понимания рассмотрим работу этого цикла при *$i=1*;

1{

if($_POST['kniga'.$i]>0) – в прошлом листинге (9.3) именем *name='kniga1'* мы назвали элемент формы – раскрывающийся список для первой книги. Элемент суперглобального массива *$_POST['kniga1']* будет содержать значение параметра *value* раскрывающегося списка. Если количество экземпляров первой книги посетитель оставит равным нулю (*value=0*), то будет считаться, что эта книга просто не была им выбрана, и код в фигурных скобках под номером 2 не выполнится (действительно, зачем нам указывать, что та или иная книга была заказана в количестве ноль экземпляров). Если первая книга была выбрана (мы пока рассматриваем исполнение условия *if* при *$i=1*), т. е. параметр *value>0*, то код между фигурными скобками под номером 2 выполнится;

2{

*$zakaz="$zakaz". $_POST['avtor'.$i]." (".$_POST['nazvanie'.$i].")-".$_POST['kniga'.$i]." шт
";* – формируем строку, куда запишем автора выбранной первой книги, фамилия которого содержится в элементе массива *$_POST['avtor1']* (именем *name=avtor1* мы назвали скрытое поле в листинге 9.3 и передали с его помощью на эту страницу значение *value='$avtor[1]'*), название книги (*$_POST['nazvanie1']*) и количество выбранных экземпляров первой книги (*$_POST['kniga1']*). Все это мы записываем в одну строку, соединив данные о книге точкой (метод конкатенации). Круглые скобки и дефис обязательно заключаем в двойные кавычки;

2}

1} – цикл *for* при *$i=1* (для первой книги) выполнен. Далее этот цикл снова начнет выполняться для второй книги (*$i=2*), если, конечно, она была выбрана (количество экземпляров не равно нулю) и т. д. После того, как будут перебраны все книги магазина, цикл полностью завершится, а в переменной *$zakaz* будет несколько строк с данными о всех выбранных книгах;

*$zakaz="Поступил заказ: $zakaz
 от клиента: $imja
 Электронный адрес клиента: $email
*

Адрес и контактные данные клиента: $text";* – добавляем в переменную *$zakaz имя посетителя, которое он указал в форме *magazin.php* (листинг 9.3), его электронный адрес и домашний адрес;

*echo "$zakaz
";* – выводим все содержимое переменной *$zakaz* в браузер;

echo " Заявка отправлена. Благодарим за заказ! "; – выводим в браузер строку с благодарностью за заказ;

mail("interstrog@narod.ru", "Заказ на товар", $zakaz, "From: $email\nReply-To: $email\nContent-Type: text/plain; charset=windows-1251"); – функция mail отправит данные заказа и данные посетителя на ваш E-mail (эта функция разобрана в разд. 9.1). Вместо *interstrog@narod.ru* подставите свой электронный адрес.

Протестируем наш магазин. Для начала в редакторе запустите программу *admin.php* (листинг 9.2 и рис. 9.4). Закачайте несколько книг и каких-нибудь фотографий к ним, как мы делали после разбора листинга 9.2. Теперь попробуйте заказать какие-нибудь книги. Запустите в PHP-редакторе программу *magazin.php* (листинг 9.3 и рис. 9.1). Если все программы вы написали правильно, то в магазине вы увидите названия и авторов всех введенных вами книг. Выбсрите один или несколько экземпляров любых книг. Заполните поля для имени и адреса посетителя и нажмите кнопку «*Заказать*». Вы попадете на страницу *magazin2.php*, где увидите текст, примерно как на рис. 9.3. Если это так, то вы все сделали правильно. Внизу вы еще увидите строку ошибки, примерно следующего содержания:

Warning: mail()[function mail]: Failed to connect... и т. д. Функция *mail* не сработала, и это понятно, мы же тестируем программу на локальном компьютере с локальным сервером. Эта функция будет работать только на сервере вашего хостинг-провайдера, где вы будете размещать свой сайт.

Ну вот, простой электронный магазин готов. Вам приходит по электронной почте заказ, и вы отправляете его наложным платежом заказчику. Все просто, но можно сделать еще проще. Например, продавать книги в электронном виде, которые заказчик может скачать себе на компьютер после оплаты. У вас, например, есть электронный счет (об этом в гл. 10). На вашем

сайте есть электронные товары (книги, музыка, программы и т. п.) и вы хотите, чтобы посетитель мог скачать какой-либо электронный товар только тогда, когда положит на ваш электронный счет определенную сумму денег. Все это вполне реализуемо, причем купля-продажа будет делаться без вашего участия. Как только покупатель перечислит деньги на ваш счет, он автоматически переходит на страницу для скачивания электронного товара. В гл. 10 я объясню, как организовать торговлю электронными товарами на вашем сайте.

Глава 10. ЭЛЕКТРОННЫЕ ДЕНЬГИ. ОПЛАТА ТОВАРОВ С ВАШЕГО САЙТА ЭЛЕКТРОННЫМИ ДЕНЬГАМИ

10.1. ОБ ЭЛЕКТРОННЫХ ДЕНЬГАХ

Электронные деньги – в широком смысле – форма организации денежного обращения в ассоциации информационных сетей.

Электронные деньги предназначены для осуществления мгновенных платежей в Интернете. За электронные деньги можно купить:

- текстовую информацию (файлы);
- музыку (файлы);
- видео (файлы);
- оплатить за услуги (мобильный телефон, доступ в Интернет);
- оплатить за товары в Интернет-магазинах;
- использовать их в Интернет-казино и Интернет-тотализаторах и т. д.

Электронные деньги – это вид валюты. В отличие от евро или долларов, пощупать их невозможно, но эти деньги существуют, и существуют в виде цифр у вас на счету. Как и любую другую валюту, электронные деньги можно обменять на рубли, доллары, евро и иные привычные для нас деньги. Электронных денег бывает несколько видов. Точно так же, как существуют доллары, евро, существует множество различных электронных денег. Например: *Яндекс-деньги, WebMoney, E-Gold* и т. д. Некоторые электронная деньги, в свою очередь, еще подразделятся на «подденьги». Например, на *WebMoney* у вас может быть 3 разных кошелька – долларовый, евро, рубли. На Яндекс-деньгах – только рубли. Как и любую валюту, электронные деньги можно обменивать между собой. Так, можно менять Яндекс-деньги на ВебМани (*WebMoney*). Платежная система – это механизм, с помощью которого вы можете пересылать деньги из одного места в другое, другими словами – это механизм транспортировки денег. С помощью платежных систем вы можете также пополнять свои электронные деньги. Электронные деньги можно обменивать друг на друга. Это можно сделать на специальных сайтах, которые называются пунктами обмена электронных денег. Например, у вас есть Яндекс-деньги, а вам нужны *WebMoney*. Вы идете на сайт обменного пункта и меняете одни деньги на другие.

10.2. YANDEX MONEY

Яндекс-деньги – универсальная платежная система. Она позволяет безопасно и быстро оплачивать товары и услуги в Интернете.

Яндекс.Кошелек – это кошелек, доступ к которому осуществляется через сайт Яндекс.Денег. Им можно пользоваться с любого компьютера. Интернет.Кошелек – это программа, которая устанавливается на ваш компьютер. Можно установить копии Интернет.Кошелька на несколько компьютеров.

Создайте себе электронный кошелек на *Yandex*. Он нам понадобится для тестирования работоспособности нашего электронного магазина, где мы разместим электронные товары для продажи. Создать электронный счет очень просто. Зайдите на сайт *http://www.yandex.ru* и выберите из меню поиска пункт «*Все службы...*» (рис. 10.1).

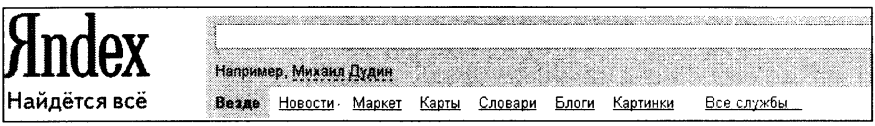


Рис. 10.1

Вы попадете на страницу, где будут представлены все службы Яндекса (рис. 10.2).

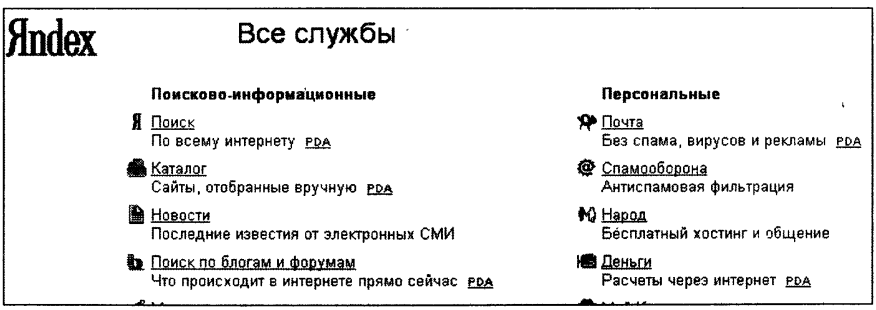


Рис. 10.2

В колонке «*Персональные*» выберите пункт «*Деньги*». Попадете на страницу, как на рис. 10.3.

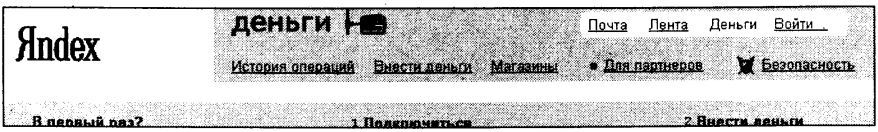


Рис. 10.3

Вверху справа нажмите на ссылку «Войти». Вы попадете на страницу, где можно зарегистрироваться. Логина и пароля у вас пока нет, поэтому жмите на ссылку «Зарегистрироваться» (рис. 10.4).

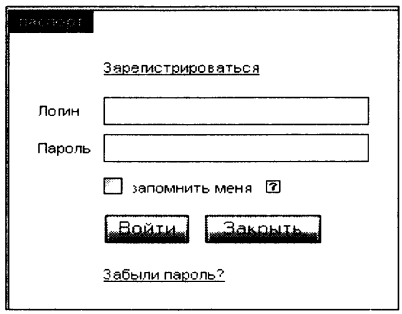


Рис. 10.4

Далее последует первый шаг мастера регистрации (рис. 10.5).

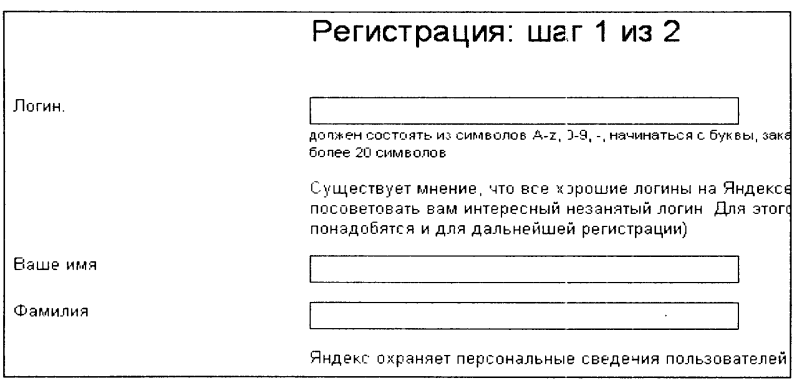


Рис.10.5

Там нужно будет придумать логин (запомните его или запишите!), указать свои имя и фамилию. Указывайте реальные данные, чтобы не было проблем с выводом денег. Если указанный вами логин уже занят в Интернете, мастер предложит вам выбрать другой логин.

Во втором шаге вам нужно будет придумать пароль (запомните его или запишите!), указать мобильный или другой телефон, электронный адрес (если есть).

После завершения регистрации у вас появится электронный адрес (рис. 10.6).

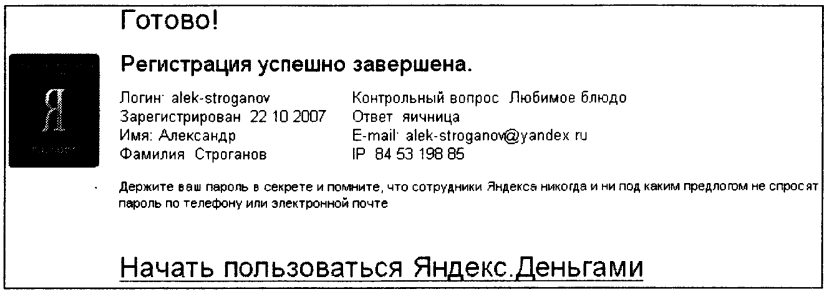


Рис. 10.6

Нажмите на ссылку «*Начать пользоваться Яндекс.Деньгами*». Вы попадете на свою почту, в раздел «*Деньги*» (рис. 10.7).

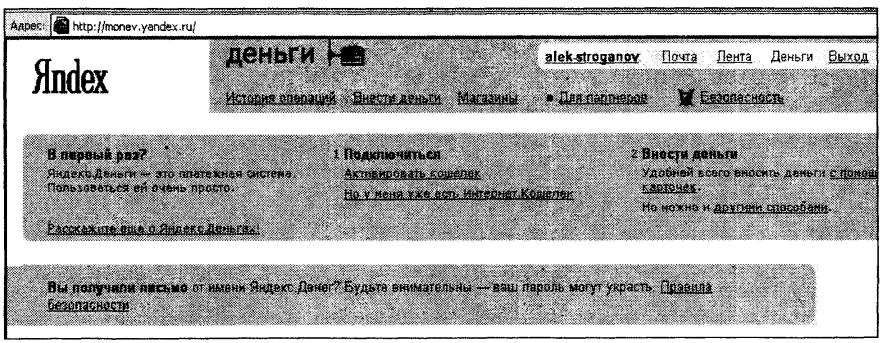


Рис. 10.7

Здесь, если захотите, можете узнать подробнее о Яндекс.Деньгах. Далее нажмите на ссылку «*Активировать кошелек*». Вы должны завести себе платежный пароль (запомните или запишите его!), чтобы вы потом могли выполнять различные операции с денежными средствами (рис. 10.8).

Заполните данные, в том числе и паспортные, если вы в будущем захотите вывести деньги из системы и получить их наличными. После заполнения всех данных нажмите на кнопку «*Сохранить*». Вы попадете опять на свою почту, в раздел «*Деньги*» (рис. 10.9).

Все, кошелек создан. Номер вашего электронного счета будет состоять из 14 цифр. Пока на счету у вас нуль. Пополнить кошелек можно разными способами: почтовым или банковским переводами. Более подробно об этом можно узнать, нажав на ссылку «*Положить в кошелек*». Счет можно пополнить и при помощи карточки Яндекс.Деньги, нажав на ссылку «*Активировать карту*». Карточки Яндекс.Деньги можно приобрести, например, в салонах мобильной связи.

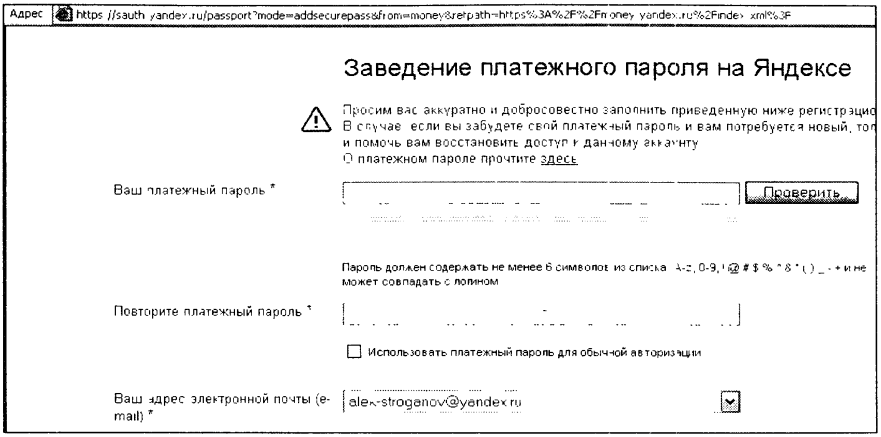


Рис. 10.8

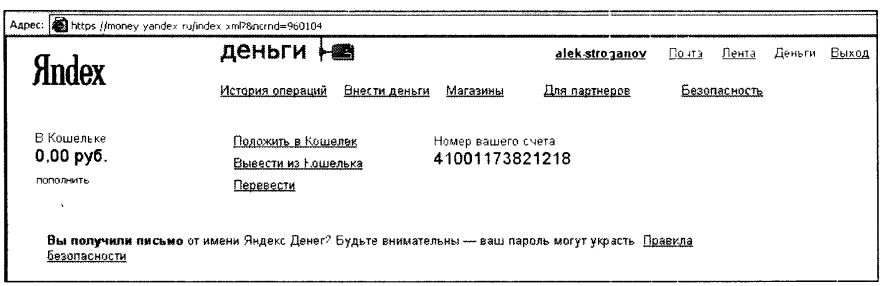


Рис. 10.9

Теперь у вас на *Yandex* есть электронный адрес и кошелек. Заходя на Яндекс по адресу *www.yandex.ru*, вы можете теперь зайти на свою электронную почту, нажав вверху на ссылку «*Войти в почту*» (рис. 10.10).

