Дмитрий Ляпин
Александр Никитин

Начинаем с видеоуроков

Санкт-Петербург
«БХВ-Петербург»
2012

УДК 681.3.068
ББК 32.973.26-018.1
Л97

Ляпин, Д. А.

Л97 PHP — это просто. Начинаем с видеоуроков / Д. А. Ляпин, А. В. Никитин. — СПб.: БХВ-Петербург, 2012. — 176 с.: ил. + Видеокурс (на CD-ROM)

ISBN 978-5-9775-0678-6

В книге есть все, что необходимо начинающему веб-программисту, который собирается научиться создавать веб-сайты, соответствующие современным требованиям веб-разработки. Приведен краткий обзор языка HTML, необходимый для отображения содержимого веб-страниц. Подробно изложены принципы веб-программирования на языке PHP. Рассказано об использовании объектно-ориентированного подхода в программировании на PHP. Раскрыты вопросы работы с файлами, базами данных, обработки пользовательского ввода. Рассмотрен современный шаблон проектирования веб-приложений MVC. Подробно изложены принципы профессионального подхода к проектированию и реализации PHP-сценариев. В книге даны самые современные принципы веб-программирования, сделан упор на профессиональный подход к разработке и полный отказ от устаревших приемов.

Прилагаемый компкт-диск содержит видеоуроки для каждой главы.

Для начинающих веб-программистов

УДК 681.3.068
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Наталия Першакова*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*
Зав. производством	*Николай Тверских*

Подписано в печать 29.02.12.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 14,19.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

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

ISBN 978-5-9775-0678-6

Оглавление

Предисловие

Учебные материалы книги разделены на 13 глав, каждой из которых соответствует видеоурок. Для того чтобы ваше обучение было максимально эффективным, знакомство с новым материалом следует начинать с видеоурока, где авторы показывают реальные примеры использования излагаемой информации. Не волнуйтесь, если на данном этапе у вас появится какое-либо непонимание или объяснений из видео будет недостаточно. Пока ваша задача — просто повторить приведенный в видеоуроке пример, пускай даже до конца не понимая, как это работает. Все ответы ждут вас в теоретической части урока, в соответствующей главе книги. После ознакомления с теорией читателю предлагается закрепить полученные знания выполнением специального задания.

Таким образом, можно выделить три основных звена при изучении каждой главы:

- ❒ просмотр видеоурока и повторение практических действий, приведенных в нем;
- ❒ ознакомление с теорией, изложенной в главе;
- ❒ выполнение предложенного в конце главы задания.

Учебный курс ориентирован на месяц регулярной работы с данной книгой. Упор на практическое использование излагаемых материалов в совокупности с видеоиллюстрациями позволяет глубоко и качественно освоить знания, заложенные в книге. Данная методика освоения языков программирования активно применяется в центре компьютерного обучения "Школа программирования" (**http://proglive.ru**), который учрежден авторами книги. За годы использования данного подхода к обучению его эффективность прекрасно зарекомендовала себя и была подтверждена результатами нескольких сотен выпускников центра.

По результатам обучения читатели без какого-либо практического опыта веб-программирования получат все необходимые знания и умения для того, чтобы самостоятельно разрабатывать сайты на языке PHP и уверенно чувствовать себя на рынке труда веб-программистов.

Введение

Здравствуйте, дорогие читатели! Рады приветствовать вас на страницах этой книги. Давайте познакомимся. Нас зовут Александр Никитин и Дмитрий Ляпин. Мы являемся основателями центра компьютерного обучения "Школа программирования" (**http://proglive.ru**), по совместительству действующими программистами и авторами-разработчиками ряда других интернет-проектов.

Так сложилось, что практически всю нашу сознательную жизнь мы посвятили одному делу — программированию. На текущий момент мы занимаемся этим уже более 13 лет. За это время нам пришлось столкнуться с логическим и функциональным программированием, объектно-ориентированным и структурным программированием, системным и веб-программированием, параллельным программированием, программированием микроконтроллеров и искусственных интеллектов, программированием драйверов и компиляторов, низкоуровневым программированием в различных операционных системах.