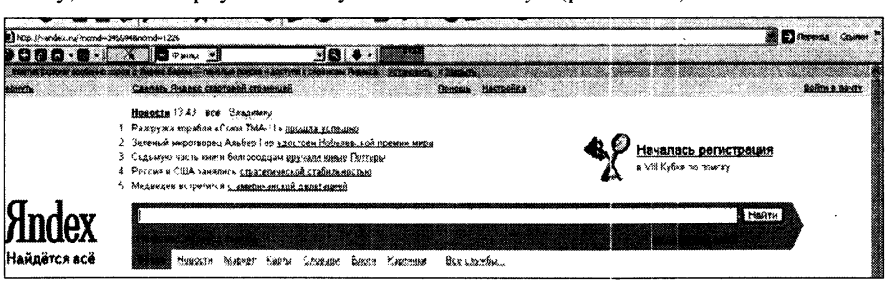


Рис. 10.10

После ввода своего логина и пароля, вы войдете в свой электронный почтовый ящик.

10.3. WEBMONEY

Кроме Яндекс.Деньги, есть в Интернете еще одна популярная платежная система. *WebMoney* очень удобна для совершения денежных переводов через Интернет. Главное преимущество заключается, пожалуй, в их мгновенности: вы нажимаете кнопку – и через пару секунд адресат получает средства на свой электронный кошелек, где бы он ни находился. Вообще в использовании *WebMoney* много приятных моментов:

- вы получаете и переводите деньги не выходя из дома;
- расстояния не имеют значения. Даже если плательщик и получатель находятся в разных точках земного шара, никаких проблем не возникнет;
- операции происходят в считаные секунды;
- как следствие из первых трех пунктов, электронные деньги *WebMoney* особенно удобны для оплаты товаров, которые могут быть доставлены по каналам Интернета, сразу же после их оплаты, – PIN-кодов, электронных книг, программного обеспечения, музыки и т. д.;
- щадящая комиссия за переводы: система берет себе 0,8 %;
- операции безотзывны, а это большое преимущество для продавцов, которые хотели бы принимать оплату за товары и услуги через Интернет и не бояться при этом, что платеж может быть отозван.

Для того чтобы вести расчеты в системе *WebMoney*, нужно иметь хотя бы один кошелек. Кошелек *WebMoney* напоминает обычный кошелек в реальной жизни – в нем тоже хранятся деньги. Только, конечно, не бумажные, а те самые, электронные.

Создадим себе счет в *WebMoney*. На этот счет мы тоже будем принимать оплату за предлагаемые на нашем сайте электронные товары.

Регистрация в системе. Для того чтобы стать участником *WebMoney Transfer*, необходимо сначала в ней зарегистрироваться и завести *кошелек*. Для этого нужно установить на свой компьютер специальную программу, с помощью которой и будут осуществляться все операции. Эта программа называется *WebMoney Keeper*. Ее можно бесплатно скачать с сайта *WebMoney*.

Для регистрации проделайте вместе со мной следующие действия:

1. Загрузите последнюю версию *WebMoney Keeper* по ссылке *http://download.webmoney.ru/wm2.EXE*. Размер дистрибутива – около 5 Мб.
2. После загрузки запустите скачанный файл, чтобы установить программу. Процедура инсталляции обычна и отличается только тем, что нужно подтвердить согласие с несколькими соглашениями, принятыми в системе. Просто следуйте появляющимся на экране инструкциям. Keeper также

попросит вас установить специальные сертификаты. Ответьте «*Да*», потому что эти сертификаты необходимы для нормальной работы программы. Сразу после инсталляции с программой можно работать.

3. Подключитесь к Сети. Запустите установленый Keeper. В появившемся окошке выберите «*Зарегистрироваться в WebMoney*» (рис. 10.11), нажмите ОК.

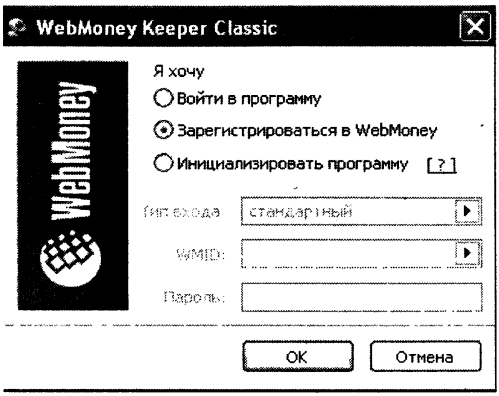


Рис. 10.11

Откроется окно Мастера регистрации (рис. 10.12). Там вам предложат ввести регистрационный код. Если вы регистрируетесь впервые, регистрационного кода, естественно, у вас пока нет.

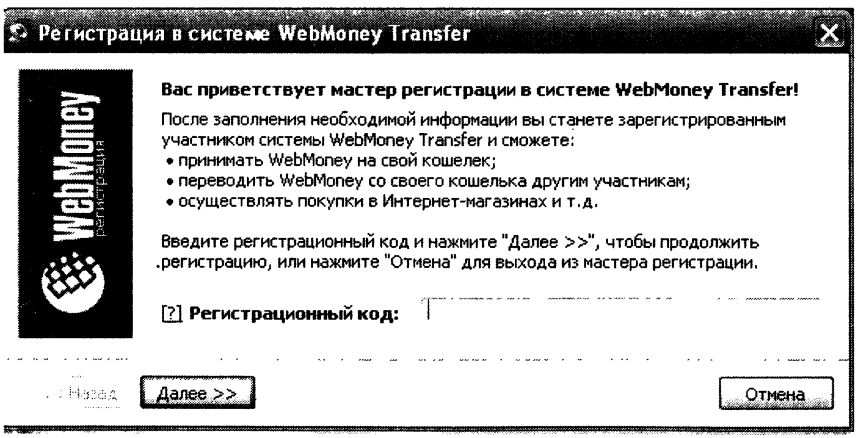


Рис. 10.12

- Просто нажмите на кнопку «*Далее*». Появится окно (рис. 10.13), где снова вам предложат ввести регистрационный код.

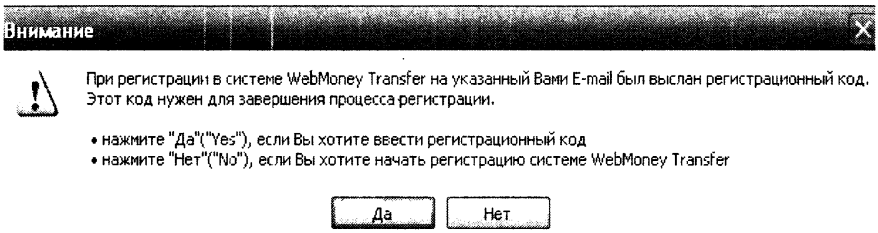


Рис. 10.13

Жмите на кнопку «Нет», чтобы начать новую регистрацию.

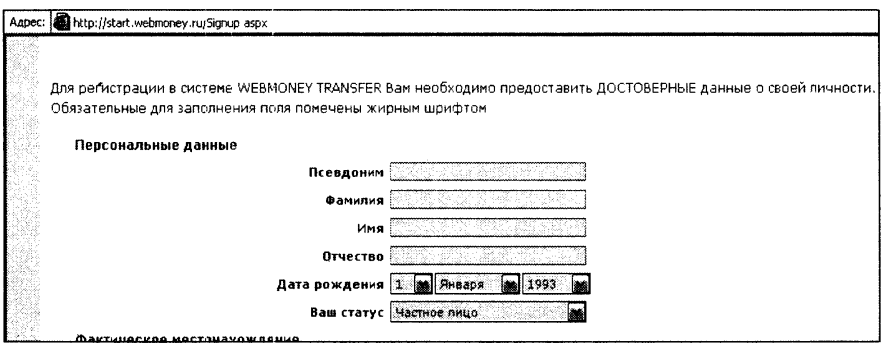


Рис. 10.14

Вы попадете на сайт *http://start.webmoney.ru/Signup.aspx* (рис. 10.14), где вам предложат ввести свои данные. Вводите достоверные данные, иначе вам могут отказать в регистрации! Правильно также укажите свой E-mail, поскольку именно на этот электронный адрес будет выслан регистрационный код. E-mail можете указать тот, который вы получили на *Yandex*, когда заводили там себе Yandex-кошелек.

Итак, после ввода и отправки всех данных вы попадете на следующую страницу регистрации (рис. 10.15).

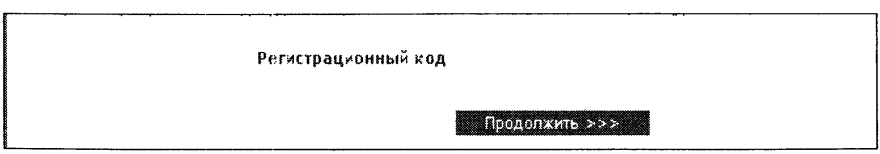


Рис. 10.15

Введите код, который был выслан на указанный вами E-mail и нажмите на кнопку «*Продолжить*». Вы попадете на следующую страницу регистрации (рис. 10.16).

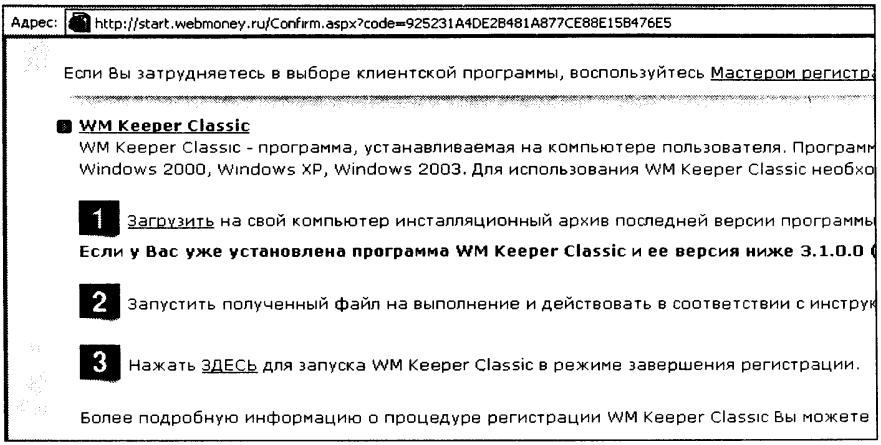


Рис. 10.16

В п. 3 нажмите на ссылку «*ЗДЕСЬ*» для перехода к завершающему этапу регистрации.

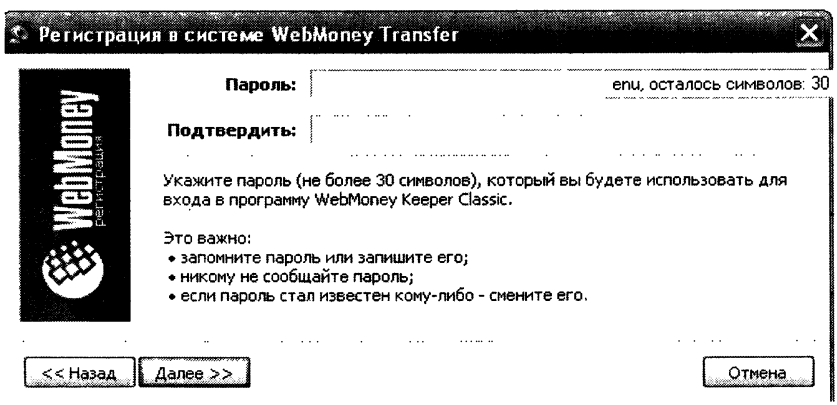


Рис. 10.17

Откроется форма Мастера регистрации (рис. 10.17). Придумайте и введите пароль, подтвердите его и нажмите на кнопку «*Далее*».

На следующем шаге Мастер предложит понажимать случайным образом кнопки или подвигать мышь – таким способом будет генерироваться файл ключей, содержащий вашу цифровую подпись в системе.

Затем Мастер будет создавать этот файл (рис. 10.18). Данный процесс может занять несколько минут. Чтобы скоротать время, займитесь прочтением сообщений-анонсов, которые программа будет показывать в том же окне.

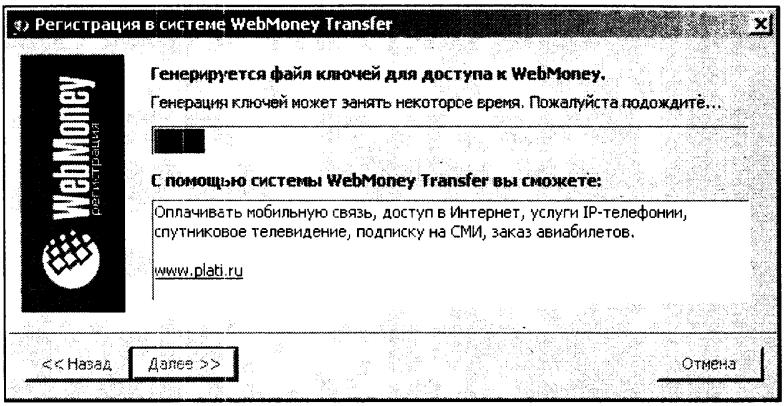


Рис. 10.18

После того, как этот процесс завершится, вам будет присвоен ваш персональный WM-идентификатор (WMID) (рис. 10.19). WMID – это уникальный номер пользователя в системе. Он состоит из 12 цифр и служит для входа в Keeper, а также выполняет некоторые другие, в основном «представительские» функции. Например, по этому номеру можно посмотреть информацию о любом пользователе системы или связаться с ним по внутренней почте.

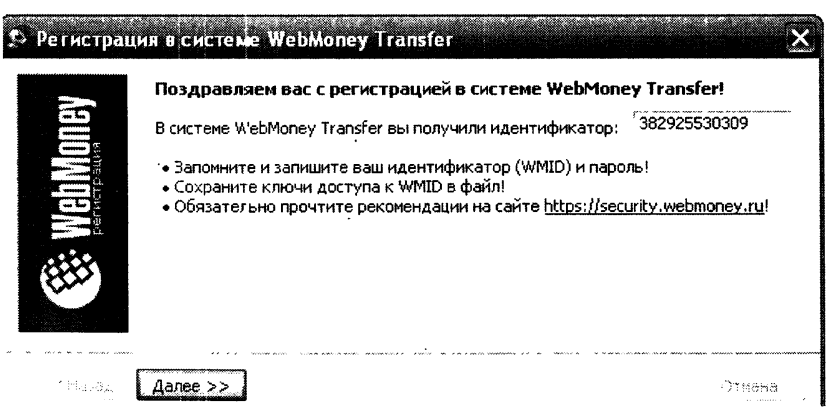


Рис. 10.19

Обязательно запомните и запишите идентификатор и пароль! Нажмите кнопку «*Далее*».

Теперь наступает ответственный момент. Нужно указать путь для сохранения файла ключей и код доступа (рис. 10.20).

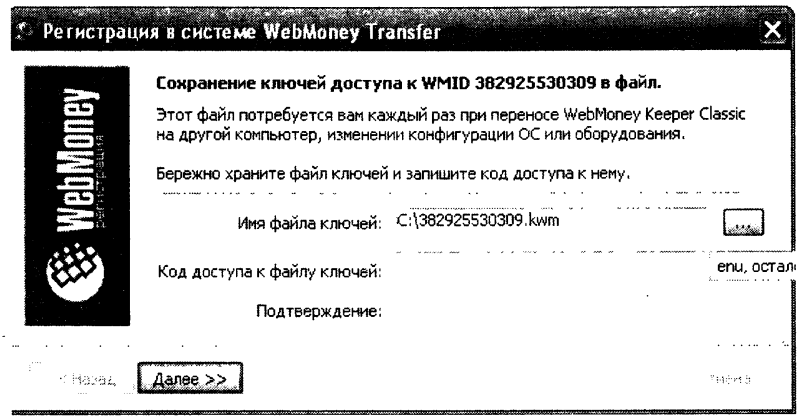


Рис. 10.20

Здесь немного теории. Файл ключей имеет расширение *.kwm* и в ежедневной работе вам не понадобится. Он служит для того, чтобы иметь возможность подключиться к WebMoney-счету с другого компьютера или после переустановки операционной системы. Зачем так устроено? Дело в том, что *WebMoney* старается максимально защитить своих пользователей от мошенников и хакеров. Если у вас, например, похитят пароль к WMID, то украсть деньги злоумышленники все равно не смогут: им нужен будет этот самый файл .kwm, чтобы войти под вашим WMID со своего компьютера. В то же время вы как владелец файла можете подключаться с его помощью к своему счету когда угодно и откуда угодно. Этим *WebMoney* принципиально отличается от других электронных систем, где для доступа к счету нужен только логин и пароль.

Теперь вы понимаете, что .kwm желательно хранить в таком месте, где он будет недоступен для посторонних, например на сменном носителе или зашифрованном диске. Впрочем, даже в случае кражи файла мошенникам нужно будет знать код доступа, которым этот файл защищен. Код доступа вы задаете в том же окне при регистрации. Следует различать код доступа и пароль на вход. Первый защищает файл ключей, второй служит для подключения к вашему счету.

В следующем окошке вам будет предложено получить персональный аттестат.

Что это такое, я разъясню позже, а пока просто нажмите кнопку «*Готово*», предварительно убрав галочку в переключателе для получения персонального аттестата.

На указанный ранее E-mail высылается письмо со специальным кодом активации, который нужно ввести на этом, последнем этапе (рис. 10.21). Без этого вы не сможете окончить регистрацию и начать пользоваться Keeper'ом. Обратите внимание, что некоторые бесплатные почтовые службы могут автоматически удалять письма от почтового робота *WebMoney*, приняв их за спам. Поэтому если вы так и не получили письмо с кодом, попробуйте пройти весь процесс регистрации снова, но теперь укажите какой-нибудь другой почтовый ящик. например свой e-mail у Интернет-провайдера. Если это не поможет – напишите письмо на *support@wmtransfer.com* с указанием своего WMID, и тогда код активации будет выслан вам техподдержкой вручную.

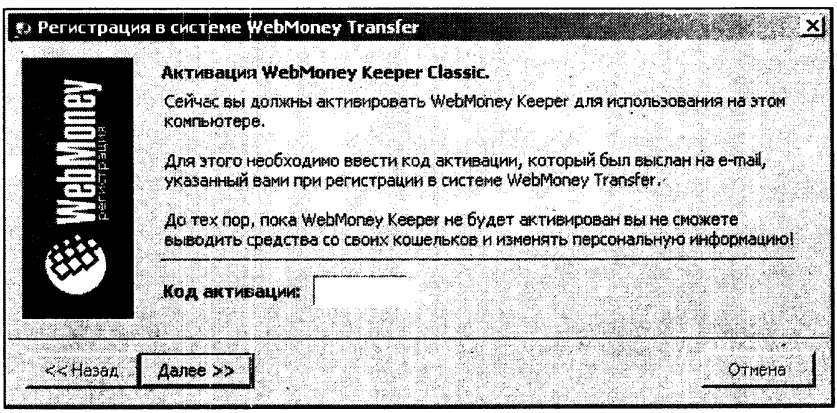


Рис. 10.21

После введения кода регистрация завершена. Сразу же вы получаете доступ к Keeper'у. Запустится ваш Keeper (рис. 10.22), и появится значок внизу справа в виде желтого муравья.

Как работать с Keeper и какие функции он выполняет, можете узнать в меню «*Справка*». Нам нужно пока только создать кошелек *WebMoney* для приема платежей с сайта. Нажмите на вкладку «*Кошельки*». Кошельков у вас пока нет (рис. 10.23).

Нажмите на вкладку «*Создать*» с изображением желтого кошелька.

Откроется форма для создания кошелька (рис. 10.24). Нажмите кнопку «*Далее*». В следующем шаге вы должны выбрать тип денежной валюты, которая будет у вас в кошельке. Выберите из списка пункт «*WMR-эквивалент RUB*», так как принимать платежи с сайта мы будем в рублях. Далее, в сле-

дующем шаге, вы должны принять условие соглашения об использовании чеков в электронной форме. На следующем шаге, наконец, рублевый кошелек будет создан (рис. 10.25).

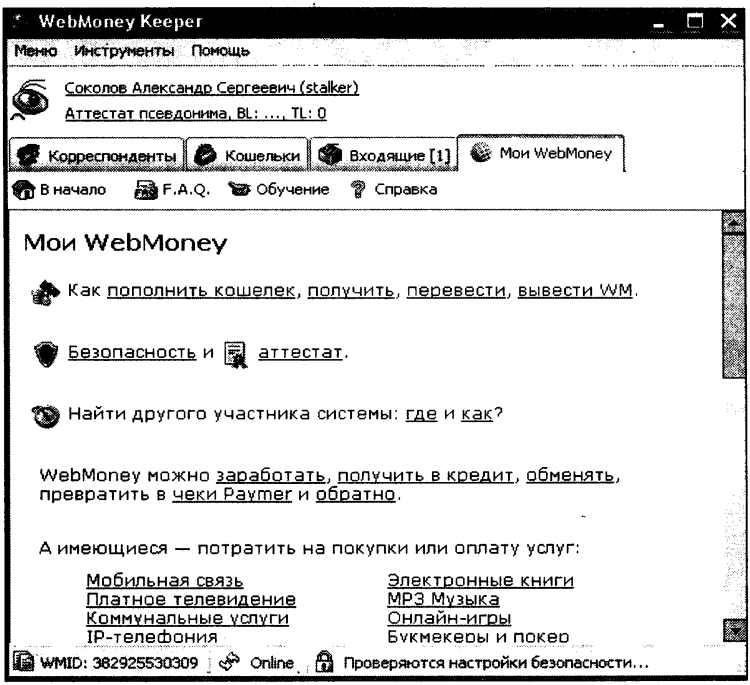


Рис. 10.22

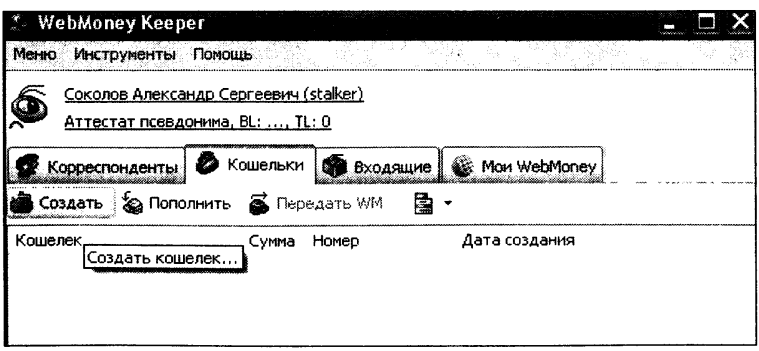


Рис. 10.23

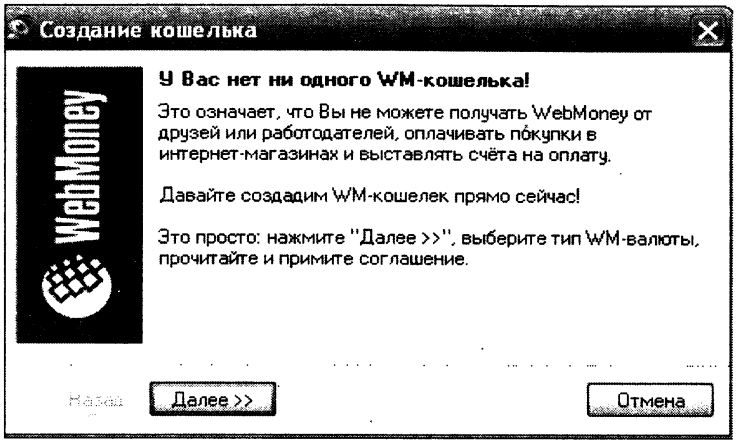


Рис. 10.24

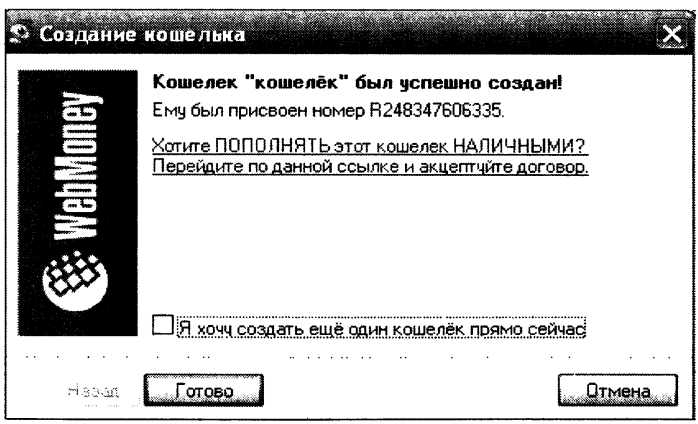


Рис. 10.25

Можно, конечно, нажать на кнопку «*Готово*» и закончить создавать кошельки. Для учебных целей этого будет достаточно, но если вы потом захотите все же пополнять созданный кошелек реальными деньгами, вам нужно будет получить хотя бы формальный аттестат. В отличие от персонального аттестата, формальный аттестат можно получить бесплатно. Для этого нажмите мышью на ссылку для заключения договора.

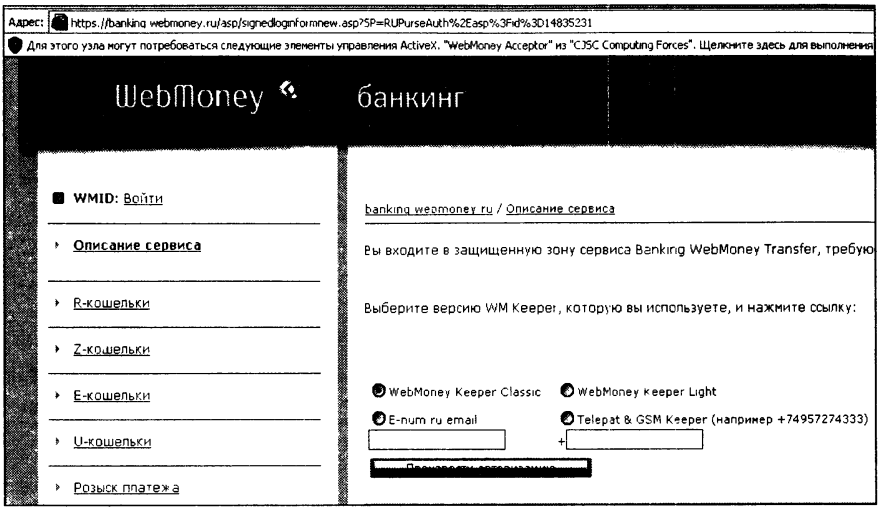


Рис. 10.26

Вы попадете на страницу *WebMoney* банкинг (рис. 10.26). Жмите на кнопку «*Провести авторизацию*» (если вверху страницы, под адресной строкой, появится строка о том, что нужны элементы управления *ActiveX*, то нажмите мышью на эту появившуюся строку, чтобы установить элементы *ActiveX* для нормальной работы сервиса, и только после этого можно будет авторизироваться). Далее откроется форма, предупреждающая, что вы входите в защищенную зону сайта *http://passport.webmoney.ru/asp* (рис. 10.27).

Убедитесь, что справа, в окошечке списка, стоит именно ваш идентификатор (WMID) и нажмите на кнопку «*Да*». Вы попадете на следующую страницу (рис. 10.28).

В отделе «Акцепт договора продажи ценных бумаг» нажмите на ссылку, чтобы попасть на страницу http://passport.webmoney.ru/asp/pasMain.asp (рис. 10.29).

Нажимаем на ссылку слева «*Получить аттестат*». Вам предложат опять авторизироваться. После чего вы попадете на страницу для ввода своих данных (рис. 10.30).

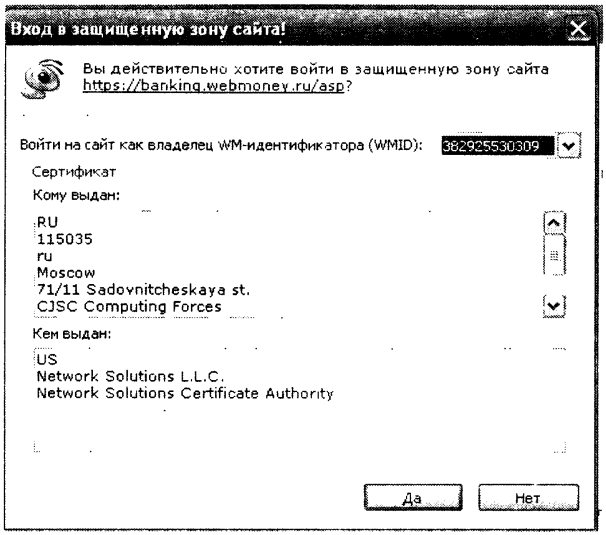


Рис. 10.27

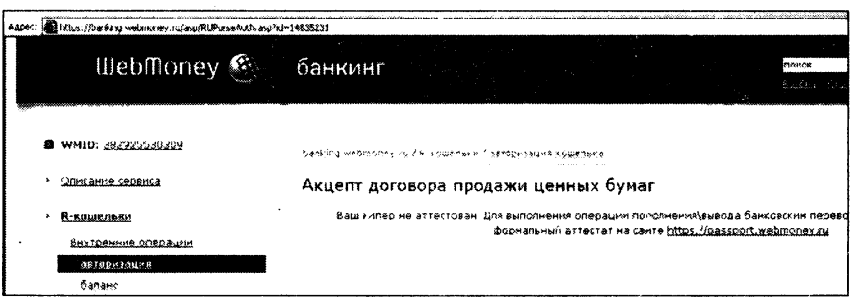


Рис. 10.28

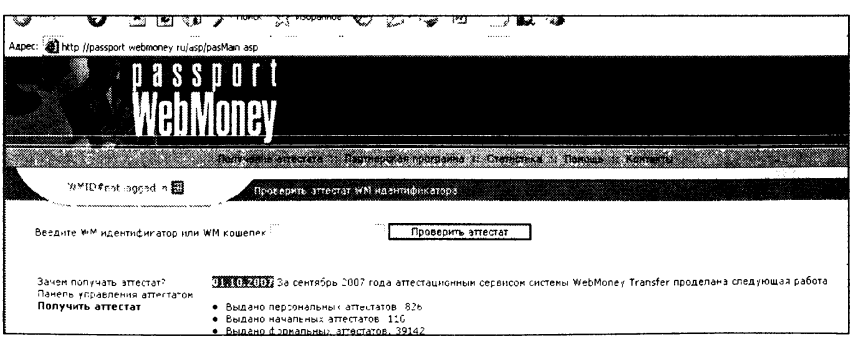


Рис. 10.29

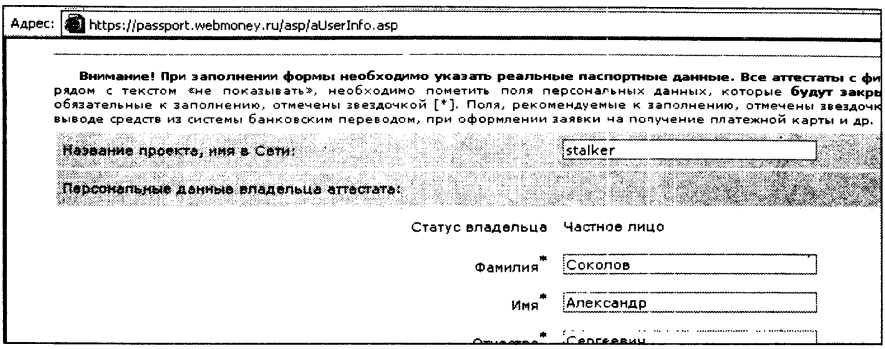


Рис. 10.30

Вводите реальные данные, в том числе и паспортные. Если данные будут ложные, а они проверяются, ваш кошелек может быть заблокирован! После заполнения и отправки всех данных вы получаете формальный аттестат.