Мы успели сменить не одну работу. Разрабатывали, внедряли, расширяли, делали сайты-визитки и крупные интернет-порталы. Прошли всё — от работы на рекламные агентства в роли HTML-верстальщиков до участия в проектах внедрения автоматизированных систем в крупнейших банках и компаниях федерального масштаба.

На текущий момент нам есть, что сказать о своей профессии. Мы прошли нелегкий путь, на котором было немало препятствий и ловушек. И нам искренне хочется облегчить эту дорогу для вас, наших читателей.

И это не пустые слова. Дело в том, что больше половины начинающих программистов бросают эту профессию из-за сложностей в обучении и высокой конкуренции на рынке. Начальный энтузиазм и легкость сменяются рутинным чтением справочников, от которых начинают "закипать мозги". Собственные программы, которые вы делаете для тренировки в ходе обучения, становятся все сложнее, и в какой-то момент вы сами перестаете понимать свой код. Все это приводит к потере интереса к обучению. Начинающий разработчик перестает развиваться в области программирования и начинает искать себя в чем-то другом. А ведь какие светлые и амбициозные чувства были у него в начале пути!

Действительно, обучение программированию можно сравнить с постоянным движением в гору. Каждый шаг будет даваться с трудом, каждую минуту вам нужно

прикладывать усилия для преодоления препятствий. Зато в какой-то момент вы окажетесь на вершине этой горы, и тогда для вас откроются невероятные возможности.

В этой книге мы постараемся максимально облегчить вашу дорогу к профессиональному использованию такого мощного языка веб-программирования как PHP. Кстати, почему именно PHP?

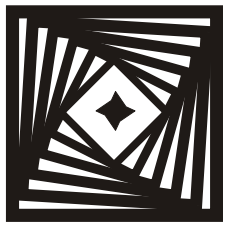
PHP — это самый популярный серверный язык в настоящее время, и это не случайно. Он прост в освоении, гибок и функционален. PHP используют как новички, так и профессионалы. Существует огромное количество готовых библиотек, написанных на PHP и призванных упростить разработку программистам. Можно сказать, что PHP в мире веб-программирования аналогичен английскому языку в языковой среде. Программируя на PHP, вы сможете пользоваться опытом и наработками огромного количества программистов по всему миру, а также ваш собственный код может быть полезным широкому кругу разработчиков.

Эта книга проведет вас от самых азов веб-разработки до приемов и концепций, которые применяются настоящими профессионалами. В первых уроках мы познакомимся с общими принципами функционирования Интернета и разберемся с тем, что же вообще собой представляют веб-страницы и сайты. Только освоив эту информацию, мы будем готовы перейти непосредственно к изучению языка PHP.

На протяжении всей книги мы особое внимание уделяем культуре написания кода. С нашей точки зрения культура кода имеет огромное значение для разработчиков, несмотря на то, что множество программистов даже не понимает этого словосочетания. Неосновательный подход веб-разработчиков к своему делу приводит к тому, что на данный момент для заказчика считается немалой удачей найти хорошего программиста. В этой связи программистам следует учиться как можно более качественно применять современные средства разработки в своей практике. В этом залог успеха нашей профессии. Именно поэтому последние главы книги мы посвятили не обучению правилам языка PHP, а методикам его грамотного использования.

Надеемся, что чтение этой книги и обучение PHP покажется вам интересным, и вы с удовольствием и энтузиазмом будете применять полученные знания на практике. Успехов!

Александр Никитин, Дмитрий Ляпин



УРОК 1

Основы протокола HTTP

Прежде чем перейти к изучению такого популярного и мощного языка веб-программирования, как PHP, важно иметь четкое представление о том, *как* вообще функционирует Интернет. Что собой представляют PHP-сценарии и HTML-код? Как взаимодействуют наши компьютеры с серверами, на которых располагаются веб-страницы? И, наконец, как эти замысловатые сценарии превращаются в те прекрасные интернет-сайты, которые мы каждый день видим на экранах своих мониторов.

К сожалению, огромное количество начинающих PHP-программистов даже не отдает себе отчет в том, где выполняются написанные ими сценарии, и вообще, что собой представляет процесс выполнения PHP-сценария. Однако без этих знаний очень трудно достичь каких-либо существенных успехов в PHP-разработке. Более того, эти знания должны быть получены в первую очередь.