Перезапустите теперь ваш Keeper, т. е. сначала закройте его, а затем запустите вновь (рис. 10.31).

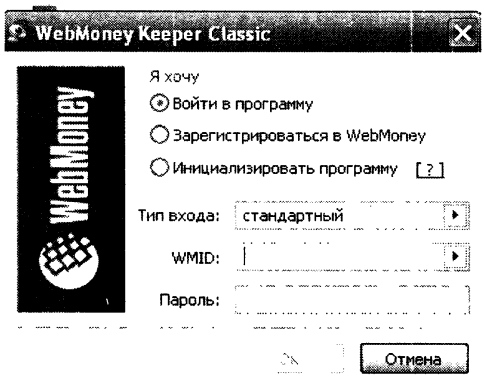


Рис. 10.31

Введите теперь свои идентификатор и пароль. Вы должны были их записать или запомнить при регистрации. Запустится Keeper (рис. 10.32).

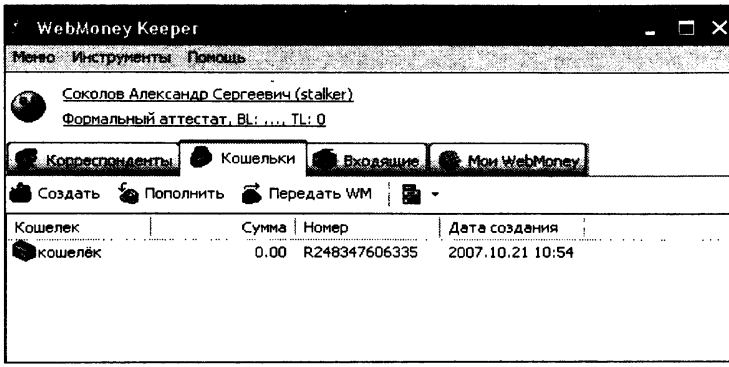


Рис. 10.32

Вверху должно быть указано, что вы обладаете формальным аттестатом. У вас уже будет рублевый кошелек, который можно будет пополнять денежными средствами. Ваш рублевый счет будет начинаться с буквы R и состоять из 12 цифр. Пополнять кошелек можно такими же способами, как и кошелек в *Yandex*, т. е. почтовым переводом, банковским переводом, при помощи карточек (*WebMoney* или WM-карты) или в специальных пунктах электронных платежей. Карточки *WebMoney* можно приобрести, например, в салонах мобильной связи. Более подробно, как пополнить кошелек, можно узнать, перейдя на вкладку «*Мои WebMoney*» и нажав потом там на ссылку «*Пополнить кошелек*».

Вообще в системе *WebMoney* существует несколько типов кошельков:

Z-кошелек, или кошелек типа Z (долларовый). Хранящиеся в нем деньги (их принято для удобства называть WMZ) эквивалентны долларам США. Таким образом, 1 WMZ=$1.

Номер Z-кошелька состоит из буквы «Z» и следующих за ней двенадцати цифр. Когда вы захотите получить деньги на Z-кошелек, просто скажите вашему покупателю номер своего Z-кошелька. И он переведет вам на него WMZ за покупку. При этом указание буквы «Z» перед номером обязательно;

R-кошелек, или кошелек типа R (рублевый), который мы создали выше. Хранящиеся в нем деньги (WMR) эквивалентны российским рублям: 1 WMR=1 RUR;

E-кошелек, или кошелек типа E. Хранящиеся в нем деньги (WME) эквивалентны евро. Номер E-кошелька состоит из буквы «E» и 12 цифр (например, E600012345678).

Есть и другие типы кошельков, например даже для китайских юаней.

И еще один важный момент. WM-идентификатор, о котором говорилось выше, и номер кошелька – абсолютно разные вещи. Первый используется для входа в Keeper и подключения к *WebMoney*, второй – для переводов внутри системы.

Все операции в системе *WebMoney* происходят между кошельками одного типа. То есть вы можете перевести деньги со своего Z-кошелька на Z-кошелек другого участника системы. Но не можете перевести деньги с Z-кошелька, к примеру, на R-кошелек и т. д.

Итак, на данный момент у вас есть:

- WM-идентификатор (12-значное число);
- пароль на вход;
- файл ключей *.kwm*, который вы храните в безопасном месте;
- код доступа к файлу ключей;
- рублевый кошелек.

10.4. ОРГАНИЗАЦИЯ ОПЛАТЫ НА САЙТЕ НА ПРИМЕРЕ СТРАНИЦЫ С MP3-ФАЙЛАМИ

Давайте создадим с вами страничку с цифровыми товарами, например с музыкальными файлами mp3. Посетитель этой странички сможет скачать один или несколько mp3-файлов себе на компьютер только после оплаты их стоимости. Оплата от клиентов автоматически будет поступать к вам на рублевый счет *WebMoney* кошелька. Оплату на вашем сайте можно осуществить при помощи сервиса *Web Merchant Interface*. Для этого нам нужно стать участником *WebMoney Transfer*, принимающий на свой WM-кошелек оплату от других участников системы с помощью сервиса *Web Merchant Interface*. *Web Merchant Interface* – это интерфейс, позволяющий быстро и с минимальными затратами дополнить свой веб-сайт или домашнюю страничку всем необходимым для того, чтобы начать принимать WM (деньги *WebMoney*) за предоставляемые вами товары или услуги. Как стать участником *WebMoney Transfer*, я объясню позже, когда весь создаваемый нами сайт вы закачаете на какой-нибудь хостинг. Сейчас я пока объясню, как организовать оплату на страницах сайта. В листинге 10.1 приведен примерный код, который необходимо установить на странице вашего веб-сайта для оплаты электронных товаров. Для начала создайте в вашей рабочей папке сервера *htdocs* новую подпапку под названием *muzic*. В PHP-редакторе создайте новый файл и сохраните его в подпапке *muzic* под названием *oplata.php*. Наберите приведенный ниже код.

Листинг 10.1 (файл oplata.php)

```
<html>
<head>
<title>Оплата</title>
</head>
<body>
<form id=pay name=pay method="POST"
action="https://merchant.webmoney.ru/lmi/payment.asp">
пример платежа через сервис Web Merchant Interface<br>
заплатить 10 WMR (10 рублей)...<br>
  <input type="hidden" name="LMI_PAYMENT_AMOUNT" value="10.0">
  <input type="hidden" name="LMI_PAYMENT_DESC" value="тестовый платеж">
  <input type="hidden" name="LMI_PAYMENT_NO" value="1">
  <input type="hidden" name="LMI_PAYEE_PURSE" value="R248347606335">
  <input type="hidden" name="LMI_SIM_MODE" value="0"><br>
  <input type="submit" value="Оплатить">
</form>
</body>
</html>
```

Здесь мы использовали только HTML-код, без использования тегов *<?php* и *?>*, поэтому данный файл можно было бы назвать и *oplata.html*.

Разберем код. Вы видите, что это форма с шестью скрытыми полями (*type="hidden"*) и одной кнопкой «*Оплатить*», после нажатия на которую все данные из формы передадутся методом *method="POST"* на адрес интерфейса оплаты *https://merchant.webmoney.ru/lmi/payment.asp*. Там эти данные обрабатываются, и в случае удачного платежа клиент направляется на страницу для скачивания того или иного электронного товара.

Рассмотрим теперь скрытые поля:

<input type="hidden" name="LMI_PAYMENT_AMOUNT" value="10.0"> – в этом поле указывается сумма платежа (в нашем случае *value="10.0"*), которую продавец желает получить от покупателя. Сумма должна быть больше нуля, дробная часть отделяется точкой;

<input type="hidden" name="LMI_PAYMENT_DESC" value="тестовый платеж"> – в этом поле вводится описание товара или услуги. Описание может быть любым, но не более 255 символов;

<input type="hidden" name="LMI_PAYMENT_NO" value="1"> – в этом поле продавец задает номер покупки в соответствии со своей системой учета. Этот параметр не является обязательным;

<input type="hidden" name="LMI_PAYEE_PURSE" value="R248347606335"> – здесь указывается *WebMoney* кошелек продавца, на который покупатель должен совершить платеж. Формат – буква и 12 цифр. В настоящее время допускается использование кошельков Z-,R-,E-,U- и D-типа;

<input type="hidden" name="LMI_SIM_MODE" value="0"> – дополнительное поле, определяющее режим тестирования. Действует только в режиме тестирования и может принимать одно из следующих значений: 0 или отсутствует – для всех тестовых платежей сервис будет имитировать успешное выполнение; 1 – для всех тестовых платежей сервис будет имитировать выполнение с ошибкой (платеж не выполнен); 2 – около 80 % запросов на платеж будут выполнены успешно, а 20 % – не выполнены. Сам режим тестирования можно задать при настройке сервиса *Web Merchant Interface*, который мы сделаем позже. В этом режиме будет только имитация приема платежей, то, что нам и нужно для обучения. Для приема реальных платежей на сайте мы позже установим в настройках сервиса *Web Merchant Interface* рабочий режим.

Ну, а теперь составим страничку с mp3-файлами, которые мы потом будем продавать всем желающим. Будем считать, что у вас есть разрешение на продажу данных электронных товаров. Вместо музыки можно, например, продавать составленные вами программы или игры. Сначала создайте в папке *muzic* две подпапки с названием *mp3* (там у нас будут находиться mp3-файлы) и *zakazi* (там будут находиться программно формируемые PHP-страницы с заказами клиентов). Загрузите в подпапку *mp3* несколько музыкальных файлов в mp3-формате.

Если вы захотите использовать составленную ранее таблицу стилей (у меня файл стилей называется *stili2.css*), скопируйте его в папку *muzic*.

Составим сначала главную музыкальную страницу, где у нас будут выставлены на продажу mp3-файлы. Посетитель может выбрать одну или несколько файлов для покупки. В PHP-редакторе создайте новый файл и сохраните его в папке *muzic* под названием *muzic.php*. Наберите код листинга 10.2.

Листинг 10.2 (файл muzic.php)

```
<html>
<head>
<title>Музыка</title>
<link rel="stylesheet" type="text/css" href="stili2.css">
</head>
<body>
```

```
<br>
<form action="muzic2.php" method="POST" name="forma">
<center><table border="2" bordercolor="cyan" id="lolo"><tr><td>Хорошая
музыка (цена каждой композиции 2 руб.)</td></tr></center><tr><td>
<?php
$myz=opendir("./mp3");
while (($file=readdir($myz)) !==false)
{
 if($file !="." && $file !="..")
 {
 $i++;
 echo "$file <input type=checkbox name='fleg$i'><I style='font-
size:11pt'>Заказать</I><br>";
 }
}
closedir($myz);
echo "<input type=hidden name=kol value=$i><br><input type=submit
value='заказать'>
</form>";
?>
</td></tr></table>
</body>
</html>
```

Разберем некоторые строчки кода:

<form action="muzic2.php" method="POST" name="forma"> – составляем форму, данные из которой мы передадим методом *POST* на страницу *muzic2.php*. Код для этой страницы мы составим позже;

* <table border="2" bordercolor="cyan" id="lolo"><tr><td>Хорошая музыка (цена каждой композиции 2 руб.)</td></tr> <tr><td>* – строим таблицу для вывода названий mp3-файлов в браузер;

<?php – начало PHP-кода;

$myz=opendir("./mp3");* – открываем папку с музыкальными файлами и присваиваем открытой папке дескриптор *$myz;

while (($file=readdir($myz)) !==false) – цикл, который мы часто использовали в других программах. Он перебирает все файлы в открытой папке и прекращается только после перебора всех файлов;

1{

if($file !="." && $file !="..") – если выбранный циклом *while* файл не является текущим или родительским каталогом, то выполнится код в фигурных скобках под номером 2;

2{

$i++; – вводим текущий счетчик файлов, подсчитывающий число музыкальных файлов в папке *mp3*;

*echo "$file <input type=checkbox name='fleg$i'><I style='font-size:11pt'>Заказать</I>
";* – выводим в браузер название музыкального файла (переменную *$file*) и переключатель *type=checkbox*, при помощи которого посетитель сможет выбрать (поставить галочку в переключателе) тот или иной музыкальный файл. Для первого в папке музыкального файла переключатель будет иметь имя *name='fleg1'*, для второго *name='fleg2'* и т. д. Рядом с переключателем выводим слово «*Заказать*». Вид шрифта можете указать свой при задании стиля *style*;

2}

1} – окончание цикла *while*. В переменной *$i* будет находиться общее число всех музыкальных файлов в папке *mp3*;

closedir($myz); – закрываем папку с mp3-файлами;

*echo "<input type=hidden name=kol value=$i>
*

<input type=submit value='заказать'>

</form>"; – при помощи оператора *echo* выводим в форме скрытое поле с параметром *value=$i* (общее число музыкальных файлов в папке *mp3*), чтобы передать его на страницу *muzic2.php*, после нажатия кнопки *type=submit* с надписью «*Заказать*»;

?> – конец PHP-кода;

</td></tr></table> – закрываем единственную ячейку таблицы, строку и саму таблицу.

Запустите программу. Если вы все сделали правильно, то в браузер будет выведен список музыкальных файлов из папки mp3. У меня, например, этот список показан на рис. 10.33.

Возле каждого названия музыкальной композиции посетитель может поставить галочку, если захочет приобрести данную композицию. После выбора композиций и нажатии внизу кнопки «*Заказать*» посетитель попадет на страницу *muzic2.php* (см. листинг 10.3), где в браузер будут выведены названия композиций, которые выбрал посетитель, общая их стоимость, кнопка для перехода на страницу оплаты сервиса *Web Merchant Interface*.

В PHP-редакторе создайте новый файл и сохраните его в папке *muzic* под названием *muzic2.php*. Наберите код листинга 10.3.

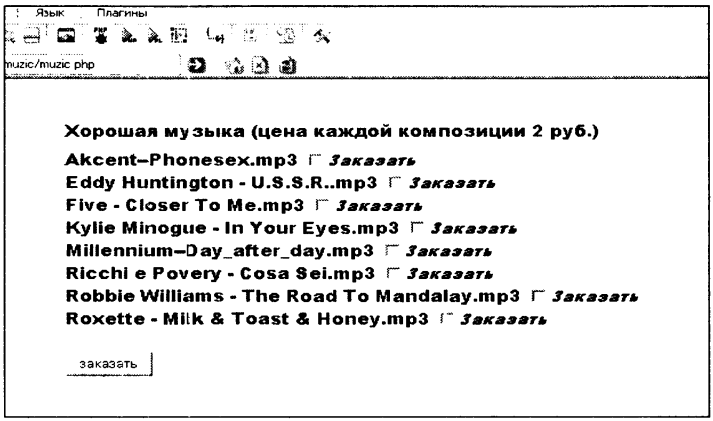


Рис. 10.33

Листинг 10.3 (файл muzic2.php)

```
<html>
<head>
<title>Заказ музыки</title>
<link rel="stylesheet" type="text/css" href="stili2.css">
</head>
<body>
<?php
$zakaz="Вы заказали: ";
$zakaz2="";
$n=$_POST['kol'];
$myz=opendir("./mp3");
while (($file=readdir($myz)) !==false)
{
 if($file !="." && $file !="..")
 {
  $j++;
  if($_POST['fleg'.$j]==true)
  {
   $zakaz="$zakaz $file<br>";
   $zakaz2=$zakaz2."<a href='../mp3/$file' id='lolo'>$file</a><br>";
   $k++;
```

```
 }
 }
}
closedir($myz);
$k=$k*2;
echo "<I id=lolo>$zakaz</I>На общую сумму: $k руб";
if ($k>0)
{
$list='<html>
<head>
<title>Untitled web-page</title>
<link rel="stylesheet" type="text/css" href="../stili2.css">
</head>
<body>
<?php
echo "Спасибо за заказ <br>'.$zakaz2.'";
?>
</body>
</html>';
$z="zakazi/".time().".php";
$open=fopen($z, "w");
fwrite($open, $list);
fclose($open);
}
echo "<form id=pay name=forma method=POST
action='https://merchant.webmoney.ru/lmi/payment.asp'>
  <input type=hidden name='LMI_PAYMENT_AMOUNT' value='$k'>
  <input type=hidden name='LMI_PAYMENT_DESC' value='тестовый платеж'>
  <input type=hidden name='LMI_PAYMENT_NO' value='1'>
  <input type=hidden name='LMI_PAYEE_PURSE' value='R248347606335'>
  <input type=hidden name='LMI_SIM_MODE' value='0'><br>
  <input type=hidden name=LMI_ SUCCESS_URL value='http://www.ваш
сайт/muzic/$z'>
  <input type=submit value='Оплатить'>
  </form>";
?>
</body>
</html>
```

Разберем некоторые строчки кода:

<?php – начало PHP-кода;

$zakaz="Вы заказали: ";* – вводим переменную *$zakaz, чтобы потом вывести ее в окно браузера с содержимым заказа;

$zakaz2=""; – в эту переменную мы потом запишем ссылки на выбранные музыкальные файлы;

$n=$_POST['kol']; – в прошлом листинге (программа *muzic.php*) при помощи скрытого поля с именем *name=kol* мы передали на эту страницу (*muzic2.php*) число, обозначающее общее количество музыкальных файлов в папке *mp3*. Это число, которое мы присвоим переменной *$n*, будет храниться в элементе суперглобального массива *$_POST['kol']*;

$myz=opendir("./mp3"); – открываем папку с музыкальными файлами;

while (($file=readdir($myz)) !==false) – цикл, перебирающий все mp3-файлы в открытой папке *mp3*;

1{

if($file !="." && $file !="..")

2{

$j++; – вводим текущий счетчик файлов, подсчитывающий число музыкальных файлов в папке *mp3*;

if($_POST['fleg'.$j]==true) – в прошлом листинге (*muzic.php*) именами *name=fleg1*, *name=fleg2* и т. д. мы называли элементы формы *type=checkbox* (переключатели). Состояние этих переключателей передалось на данную страницу (*muzic2.php*) в элементы суперглобального массива *$_POST['fleg1']*, *$_POST['fleg2']* и т. д. Если посетитель на странице *muzic.php* выбрал, например, первую композицию (поставил галочку в переключателе), то элемент *$_POST['fleg1']* примет значение *true*. Итак, если посетитель выбрал mp3-композицию, которая на данный момент перебирается циклом *while*, то выполнится код между фигурными скобками под номером 3;

3{

*$zakaz="$zakaz $file
";* – к переменной *$zakaz* добавляем название выбранного музыкального файла;

*$zakaz2=$zakaz2."$file
";* – а к переменной *$zakaz2* добавляем ссылку на выбранный файл. Заметьте, что перед каталогом *mp3* стоят две точки со слешем (../), указывающие на родительский каталог. Дело в том, что данную переменную мы будем использовать только на новых страницах, которые ниже сформируются программно. Эти сформированные страницы будут находиться в подпапке *zakazi*, а сами mp3-файлы находятся в подпапке *mp3*. Если мы бы записали ссылку в виде:

a href='./mp3/$file' (с одной точкой, указывающей на текущий каталог), то программа искала бы подпапку *mp3* в папке *zakazi* и, естественно, не нашла бы их, так как подпапка *mp3* находится в папке *muzic* (для подпапки *zakazi*, папка *muzic* будет являться родительским каталогом);

$k++;* – увеличиваем счетчик выбранных файлов на единицу (первоначальное его значение *$k=0). После перебора циклом *while* всех файлов, в этой переменной будет общее число выбранных посетителем mp3-файлов;

3}

2}

1} – окончание цикла *while*;

closedir($myz); – закрываем папку с музыкальными файлами;

*$k=$k*2;* – высчитываем общую стоимость выбранных посетителем файлов. Мы устанавливаем стоимость каждого файла 2 рубля, но можете установить по-своему;

echo "<I id=lolo>$zakaz</I>На общую сумму: $k руб";* – выводим в браузер заказ посетителя (переменную *$zakaz) и общую стоимость заказа в рублях;

if ($k>0) – если число выбранных файлов больше нуля, т. е. посетитель выбрал хотя бы одну композицию, то выполнится код между фигурными скобками под номером 4. Начнет формироваться страница со ссылками для скачивания выбранных посетителем файлов. Перейти к ней можно будет только после оплаты;

4{

$list='<html>* – создаем переменную *$list, где мы будем формировать код для программно создаваемой страницы со ссылками для скачивания выбранных mp3-файлов;

<head>

<title>Заказ </title>

<link rel="stylesheet" type="text/css" href="../stili2.css">

</head>

<body>

<?php – начало PHP-кода на формируемой странице;

*echo "Спасибо за заказ
'.$zakaz2.'";* – при помощи оператора *echo* (в двойных кавычках) выводим в браузер благодарность за заказ и ссылки для скачивания выбранных файлов. Эти ссылки, если помните, у нас в переменной *$zakaz2*. Эту переменную мы вставляем в код создаваемой страницы так, как это делали раньше в других программах. Мы как бы прерываем писать переменную *$list*, т. е. закрываем одинарную кавычку, вписываем переменную *$zakaz2* и вновь открываем одинарную кавычку;

?> – окончание PHP-кода на формируемой странице;
</body>

</html>'; – заканчиваем писать переменную *$list* (закрываем одинарную кавычку);

$z="zakazi/".time().".php";* – в переменную *$z запишем название новой формирующейся страницы, точнее, путь к ней. Все новые сформированные страницы для скачивания заказанных файлов будут храниться в папке *zakazi* (вы должны были создать ее заранее). Функция *time()* (см. гл. 4) содержит многозначное число, обозначающее количество секунд, прошедших с 1 января 1970 г. Следовательно, как нетрудно догадаться, названия файлов будут в виде многозначного числа и имеющие расширения *php*;

$open=fopen($z, "w"); – создаем и открываем для записи новую страницу с названием, сформированной в переменной *$z*;

fwrite($open, $list);* – записываем в новый файл код, который мы создали в переменной *$list;

fclose($open); – закрываем созданную страницу;

4} – окончание условия *if*, при выполнении которого и формировались новые страницы;

echo "<form id=pay name=forma method=POST action='https://merchant.webmoney.ru/lmi/payment.asp'> – выводим форму, для отправки данных на сайт для оплаты https://merchant.webmoney.ru/lmi/payment.asp (см. листинг 10.1). Почти все поля формы были разобраны в листинге 10.1;

<input type=hidden name='LMI_PAYMENT_AMOUNT' value=' $k'>* – в поле с именем *name='LMI_PAYMENT_AMOUNT'*, если помните, указывается сумма платежа, которую должен заплатить клиент. Эта сумма у нас хранится в переменной *$k. Ее мы и передаем на сайт для оплаты.

<input type=hidden name='LMI_PAYMENT_DESC' value='тестовый платеж'>

<input type=hidden name='LMI_PAYMENT_NO' value='1'>

<input type=hidden name='LMI_PAYEE_PURSE' value='R248347606335'>

*<input type=hidden name='LMI_SIM_MODE' value='0'>
*

<input type=hidden name=LMI_SUCCESS_URL value='http://www.ваш сайт/muzic/$z'> – в поле с именем *name=LMI_ SUCCESS_URL* мы указываем страницу, куда должен перейти посетитель в случае оплаты. А перейти он должен на страницу для скачивания заказанных файлов. Путь к этой странице, которая была программно сформирована, хранится в переменной *$z*. Заметьте, в параметре *value* нужно указывать полный путь к этой странице,

с указанием названия вашего сайта на сервере. Пока вы используете домашний сервер на своем компьютере, ваш сайт не имеет еще названия во всемирной сети. Только после того, как вы закачаете созданный сайт на какой-нибудь сервер, поддерживающий php4 и php5, у вашего сайта появится доменное имя, которое вы потом и укажете в параметре *value*. Затем мы с вами на сайте *https://merchant.webmoney.ru* сделаем настройки для приема оплаты с вашего сайта с помощью сервиса *Web Merchant Interface*. Все это будет сделано в следующей главе;

<input type=submit value='Оплатить'> – кнопка, при нажатии на которую данные из этой формы передаются на сайт для оплаты товаров (в нашем случае для оплаты файлов mp3);

</form>"; – закрываем форму;

?> – окончание PHP-кода.

В PHP-редакторе снова запустите файл *muzic.php* (см. листинг 10.2 и рис. 10.33), выберите несколько mp3-файлов (поставьте галочки напротив них) и нажмите на кнопку «*Заказать*». Сначала, возможно, вам будет предложено повторно отправить данные (см. рис. 6.1). Жмите на кнопку «*Отмена*», поскольку повторно отправлять данные нет необходимости. Вы попадете на страницу *muzic2.php*, где увидите свой заказ и общую сумму заказа (рис. 10.34).

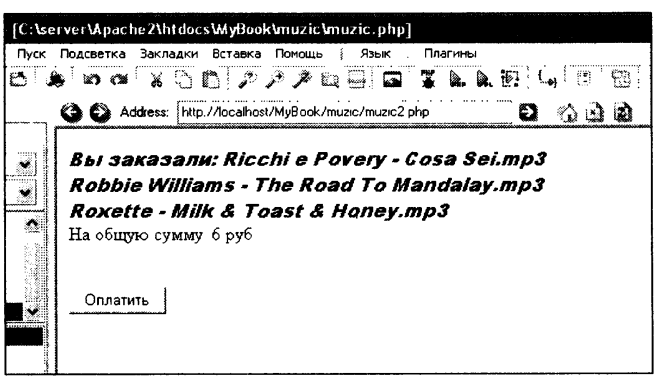


Рис. 10.34

Я выбрал 3 последних файла на общую сумму 6 рублей. Цвет и тип шрифта, как вы поняли, задан в идентификаторе стиля *id='lolo'*, который, в свою очередь, задан в листе стилей *stili2.css*. Нажимать на кнопку «*Оплатить*» еще рано, даже если вы подключили свой компьютер к Интернету, поскольку мы еще не сделали настройки для оплаты на сайте *https://merchant.webmoney.ru* (мы это сделаем в гл. 11). Сформировавшуюся

страницу с ссылками к файлам mp3 можно увидеть в папке *zakazi*. У меня, например, в этой папке появился файл *1192778001.php* (у вас цифра будет другой). Многозначная цифра, как вы уже догадались, означает количество секунд, прошедших с нуля часов 01.01.1970 до момента формирования файла со ссылками. Откройте в редакторе этот сформированный файл. У меня там, соответственно, 3 ссылки на 3 заказанных файла (рис. 10.35).

Рис. 10.35

Ссылки на mp3-файлы, которые у нас в папке *mp3*, должны сработать. Если этого не происходит, проверьте, правильно ли вы указали ссылки на mp3-файлы в переменной *$zakaz2* в листинге 10.3. Имейте в виду, что папка *mp3* с музыкальными файлами находится в каталоге *muzic*, который является родительским каталогом (стоит на ступень выше) для каталога *zakazi*, где и находятся сформированные страницы со ссылками. Поэтому, чтобы на странице со ссылками, находящейся в подпапке *zakazi*, указать ссылку на музыкальный файл в подпапке *mp3*, надо сначала «подняться» в родительский каталог *muzic*. Вот почему в начале ссылки *a href='../mp3/$file'* мы ставим не одну, а две точки. В этом случае ссылки на mp3-файлы сработают.

Если посетитель захочет скачать файлы, он просто нажмет на ссылке правой кнопкой мыши и в контекстном меню выберет пункт «*Сохранить объект как...*».

Как я уже отмечал выше, посетитель должен заполучить сформированную страницу со ссылками не «на халяву», а только после оплаты. Как это сделать и как стать участником *WebMoney Transfer*, чтобы иметь возможность продавать электронные товары со своего сайта, рассказано в гл. 11. Но сначала составим страницу, на которую попадет посетитель в случае неудачного платежа (сбой на сервере или сумма в электронном кошельке посетителя недостаточна для оплаты). В PHP-редакторе создайте новый файл и сохраните его в папке *muzic* под названием *muzic3.php*. Наберите небольшой код листинга 10.4.

Листинг 10.4 (файл muzic3.php)

```
<html>
<head>
<title>Неудачный платеж</title>
</head>
<body>
<b>Извините, но платеж не прошел!</b><br>
<a href="muzic.php">Вернуться</a>
<?php

?>
</body>
</html>
```

Мы здесь не использовали PHP-код, поэтому страницу можно было бы сохранить и с расширением *html*.

Посетитель, попав на эту страницу (рис. 10.36), может опять вернуться на страницу для заказа mp3-товаров *muzic.php*.

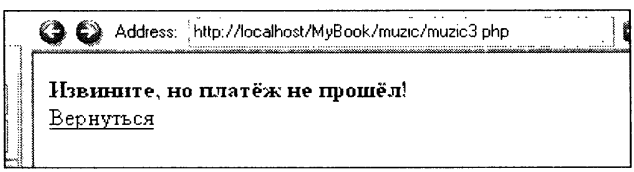


Рис. 10.36

Глава 11. ЗАКАЧКА САЙТА НА ХОСТИНГ. ПРАКТИЧЕСКОЕ ОСУЩЕСТВЛЕНИЕ ОПЛАТЫ ТОВАРОВ С САЙТА

11.1. ВЫБОР ХОСТИНГА ДЛЯ САЙТА, ЗАКАЧКА САЙТА

Итак, сайт на локальном компьютере создан. Еще раз на своем компьютере проверьте его работоспособность. Убедитесь, что срабатывают ссылки на другие страницы. Если все в порядке, то теперь созданный сайт можно закачать на какой нибудь хостинг. Есть, конечно, бесплатные хостинги, типа *yandex.narod.ru*, но там невозможно использовать составленные вами PHP-скрипты. При выборе хостинга (сервера) вы должны убедиться, что данный сервер поддерживает php4 и php5 и позволяет использовать базу данных. Мой сайт, например, находится на хостинге *majordomo.ru*. Там разные тарифы. Все зависит от того, сколько места вам нужно под сайт, нужна ли вам база данных и т. п.

Допустим, вы выбрали сервер, где будете хранить свой сайт. Придумав себе доменное имя и оплатив хостинг, вы можете закачать на сервер созданный вами сайт. Если вам непонятно, как и куда закачивать файлы, можете спросить у хостера по электронной почте.

Закачайте в корневой каталог выбранного вами сервера все созданные вами файлы из папки *htdocs*. Ваш компьютер, естественно, должен быть подключен к Интернету. Затем, после того как все закачаете, в верхней строке браузера наберите адрес вашего сайта (доменное имя): ***http://Ваше доменное имя.ru*** (если вы выбрали домен с другим расширением, например *com*, то наберите в строке браузера: ***http://Ваше доменное имя.com***). У вас должна открыться главная страница сайта *index.php* (рис. 3.7). Проверьте, срабатывают ссылки на другие страницы, работоспособен ли счетчик посещений. Запишите в гостевую книгу какое-нибудь сообщение и посмотрите потом, показывается ли ваше сообщение в браузере. Оставьте заказ какой-либо книги в вашем магазине, посмотрите, обрабатывается ли заказ. В общем, проверьте свой сайт.

У вас пока не будет работать часть сайта для оплаты музыкальных композиций. Как было сказано в прошлой главе, для организации оплаты электронных товаров с сайта нам нужно зарегистрироваться в системе оплаты *webmoney*, точнее, нужно стать участником *WebMoney Transfer*. Этим мы сейчас и займемся.

11.2. ИСПОЛЬЗОВАНИЕ СЕРВИСА *WEB MERCHANT INTERFACE* ДЛЯ ОПЛАТЫ

Сервис *Web Merchant Interface* позволяет получать денежные средства на свои электронные кошельки от клиентов при оплате ими электронных товаров на вашем сайте. Электронные кошельки *WebMoney* и *Yandex* мы с вами создали в прошлой главе.

Итак, зайдите на сайт: *http://merchant.webmoney.ru*. Вы попадете на сайт для участников *WebMoney Transfer* (рис. 11.1).