Представьте себе архитектора, который абсолютно не разбирается в строительных материалах. Можно ли назвать его настоящим профессионалом? Скорее всего, такой человек не сможет построить что-нибудь сложнее сарая. Так и веб-программист, не понимающий основных принципов интернет-взаимодействия, просто не способен решать более-менее серьезные задачи.

Поэтому данный урок нашей книги мы посвящаем базовым вопросам взаимодействия веб-серверов, на которых располагаются интернет-страницы, и клиентских программ для просмотра этих самых страниц. Но если вы думаете, что вас ожидает лишь важная, но скучная теория, то ошибаетесь. В конце мы предложим пару практических заданий, которые позволят еще лучше понять принципы клиент-серверного взаимодействия программ при загрузке веб-страниц.

Что происходит между тем, как вы набрали адрес сайта и увидели страницу на экране?

Все мы знакомы с браузерами. Это те самые программы, которые позволяют нам просматривать страницы в Интернете. Но если вы спросите десять программистов, как работает веб-обозреватель, то восемь из них не смогут дать внятного ответа.

Давайте попробуем самостоятельно разобраться, что же происходит в промежутке между запросом какого-либо сайта и отображением страницы на экране.

Рассмотрим адрес какой-нибудь реальной веб-страницы, например **http://prog-school.ru/catalog**. На рис. 1.1 приведено ее отображение в браузере.

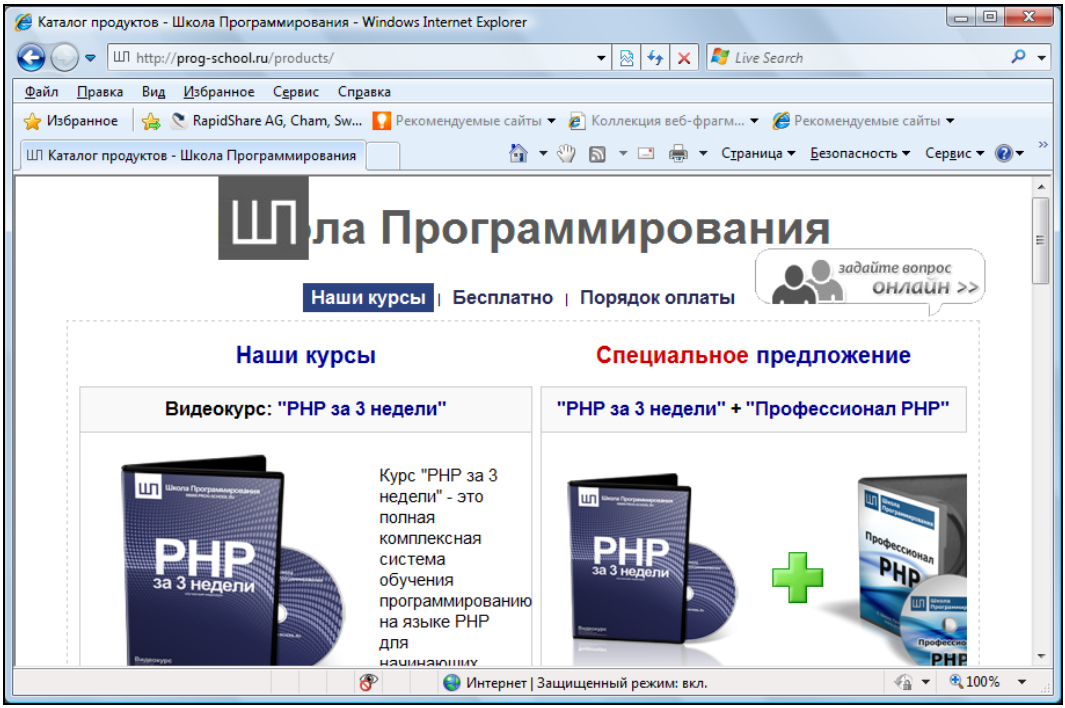


Рис. 1.1. Отображение в браузере страницы **http://prog-school.ru/catalog**

Обратите внимание на строку в адресном поле: **http://prog-school.ru/products/**. Она однозначно идентифицирует веб-страницу, которую должен отобразить для вас браузер. Эту строку также называют URL или URI. Давайте подробнее разберем обе эти аббревиатуры.