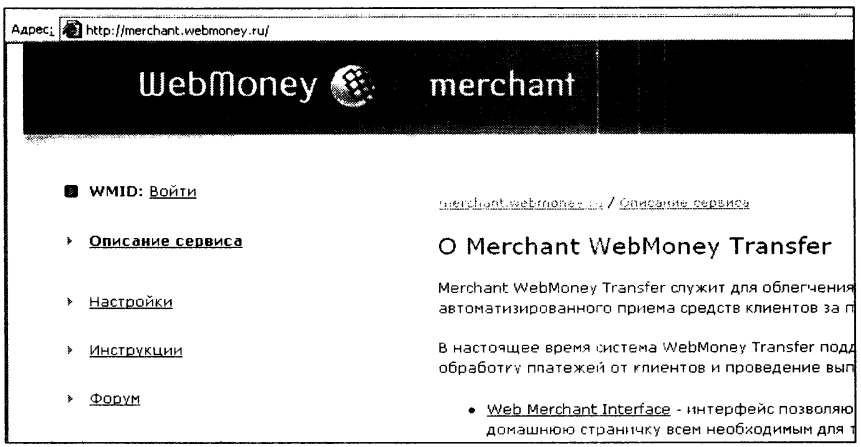


Рис. 11.1

Нажимаем слева ссылку «*Настройки*» и попадаем на следующую страницу (рис. 11.2).

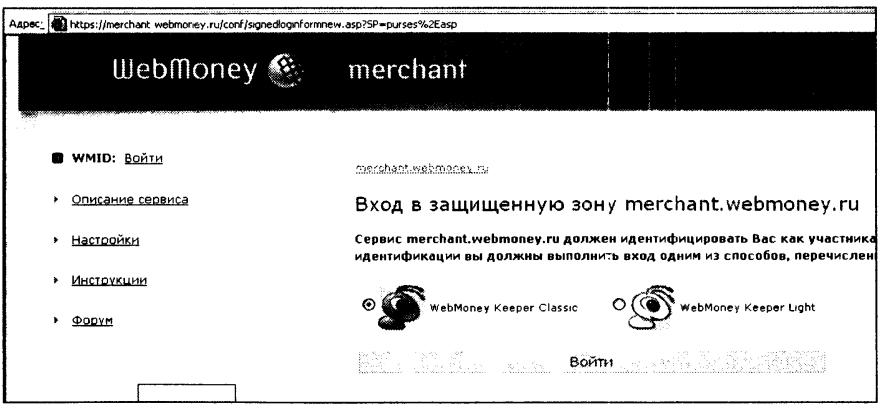


Рис. 11.2

Потребуют запустить ваш Кипер (желательно сделать это заранее). Нажмите на ссылку «*Войти*». После чего введите *WMID* (идентификатор) и пароль вашего Кипера *WebMoney* (см. гл. 10). Если запускаете созданный вами в прошлой главе кошелек на компьютере впервые, то понадобится файл ключей (файл с расширением *kwm*, который должен был быть сохранен на вашем компьютере, когда вы создавали кошелек *WebMoney*).

После этого попадаем на список ваших кошельков (рис. 11.3). В прошлой главе мы пока создали только один *WebMoney* кошелек (рублевый).

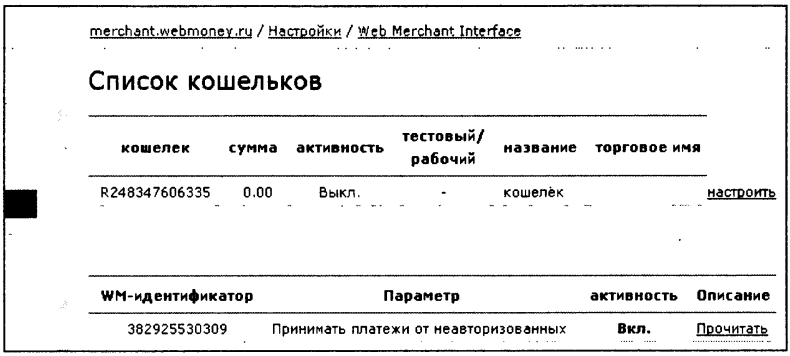


Рис. 11.3

Далее жмите на ссылку «*Настроить*» справа от указанного кошелька. Настроим наш кошелек (рис. 11.4).

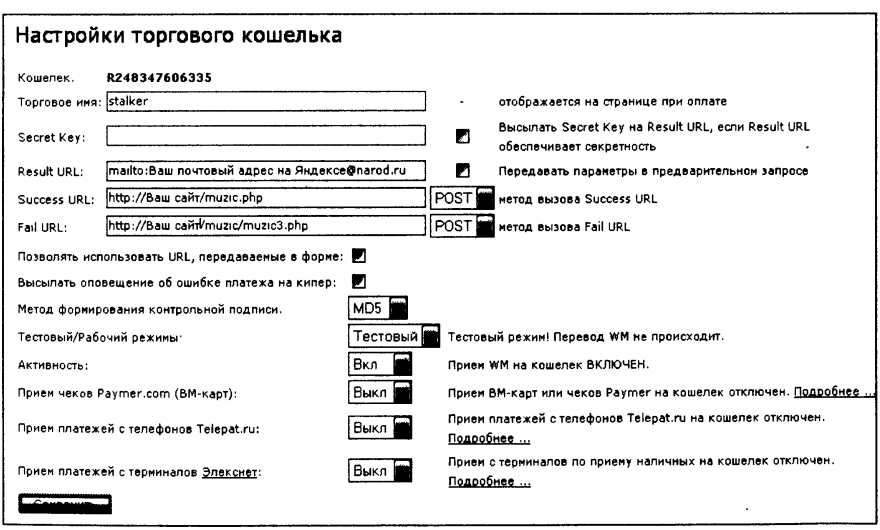


Рис. 11.4

Заполняем поля. Торговое имя можно придумать свое. В поле *Result URL* укажите ваш электронный адрес, который вы создали на *Yandex*. Впрочем, можно указать и любой другой, если он у вас есть. По этому адресу вам будет высылаться оповещение о платеже. Перед адресом обязательно укажите «*mailto:*». В поле *Success URL* указывается путь к странице, на которую попадет посетитель в случае удачного платежа. Пока укажите там путь к странице с музыкальными товарами *muzic.php* (листинг 10.2). Она у нас, если помните, находится в папке *muzic*. Вместо фразы «*Ваш сайт*» укажите доменное имя вашего сайта на выбранном вами сервере. Справа от данного поля выберите метод вызова POST.

В поле *Fail URL* указывается путь к странице, на которую попадет посетитель в случае неудачного платежа. Эта страница у нас тоже находится в папке *muzic* и называется *muzic3.php* (листинг 10.4). Справа от данного поля выберите метод вызова POST.

Поставьте галочку справа от предложения: «*Позволять использовать URL, передаваемые в форме*». Это нам даст возможность изменять страницу, указанную в поле *Success URL*, т. е. страницу, на которую попадет посетитель, в случае удачного платежа. Если вы разобрались с листингом 10.3 (файл *muzic2.php*), то поймете, что для разных клиентов, страницы с оплаченными товарами будут и называться по-разному. Вспомните листинг файла *muzic2.php*. В скрытом поле *LMI_SUCCESS_URL* мы указали:

<input type=hidden name=LMI_SUCCESS_URL value='http://www.ваш сайт/muzic/$z'>*, где в переменной *$z находится название файла для скачивания оплаченных электронных товаров. Содержимое этого файла, естественно, будет меняться, поскольку разные посетители сайта будут заказывать и разные товары. Само название файла для разных посетителей тоже будет различным. Чуть выше, в том же листинге, вы увидите строчку:

$z="zakazi/".time().".php". Все новые сформированные страницы для скачивания заказанных файлов будут храниться в папке *zakazi* (вы должны были создать ее заранее). Функция *time()* (см. гл. 4) содержит многозначное число, обозначающее количество секунд, прошедших с 1 января 1970 г. до момента заказа музыкальных товаров каким-либо посетителем. Естественно, для разных посетителей и время заказа будет разным.

Далее установите тестовый режим работы, при котором перевод денег на ваш кошелек не будет осуществляться. Чтобы посетители могли осуществлять реальные платежи на ваш *WebMoney* кошелек, вы должны получить персональный аттестат. Если помните, для своего *WebMoney* Кипера вы получили формальный аттестат (см. гл. 10). Имея формальный аттестат, вы сможете зачислять на свой кошелек деньги (при помощи карточек или почто-

вого перевода) и тратить их (оплачивать покупки в Интернет-магазинах, мобильную связь и т. п.). Однако формальный аттестат не позволяет вам получать деньги от посетителей вашего сайта. Тут уже нужен как минимум персональный аттестат.

Что такое аттестат и зачем он нужен, можно узнать по адресу: *http://passport.webmoney.ru/asp/WMCertify.asp*.

Документы, необходимые для получения персонального аттестата, указаны по адресу: *http://passport.webmoney.ru/asp/documents.asp*.

Аттестат выдается аттестаторами (регистраторами) либо при личной встрече, либо при получении ими копии ваших документов. Список аттестаторов, стоимость аттестата и условия получения можно узнать по адресу: *http://passport.webmoney.ru/asp/Reglist.asp?rettid=130*.

Если в вашем городе нет аттестаторов, то зайдите на адрес: *http://passport.webmoney.ru/asp/faq2.asp?q=4*.

Там подробно рассмотрен данный вопрос.

Вообще, если вы захотите заняться серьезным Интернет-бизнесом, персональный аттестат вам будет просто необходим. А пока мы будем использовать тестовый режим платежей.

После заполнения всех полей нажмите внизу кнопку «*Сохранить*» (см. рис. 11.4).

Проверим работоспособность платежной системы *WebMoney Transfer*. Наберите в браузере адрес нашей музыкальной страницы *muzic.php* (см. рис. 10.34): *http://Название вашего домена/muzic/muzic.php*.

Выберите несколько композиций и нажмите «*Заказать*». Вы попадете на страницу *muzic2.php* (см. рис. 10.35). Нажимаете «*Оплатить*» и попадаете на страницу оплаты *WebMoney Transfer* (рис. 11.5). Если ваш Кипер еще не запущен (где мы создавали кошелек WebMoney), тот запустите его. Впрочем, он запустится автоматически. Вам только нужно будет указать идентификатор и пароль. Вообще желательно создать другой кошелек, чтобы производить с него оплату на кошелек нашего сайта, который мы создали раньше.

Итак, далее нажимаете «*Оплатить*» и выбираете вид оплаты. У нас это будет оплата с кошелька *WebMoney Keeper Classic*.

После появится надпись, что вы входите в защищенную зону сайта. Нажмите «*Да*». Далее нужно подтвердить платеж. После оплаты вы попадете на страницу для скачивания оплаченных товаров (см. рис. 10.36). Все – платеж завершен, товар получен. Все это, правда, в тестовом режиме, поскольку персонального аттестата у вас нет. Если вы его получите, то в настройках своего кошелька (см. рис. 11.4) тестовый режим замените на рабочий, а в листинге 10.3 (файл *muzic2.php*) можете удалить строку:

<input type=hidden name='LMI_SIM_MODE' value='0'>.

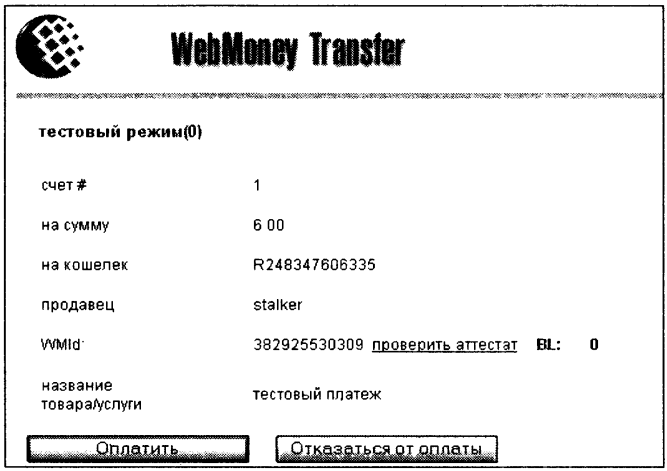


Рис. 11.5

11.3. ИСПОЛЬЗОВАНИЕ СЕРВИСА *ROBOXCHANGE* ДЛЯ ОПЛАТЫ

Если у вас нет никакой возможности получить персональный аттестат, или просто лень это делать, то для оплаты товаров со своего сайта вы можете воспользоваться сервисом *Roboxchange*. Этот сервис дает возможность получать деньги с клиентов на *Yandex* кошелек. Персональный аттестат при этом не требуется. При этом существуют некоторые ограничения, но для вашего, построенного при помощи этой книги сайта сервис *Roboxchange* вполне приемлем. Мы будем использовать *Yandex* кошелек, который вы создали в гл. 10. Но одного кошелька нам мало. Чтобы осуществлять платежи с одного кошелька на другой, нужно, естественно, иметь два Яндекс-кошелька. Зарегистрируйтесь на Яндексе под новым именем и создайте второй кошелек. Если забыли, как это делается, перечитайте прошлую главу. Далее в кошельки (хотя бы во второй созданный кошелек, откуда мы будем перечислять деньги) нужно положить хоть какую-то сумму денег, иначе вы не сможете осуществлять платежи. Тестового режима в *Roboxchange* нет, поэтому в Яндекс-кошелек вашего сайта будут перечисляться реальные деньги.

Для начала зайдите на сайт Roboxchange: *http://www.roboxchange.com* (см. рис. 11.5).

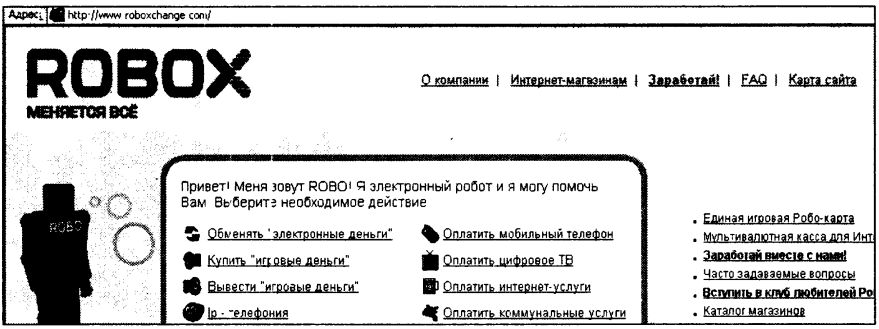


Рис. 11.5

На этой странице вы можете узнать практически все о платежной системе *Roboxchange*. Прочитаете на досуге. Нас пока интересует, как организовать платежи со своего сайта при помощи *Roboxchange*. Вверху нажмите на ссылку «*Интернет-магазинам*». Попадете на страницу для Интернет-магазинов (рис. 11.6).

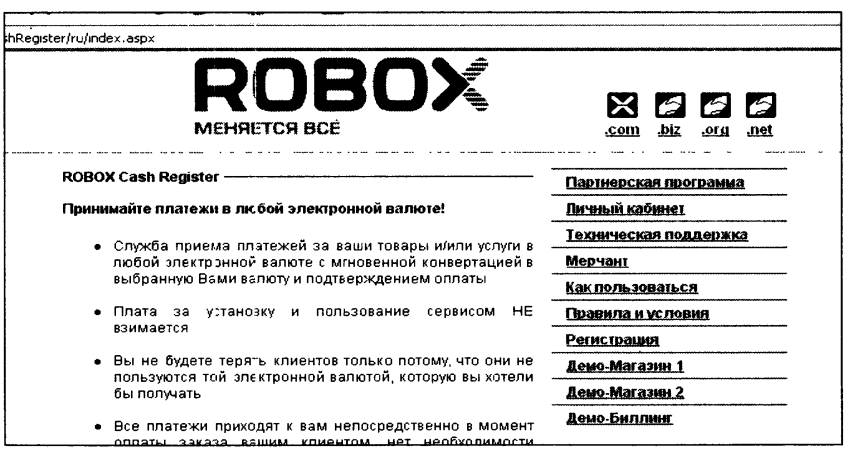


Рис. 11.6

Поскольку мы еще не зарегистрировали здесь свой магазин, жмем далее справа на ссылку «*Регистрация*». Далее вы попадете на страницу, где вы должны придумать свой логин. На следующей странице вам предложат придумать пароль. Потом вы должны указать свой электронный адрес (рис. 11.7). Укажите тот адрес, который вы получили на *Yandex* в прошлой главе или, если хотите, укажите любой другой ваш электронный ящик.