URL и URI

URL (Uniform Resource Locator) — единообразный локатор (определитель местонахождения) ресурса. URL — это стандартизированный способ записи адреса ресурса в Интернете.

URI (Uniform Resource Identifier) — унифицированный (единообразный) идентификатор ресурса. URI — это последовательность символов, идентифицирующая абстрактный или физический ресурс.

URI — это более общее понятие, нежели URL. URI не всегда указывает то, как получить ресурс, в отличие от URL, а только идентифицирует его. URL — это URI,

который, помимо идентификации ресурса, предоставляет еще и информацию о местонахождении этого ресурса. Действительно, любой URL-адрес несет достаточно информации для точного нахождения страницы. Далее в этой книге при использовании адресов сайтов будем придерживать аббревиатуры URL.

Структура адреса сайта

Вернемся к URL-адресу **http://prog-school.ru/catalog**. Его можно разделить на 3 части:

- **http://**;
- **prog-school.ru**;
- **/catalog**.

Первая часть адреса (**http://**) определяет протокол взаимодействия браузера с сервером. В нашем случае это протокол HTTP, о нем речь пойдет далее.

Вторая часть адресной строки называется *доменом* и служит для идентификации конкретного сайта с помощью службы DNS. DNS (Domain Name System, система доменных имен) — компьютерная распределенная система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства). В сети существует большое количество DNS-серверов, которые по доменному имени ресурса могут "подсказать" его реальное местоположение, определяемое IP-адресом.

Третья часть адресной строки определяет подзапрос к сайту. Это может быть путь до определенного файла или каталога на сервере. Здесь же могут быть заданы различные параметры HTTP-запроса.

Исходный код HTML-страницы

Теперь давайте разберем, что же получает браузер в ответ на сформированный HTTP-запрос. Для этого еще раз взгляните на рис. 1.1. Как видно, страница может состоять из текста, картинок, гиперссылок, полей ввода, кнопок и других элементов. Информация обо всем этом была передана от веб-сервера браузеру, который и сгенерировал конечный внешний вид страницы. Передаваемые данные описываются с помощью протокола HTML.

HTML (HyperText Markup Language, язык разметки гипертекста) — стандартный язык разметки документов в Интернете. Язык HTML интерпретируется браузером и отображается в виде документа в удобной для человека форме.

Можно сказать, что браузеры выполняют две основные функции — это взаимодействие с веб-серверами посредством HTTP-запросов, а также преобразование полученного от сервера HTML-кода в визуальное представление.

Схема клиент-серверного взаимодействия при работе браузера

Итак, давайте теперь ответим на главный вопрос этого урока: что же происходит в промежутке между тем, как мы набрали адрес сайта в браузере, и моментом, когда его содержимое отобразилось на экране.

После того как URL сайта введен в адресную строку и нажата клавиша <Enter>, браузер формирует пакет данных, который посылает по сети. Этот пакет содержит URL запрашиваемого сайта, а также прочие данные запроса, оформленные согласно протоколу HTTP. Содержание пакета нас сейчас не интересует, важно то, что переданный URL позволяет промежуточным узлам в Интернете доставить наш пакет с HTTP-запросом по адресу до нужного сервера.

На физическом веб-сервере (т. е. некоем реальном компьютере) должна быть запущена соответствующая программа, которая также называется *веб-сервером* и служит для обработки входящих HTTP-запросов. Самым популярным веб-сервером на данный момент является программа Apache, логотип которой приведен на рис. 1.2.

После получения пакета с HTTP-запросом веб-сервер определяет, какие действия необходимы для его обработки. Если HTTP-запрос осуществляется к обычной HTML-странице, то веб-сервер просто передает ее содержимое браузеру. Если же HTTP-запрос осуществляется к какому-либо сценарию (например, PHP-сценарию), веб-сервер передает запрос на обработку соответствующей программе, отвечающей за обработку этого типа сценариев. Обработчик сценария в свою очередь может вызывать другие программы в ходе своей работы, например СУБД MySQL.