После записи вами вашего E-mail нажмите кнопку «*Сохранить*». Вы попадете на первую страницу регистрации (рис. 11.8).

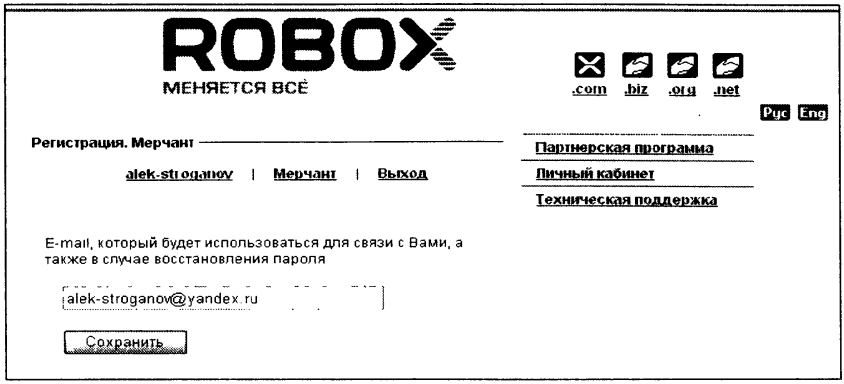


Рис. 11.7

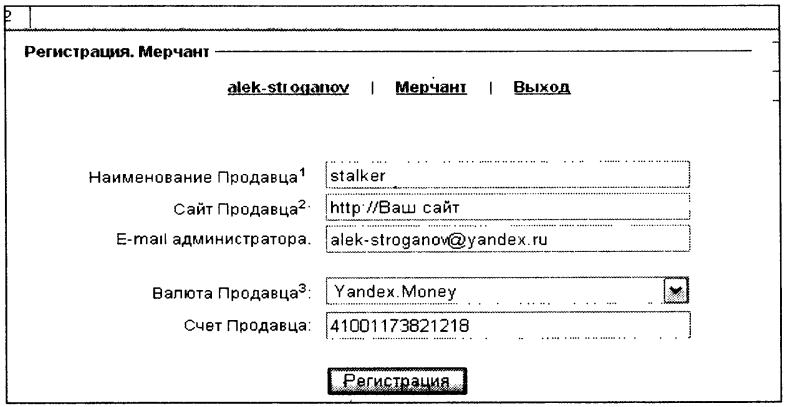


Рис. 11.8

В первое поле придумайте и запишите название вашего магазина латинскими буквами. Во втором поле укажите название вашего сайта (домена). В третьем поле укажите свой E-mail. В строке «*Валюта продавца*» выберите *Yandex.Money*. В строке «*Счет продавца*» укажите электронный счет вашего *Yandex* кошелька, который вы создали в прошлой главе (если помните, этот счет состоит из 14 цифр, начиная с 410...). Именно на этот кошелек будут поступать деньги от клиентов с вашего сайта. Второй созданный вами кошелек будет играть роль кошелька клиента, с которого будут перечисляться деньги.

Нажмите на кнопку «*Регистрация*». Вы попадете на вторую страницу регистрации (рис. 11.9). Если вы все заполнили правильно, то вы увидите запись о том, что вы зарегистрированы в системе *Roboxchange* в качестве продавца.

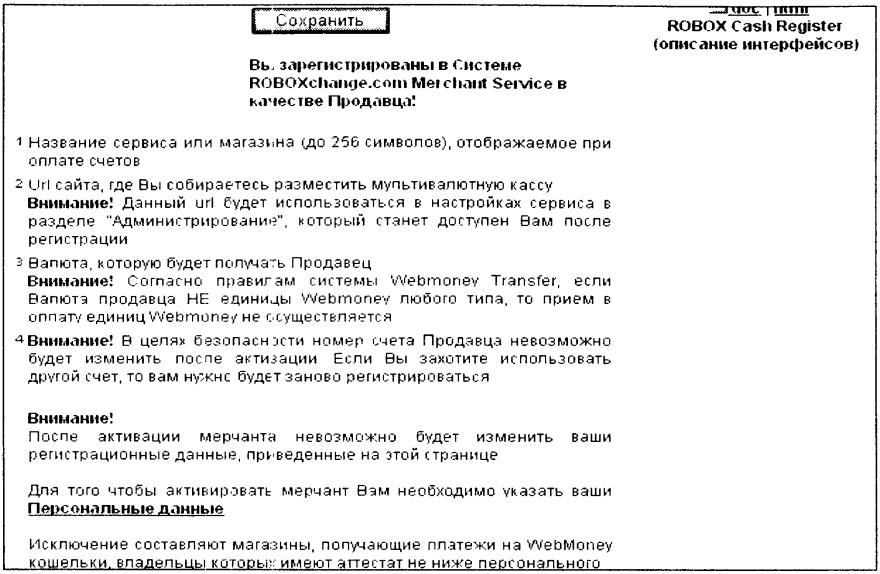


Рис. 11.9

Для того чтобы ваш магазин (мерчант) был активирован, нужно будет ввести ваши персональные данные. Жмите на ссылку «*Персональные данные*», и вы попадете на страницу ввода данных. Вы должны указать имя, фамилию, отчество, место жительства, телефон и т. п. Указывайте реальные данные, поскольку администрация сервиса *Roboxchange* эти данные будет проверять. Сначала вам предложат ввести свои данные на вашем национальном языке, затем, на следующей странице, на английском языке. Если вы английским не владеете, можете записать все на латинском, например, *улица Полевая* можно на латинском написать так: *ylica Polevaja*. После ввода всех данных нажмите внизу на кнопку «*Сохранить*».

Если все в порядке, то появится надпись, что данные успешно сохранены. Далее вверху нажмите на ссылку «*Мерчант*». Затем справа нажмите на ссылку «*Администрирование*». Попадем на страницу управления магазином (рис. 11.10).

Там нужно заполнить несколько полей:

1. В поле 1 придумайте и запишите первый пароль. Он используется интерфейсом инициации оплаты.
2. В поле 2 придумайте и запишите второй пароль. Он используется интерфейсом оповещения о платеже. Все пароли запишите еще куда-нибудь для себя, чтобы не забыть.

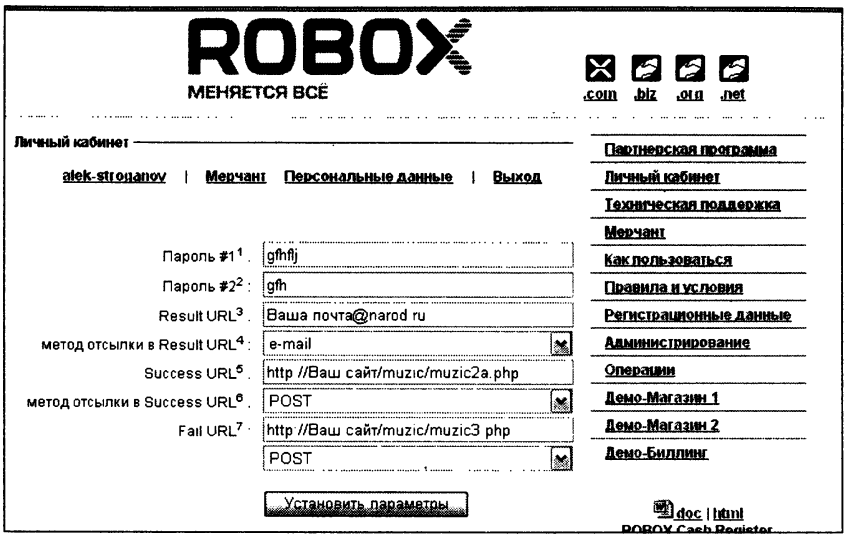


Рис. 11.10

3. В поле 3 впишите свой e-mail-адрес. На этот адрес вам будут приходить оповещения о платеже. Укажите электронный адрес, полученный вами на Яндексе.
4. В поле 4, где указывается метод отсылки данных в *Result URL*, выберите «*e-mail*».
5. В поле *Success URL* указывается путь к странице, на которую попадет посетитель в случае удачного платежа. Укажите там путь к странице с музыкальными товарами *muzic2a.php*. Мы ее создадим позже. Она у нас будет находиться в папке *muzic*. Вместо фразы «*Ваш сайт*» укажите доменное имя вашего сайта на выбранном вами сервере. В отличие от сервиса *WebMoney Transfer*, здесь параметр *Success URL* программно, увы, изменить нельзя. Никакой PHP-страницы для скачивания создаваться не будет. После оплаты мы направим клиента на уже готовую страницу *muzic2a.php*, которую мы потом создадим (листинг 11.2).
6. В поле 6 – метод отсылки данных в *Success URL*. Укажите *POST*.
7. В поле *Fail URL* указывается путь к странице, на которую попадет посетитель в случае неудачного платежа. Эта страница у нас тоже находится в папке *muzic* и называется *muzic3.php* (листинг 10.4).
8. В поле 8 – метод отсылки данных в *Fail URL*. Укажите *POST*.

Далее нажимаете на кнопку «*Установить параметры*». Если все в порядке, то появится надпись «Данные успешно изменены».

Все, теперь нужно обратиться по e-mail: *support@roboxchange.com* или в техподдержку *ROBOXchange*, указав свой логин, который вы придумали еще при регистрации. Напишите, что нужно активировать магазин с таким-то логином. Через некоторое время на ваш e-mail придет сообщение, что ваш магазин активирован (если вы написали корректно свои данные при регистрации).

Итак, если вы намерены для приема платежей пользоваться только сервисом *Roboxchange*, то листинг 10.2 (файл *muzic.php*) нужно переделать (листинг 11.1).

Листинг 11.1 (листинг файла muzic.php, переделанный для приема платежей сервисом Roboxchange)

```
<html>
<head>
<title>Музыка</title>
<link rel="stylesheet" type="text/css" href="stili2.css">
</head>
<body>
<br>
<center><table border="2" bordercolor="cyan" id="lolo">
<tr><td>Хорошая музыка (цена всех композиции 10
руб.)</td></tr></center><tr><td>
<?php
$myz=opendir("./mp3");
while (($file=readdir($myz)) !==false)
{
 if($file !="." && $file !="..")
 {
 $i++;
 echo "$file<br>";
 }
}
closedir($myz);
// регистрационная информация
$mrh_login="alek-stroganov";        //логин
$mrh_pass1="gfhflj";                // пароль1
//параметры магазина
$inv_id=0;                  //номер счета
```

```
//описание покупки
$inv_desc="ROBOXchange Cash Register Advanced User Guide";
$out_summ="10.00";        //сумма покупки
$shp_item=2;              //тип товара
// формирование подписи
$crc=md5("$mrh_login:$out_summ:$inv_id:$mrh_pass1:shp_item=$shp_item");
echo "<form action='https://www.roboxchange.com/ssl/calc.asp' method=POST>
  <input type=hidden name=mrh value=$mrh_login>
  <input type=hidden name=out_summ value=$out_summ>
  <input type=hidden name=inv_id value=$inv_id>
  <input type=hidden name=inv_desc value=$inv_desc>
  <input type=hidden name=crc value=$crc>
  <input type=hidden name=shp_item value=$shp_item>
  <input type=submit value='Оплатить'>
  </form></html>";
?>
</td></tr></table>
</body>
</html>
```

Здесь мы просто выводим названия всех композиций из папки mp3 в один столбик таблицы. Тут все аналогично, как в листинге 10.2, но без формы и переключателей *type=checkbox*. Далее все по-другому.

$mrh_login="alek-stroganov"; – логин Продавца, т. е. ваш в *ROBOXchange*, т. е. тот самый логин, который вы вводили при регистрации в *ROBOXchange*;

$mrh_pass1="gfhflj"; – пароль #1, установленный Продавцом через интерфейс администрирования (см. рис. 11.10);

$inv_id=0; – номер заказа в магазине. У вас пока ноль заказов, поэтому и установите ноль;

$inv_desc="ROBOXchange Cash Register Advanced User Guide"; – описание заказа, можно использовать только английские или русские символы, цифры и знаки препинания. Максимальная длина 100 символов;

$out_summ="10.00"; – стоимость заказа в валюте Продавца. Допустим, вы хотите получить 10 рублей на свой кошелек за все композиции;

$shp_item=2; – тип товара. Можете поставить любое число;

$crc=md5("$mrh_login:$out_summ:$inv_id:$mrh_pass1:shp_item=$shp_item"); – контрольная сумма MD5 (подпись) – строка, представляющая собой 32-разрядное число в 16-теричной форме и любом регистре (всего 32 символа 0–9,

A-F). Формируется по строке, содержащей все обязательные параметры, разделенные двоеточием;

echo "<form action='https://www.roboxchange.com/ssl/calc.asp' method=POST> – форма для отправки указанных выше параметров методом *POST* на сайт для оплаты *https://www.roboxchange.com/ssl/calc.asp*. Все параметры передаются в скрытых полях;

<input type=hidden name=mrh value=$mrh_login>

<input type=hidden name=out_summ value=$out_summ>

<input type=hidden name=inv_id value=$inv_id>

<input type=hidden name=inv_desc value=$inv_desc>

<input type=hidden name=crc value=$crc>

<input type=hidden name=shp_item value=$shp_item>

<input type=submit value='Оплатить'> – вывод кнопки для отправки данных на сайт для оплаты;

</form>";

?>

</td></tr></table> – закрываем таблицу.

Результат исполнения этого скрипта на рис. 11.11.

Рис. 11.11

Теперь нам нужно составить страницу *muzic2a.php*, которую мы указали в качестве параметра *Success URL* (рис. 11.10). Там будут ссылки на mp3-файлы, которые может скачать себе посетитель после оплаты. Музыкальные файлы, если помните, находятся в папке *mp3*. В PHP-редакторе создайте новый файл, назвав его *muzic2a.php*.

Листинг 11.2 (файл muzic2a.php)

```
<html>
<head>
<title>Музыка</title>
```

```
<link rel="stylesheet" type="text/css" href="stili2.css">
</head>
<body>
<br>
<center><table border="2" bordercolor="cyan" id="lolo">
<tr><td>Спасибо за заказ</td></tr></center><tr><td>
<?php
$myz=opendir("./mp3");
while (($file=readdir($myz)) !==false)
{
 if($file !="." && $file !="..")
 {
 $i++;
 echo "<a href='mp3/$file' id='lolo'>$file</a><br>";
 }
}
closedir($myz);
?>
</body>
</html>
```

Разбирать листинг не буду. Вам все должно быть и так понятно. Результат работы листинга на рис. 11.12.

Рис. 11.12

Итак, после оплаты посетитель перейдет на страницу для скачивания. Созданные вами страницы *muzic.php* и *muzic2a.php* закачайте на свой хостинг в папку *muzic*.

Наберите в строке браузера адрес страницы *muzic.php* (*http://Ваш сайт/muzic/muzic.php*). Вы увидите все, как на рис. 11.11. Нажмите кнопку «*Оплатить*». Если ваш магазин (мерчант) в *ROBOXchange* был активирован, то попадете на страницу *https://www.roboxchange.com/ssl/calc.asp*, которая была указана в форме отправки (рис. 11.13).

Рис. 11.13

В поле «*Платите в:*» веберите Яндекс.Деньги. Оплату оставьте «*через веб-интерфейс*». В строке «Отдаете» будет указанна сумма, которую заплатит клиент. Десять рублей (или та сумма, которую вы укажете) пойдет в кошелек продавца. Небольшой процент (в нашем случае 78 копеек) данный сервис берет за услуги. В поле «*ваш E-mail*» введите созданный вами электронный адрес второго почтового ящика, откуда вы будете платить деньги. Нажмите на кнопку «*Оплатить*». Откроется следующее окно для оплаты (рис. 11.14), а затем, через несколько секунд, откроется окно для авторизации на Яндексе (рис. 11.15). Иногда на этом этапе может появиться форма с сообщением об ошибке (рис. 11.16). Дело в том, что данный платежный сервис пользуется большой популярностью, поэтому иногда из-за перегрузки возникают сбои. Если это случится, то ничего страшного. Ваши деньги не пропадут. Вернитесь на страницу *muzic.php* и повторите платеж.