Результатом работы обработчика сценариев является HTML-код, который веб-сервер посылает обратно на компьютер пользователя. Сгенерированный HTML-код веб-сервер упаковывает в HTTP-пакет, который и передается по сети обратно клиенту. Полученный HTTP-ответ попадает в браузер клиента, который извлекает из него HTML-код и генерирует на его основе графическое представление запрошенной страницы.

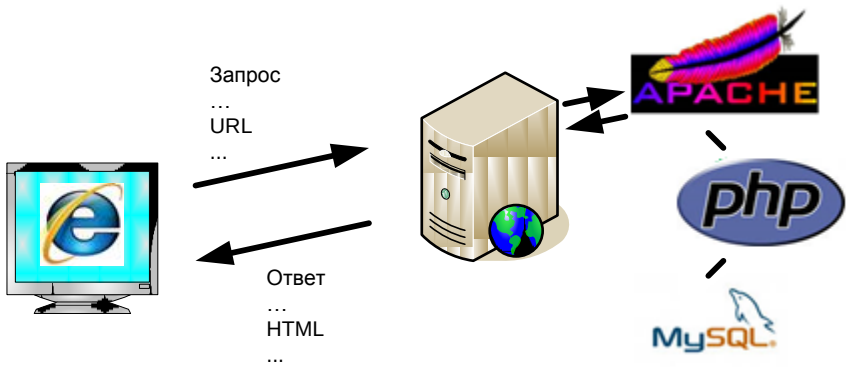


Рис 1.2. Схема взаимодействия браузера и веб-сервера

Протокол HTTP

Давайте дадим более четкое определение протоколу HTTP и разберемся, зачем он нужен.

HTTP (HyperText Transfer Protocol, протокол передачи гипертекста) — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов). Основой HTTP является технология "клиент-сервер", т. е. предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом. HTTP в настоящее время повсеместно используется во Всемирной паутине для получения информации с веб-сайтов.

Иными словами, HTTP — это язык, на котором общаются браузер и веб-сервер при обмене пакетами с данными. Обмен сообщениями в ходе HTTP-соединения идет по обыкновенной схеме "запрос—ответ"

Каждое HTTP-сообщение состоит из трех частей, которые передаются в указанном порядке:

1. Стартовая строка — определяет тип сообщения.
2. Заголовки — характеризуют тело сообщения, параметры передачи и прочие сведения.
3. Тело сообщения — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Заголовки и тело сообщения могут отсутствовать, но стартовая строка является обязательным элементом, т. к. указывает на тип запроса/ответа.

Стартовые строки HTTP-запросов и HTTP-ответов различны. Для HTTP-запроса стартовая строка имеет следующий вид:

```
МЕТОД URI HTTP/Версия
```

Здесь:

- `МЕТОД` — название запроса, одно слово прописными буквами. В версии HTTP 0.9 использовался только метод GET, список запросов для версии 1.1 представлен далее;
- `URI` определяет путь к запрашиваемому документу;
- `Версия` — пара разделенных точкой арабских цифр. Например: `1.0`. Определяет версию используемого HTTP-протокола. На данный момент наиболее актуальной является версия 1.1.

Для HTTP-ответа стартовая строка выглядит следующим образом:

```
HTTP/Версия КодСостояния Пояснение
```

Здесь:

- `Версия` — пара разделенных точкой арабских цифр как в запросе;
- `КодСостояния` — три цифры. По коду состояния определяется дальнейшее содержимое сообщения и поведение клиента;

- `Пояснение` — текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным.

Заголовки HTTP

Заголовки HTTP (HTTP Headers) — это строки в HTTP-сообщении, содержащие разделенную двоеточием пару "параметр: значение". Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.

Примеры заголовков:

```
Server: Apache/2.2.11 (Win32) PHP/5.3.0
Last-Modified: Sat, 16 Jan 2010 21:16:42 GMT
Content-Type: text/plain; charset=windows-1251
Content-Language: ru
```

В приведенном примере каждая строка представляет собой один заголовок. При этом то, что находится до первого двоеточия, называется *именем* (name), а что после нее — *значением* (value).