После нажатия кнопки «*Авторизация*» (см. рис. 11.15), вам нужно будет ввести логин и пароль для того Яндекс-кошелька, откуда мы будем осуществлять платеж. Далее нужно будет ввести платежный пароль, который вы должны были получить при регистрации. После совершения платежа можно будет закрыть данное окно, и вы опять попадете на платежную страницу *ROBOXchange* (рис. 11.14), только там будет кнопка «*Продолжить*», при нажатии на которую вы попадете на следующую платежную страницу. Там будет написано о завершении платежа, а внизу будет кнопка «*Вернуться на сайт продавца*», при нажатии на которую мы должны попасть на страницу, указанную при регистрации в *ROBOXchange* (см. рис. 11.10) в поле *Success URL*. А там у нас страница для скачивания композиций *muzic2a.php* (см. рис. 11.12). Все, платеж совершен – товар получен.

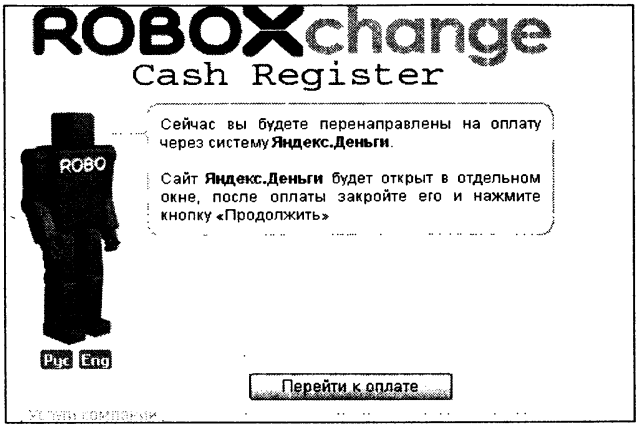


Рис. 11.14

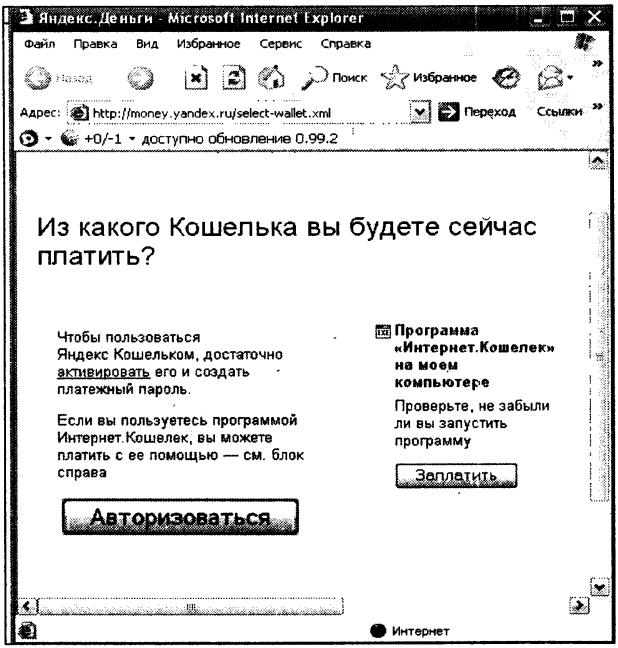


Рис. 11.15

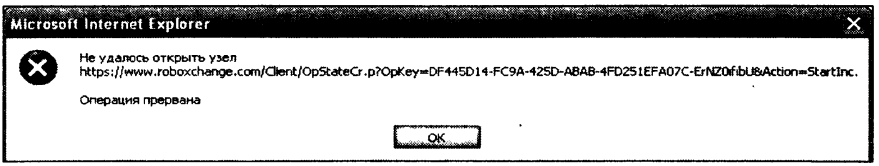


Рис. 11.16

Теперь, кстати, вы можете протестировать свой книжный магазин, который мы составляли в гл. 9. Введите в браузере: *http://Ваш сайт/magazin/admin.php*. После ввода пароля, который вы раньше придумали сами (рис. 9.5), вы попадете на страницу для ввода товара (рис. 9.7). Введите название и автора какой-нибудь книги, укажите стоимость книги, опишите книгу и введите какую-нибудь фотографию или любой другой графический файл формата gif или jpg. В общем, сделайте все так, как вы делали раньше на локальном сервере в гл. 9. Нажмите на кнопку «*Отправить*». Через несколько секунд или минут появится внизу надпись, что указанный вами графический файл закачан (если, конечно, не будет никаких ошибок и сбоев). Далее вернитесь в книжный магазин по адресу: *http://Ваш сайт/magazin/magazin.php* (см. рис. 9.1). Вы должны увидеть там свой новый товар. Теперь сделайте заказ нескольких экземпляров нескольких книг. После заказа вы должны попасть на страницу заказанных товаров (см. рис. 9.3). Описание заказа должно также через несколько минут поступить к вам на электронный адрес, который вы указали в функции *mail* (см. листинг 9.4 файла *magazin2.php*).

Если что-то у вас получилось не так, как надо, внимательно перечитайте гл. 9, 10, 11. Более подробно об организации платежей в системах *WebMoney Transfer* и ROBOXchange можно узнать на сайтах: *http://merchant.webmoney.ru* и *http://www.roboxchange.com*.

ЗАКЛЮЧЕНИЕ

Итак, при помощи данной книги вы создали сайт. Конечно, сайт в таком виде не очень привлекателен, так как он создавался исключительно для примера. Но используя полученные знания в этой книге, вы сможете создать сайт любой сложности и для любых целей. Здесь будет важна и ваша фантазия для создания привлекательного дизайна web-страниц, и ваше желание совершенствоваться в создании PHP-скриптов и приложений. Помимо этой читайте и другие книги по написанию PHP-скриптов, ставьте сами перед собой задачи и решайте их. Только так можно стать профессиональным php-программистом.

ОГЛАВЛЕНИЕ

e-mail:
dialog_mephi@mail.ru
http:
www.dialog-mifi.ru

издательство
ДИАЛОГ-МИФИ

Тел.: 320-30-77, 320-43-77,
факс: 320-31-33

Предлагает книги

Алгазин С. Д., Кондратьев В. В.
Программирование на Visual Fortran

Артёмов И. Л.
FORTRAN: основы программирования.

РАЗРАБОТКА БИЗНЕС-ПРИЛОЖЕНИЙ В ЭКОНОМИКЕ НА БАЗЕ MS EXCEL. Под ред. к. т. н. А. И. Афоничкина.

Ашихмин А. С.
Цифровая схемотехника. Шаг за шагом.

Бартеньев О. В.
1С:ПРЕДПРИЯТИЕ: программирование для всех.

Бартеньев О. В.
Microsoft Visual FoxPro. *Учебно-справочное пособие.*

Березин Б. И., Березин С. Б.
НАЧАЛЬНЫЙ КУРС С и С++.

Боресков А. В.
Графика трехмерной компьютерной игры на основе OPENGL.

Васильченко В. В.
FORTRAN. Программирование Windows-приложений на языке Fortran. Элементы управления и графика Windows.

Ватолин Д., Ратушняк А., Смирнов М., Юкин В.
МЕТОДЫ СЖАТИЯ ДАННЫХ. Устройство архиваторов, сжатие изображений и видео.

Вихнин А. Г., Сакипов Н. З.
ШТУРМ ЧЕТВЕРТОГО МЕГАПРОЕКТА: Кто будет новым Биллом Гейтсом?

Воробьев Е. М.
МАТЕМАТИКА. Введение в систему символьных, графических и численных вычислений.

Гробов И. Д.
ASP.NET 2.0. Теория и практика.

Гусева А. И. Учимся программировать: PASCAL 7.0.
Задачи и методы их решения

Гусева А. И.
УЧИМСЯ ИНФОРМАТИКЕ.

Дубейковский В. И. Практика функционального моделирования
с ALLFUSION PROCESS MODELER.

Дубейковский В. И. Эффективное моделирование
с ALLFUSION PROCESS MODELER.

Дунаев Сергей.
JAVA для Internet в Windows и Linux.

Епанешников А., Епанешников В.
Программирование в среде TURBO PASCAL 7.0.

Епанешников А., Епанешников В.
Локальные вычислительные сети.

Ковтанюк Юрий
Corel DRAW X3 на примерах.

Костров Б. В., Ручкин В. Н.
Архитектура микропроцессорных систем

Костров Б. В., Ручкин В. Н., Фулин В. А.
Искусственный интеллект и робототехника.

Куправа Т. А.
EXCEL. Практическое руководство.

Лавров К. Н., Цыплякова Т. П.
Финансовая аналитика. MATLAB 6.
(Под общ. ред. к. т. н. В. Г. Потемкина).

Лукин С. Н.
ТУРБО-ПАСКАЛЬ 7.0. ***Самоучитель для начинающих.***

Лукин С. Н.
VISUAL BASIC. ***Самоучитель для начинающих.***

Лукин С. Н.
WORD и WINDOWS. ***Самоучитель для начинающих.***

Лукин С. Н.
Понятно о VISUAL BASIC.NET. ***Самоучитель.***

Маклаков С. В. Моделирование бизнес-процессов
с ALLFUSION PROCESS MODELER (BPWIN 4.1).

Маклаков С. В. Создание информационных систем
с ALLFUSION MODELING SUITE.

Мещеряков В. В.
Задачи по математике с MATLAB®&SIMULINK®

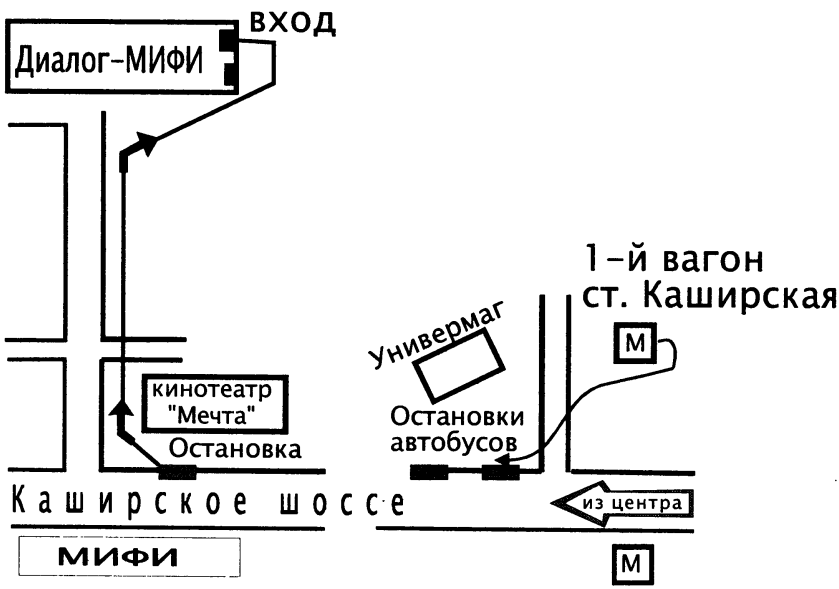
Михайлов А. В.
Электронная почта и ее защита.

Молочков В. П., Карпинский В. Б.
ОТ DELPHI 7 К DELPHI 2006.
Моргун А. Н.
MS WORD. Руководство к действию.
Пильщиков В. Н.
Программирование на языке АССЕМБЛЕРА.
Потемкин В. Г.
Вычисления в среде MATLAB.
Потемкин В. Г.
Инструментальные средства MATLAB 5.x.
Потемкин В. Г.
MATLAB 6: среда проектирования инженерных расчетов.
Рашевская М. А.
CORELDRAW. Практическое руководство.
Ручкин В. Н., Фулин В. А.
Архитектура компьютерных сетей
Скворцов В. И.
Технологические основы использования системы ARIS Toolset 7.0.
Труб И. И.
СУБД Caché: работа с объектами.
Туманов В. Е., Маклаков С. В.
Проектирование реляционных хранилищ данных.
Уваров А. С.
Программа P-CAD. Электронное моделирование
Федоров А., Елманова Н.
Введение в OLAP-технологии Microsoft.
Финогенов К. Г.
WIN 32. Основы программирования.
Фомичев В. М.
Дискретная математика и криптология.
Фролов А. В., Фролов Г. В.
ЯЗЫК С #. Самоучитель.
Хейфец А. Л.
Инженерная компьютерная графика. AUTOCAD.
Цисарь И. Ф., Нейман В. Г.
КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ЭКОНОМИКИ.
Шикин Е. В., Боресков А. В.
КОМПЬЮТЕРНАЯ ГРАФИКА. Полигональные модели.
Шумаков П. В. ADO.NET и создание приложений баз данных в среде Microsoft Visual Studio.NET.

Как нас найти

Адрес: **Москва, ул. Москворечье, дом 31, корп. 2**
Проезд:

м. Каширская *(первый вагон из центра)*,
авт. 95, 117, 275, 298, 738 (не экспресс), **740, 742**
и тролл. 71
до ост. "Институт МИФИ" (кинотеатр "Мечта")



Уважаемые читатели!

Если Вас заинтересовала какая-либо книга нашего издательства, то Вы можете сделать на нее заказ, и мы вышлем ее почтой на условиях *предоплаты*.

Обратите внимание – стоимость книг с доставкой ниже, чем в книжном магазине (см. прайс-лист www.dialog-mifi.ru).

Оформление заказа

Для получения книг Вам нужно отправить заявку одним из способов: на e-mail: **zakaz@dialog-mifi.ru** или **dialog_mephi@mail.ru**; по тел.: **(495) 320-30-77, 320-43-77**; по факсу: (495) **320-31-33**; почтой по адресу: **115409, г. Москва, ул. Москворечье, д. 31, корп. 2**. Издательство Диалог-МИФИ (*не забудьте вложить конверт с Вашим обратным адресом*).

В заявке обязательно должно быть указано:

- Фамилия, имя, отчество (полностью);
- Полный почтовый адрес с индексом;
- Наименование книги;
- Количество экземпляров;
- Телефон и e-mail указываются при наличии.

При получении заявки мы сообщим Вам о наличии книг, номер заказа и сумму оплаты. Только после получения нашего сообщения Вы должны оплатить заказ.

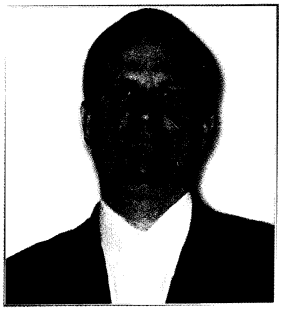
Наши реквизиты:

Название организации	ООО "Издательство Диалог-МИФИ"
Фактический адрес	115409, г. Москва, ул. Москворечье, д. 31, корп. 2
ИНН	7724582966
Расчетный счет (рублевый)	40702810600000003090
Наименование банка	АБ «Интерпрогрессбанк» (ЗАО)
БИК	044525402
Корр счет	30101810100000000402
Адрес банка	г. Москва, Старо-Каширское шоссе, д. 2, корп. 8

При оформлении бланка оплаты заказа в графе Назначение платежа должно быть указано: "Оплата за заказ №... и Ф. И. О."

Получение книг

Книги отправляются в течение трех рабочих дней со дня поступления денег за заказ на расчетный счет издательства.



PHP

Ваш первый сайт с использованием PHP-скриптов

Строганов Александр Сергеевич,
по специальности – инженер-физик.

Вы решили создать свой сайт, но не знаете с чего начать? Сделать это поможет данная книга. Вы познакомитесь с PHP программированием, научитесь легко и быстро создавать странички для своего сайта.

Бродя по просторам Интернета, Вам наверняка приходилось сталкиваться с сайтами, имеющими тысячи страниц разнообразного контента, интересного материала и тому подобное. Вы думаете, что эти страницы создаются вручную? Ничего подобного!

Создавать каждую страничку, используя только язык HTML невозможно, да и не нужно. С помощью PHP, Вы сможете сами наполнить свой сайт тысячами страниц, создав при этом вручную всего одну. Книга написана простым и понятным языком, который доступен даже новичку, никогда не слышавшему о PHP.

Освоив материал данной книги, Вы сможете создавать сайты любой сложности, писать любые PHP-скрипты для любых целей, используя всю Вашу фантазию.

ИЗДАТЕЛЬСТВО
ДИАЛОГ-МИФИ
Тел.: (495) 320-30-77, 320-43-77
Факс: (495) 320-31-33
Почта: dialog_mephi@mail.ru
http://www.dialog-mifi.ru