Все заголовки разделяются на четыре основных группы:

- General Headers (Основные заголовки) — должны включаться в любое сообщение клиента и сервера;
- Request Headers (Заголовки запроса) — используются только в запросах клиента;
- Response Headers (Заголовки ответа) — только для ответов от сервера;
- Entity Headers (Заголовки сущности) — сопровождают каждую сущность сообщения.

Именно в таком порядке рекомендуется посылать заголовки получателю. Заголовки запроса и ответа, как и основные заголовки, описывают все сообщение в целом и размещаются только в начальном блоке заголовков, в то время как заголовки сущности характеризуют содержимое каждой части в отдельности, располагаясь непосредственно перед ее телом.

Мы не будем углубляться в изучение форматов HTTP-сообщений. На данном этапе достаточно иметь общее представление о функционировании механизмов, обеспечивающих клиент-серверное взаимодействие между браузерами и веб-серверами. Дополнительную информацию о HTTP-заголовках вы сможете найти в Википедии: **http://ru.wikipedia.org/wiki/Заголовки_HTTP**.

Как посмотреть данные HTTP-запроса и HTTP-ответа в браузере

Давайте подробнее рассмотрим содержимое пакетов, которыми обмениваются браузер и веб-сервер. Для этого нам понадобится специальная программа, которая позволит просматривать передаваемые данные. Для удобства использования будем применять к браузеру FireFox плагин, который называется Firebug. Его можно скачать по адресу **http://firebug.ru**. Если же вы пользуетесь другим веб-обозревателем,

то вам подойдет другой продукт той же компании под названием Firebug Lite (**http://getfirebug.com/firebuglite**).

После установки плагина в браузере в разделе **Инструменты** главного меню появится новый пункт **Firebug**. Выберем его, а затем в раскрывшемся подменю — пункт **Открыть Firebug**, после чего увидим окно плагина (рис. 1.3).

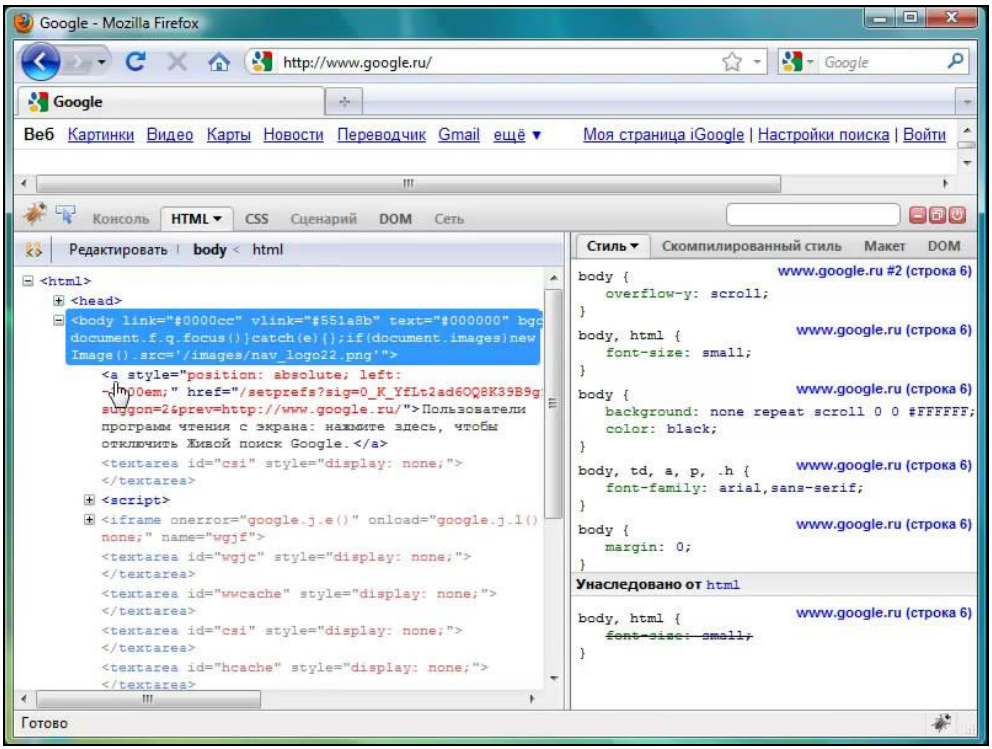


Рис. 1.3. Плагин Firebug

На вкладке **HTML** вы сможете увидеть HTML-код загруженной страницы. Обратите внимание, Firebug позволяет скрывать и раскрывать содержание отдельных элементов, из которых состоит HTML-страница.

Но нас в данный момент больше интересует содержание всего HTTP-пакета, нежели его HTML-составляющей. Чтобы увидеть непосредственное содержимое HTTP-запросов и ответов перейдем на вкладку **Сеть** (рис. 1.4). По умолчанию эта функциональность отключена. Для того чтобы ее включить, нажмите на стрелку рядом с надписью "Сеть" и выберите пункт **Панель включена**. После чего давайте откроем любую интернет-страницу и посмотрим на окно плагина Firebug.

Теперь в окне Firebug мы видим список запросов, которые совершил наш браузер. Обратите внимание, что инициатором взаимодействия между браузером и веб-сервером всегда является именно браузер. Веб-сервер же просто формирует ответы на запросы браузеров клиентов. Также обратите внимание: несмотря на то, что мы запросили всего одну веб-страницу, браузер выполнил несколько HTTP-запросов. Это связано с тем, что некоторые элементы страницы, такие как картинки, таблицы

стилей (css), JavaScript-файлы, подгружаются в отдельных запросах. Иногда для загрузки одной веб-страницы браузер выполняет более сотни запросов.

Если раскрыть любой из пунктов, щелкнув по значку +, мы увидим данные соответствующего HTTP-запроса. Они поделены на четыре категории: **Параметры**, **Заголовки**, **Ответ**, **Кэш**. Параметры у запроса могут отсутствовать, поэтому одноименной вкладки вы можете не увидеть. На закладке **Заголовки** представлены HTTP-заголовки запроса браузера и ответа сервера. Вкладка **Ответ** содержит HTML-код, полученный браузером от сервера. А на вкладке **Кэш** можно найти служебную информацию по кэшированию страницы.

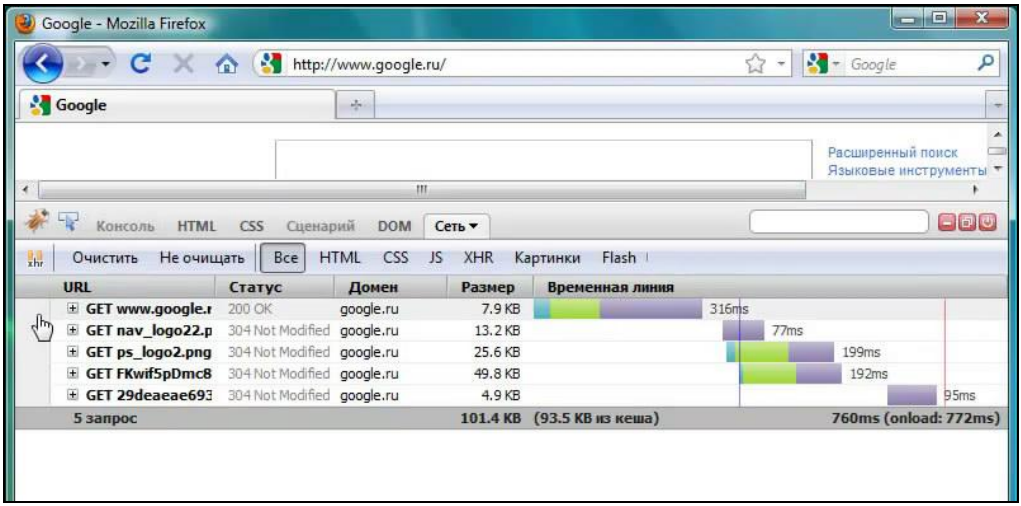


Рис. 1.4. Плагин Firebug, вкладка **Сеть**

Самостоятельно откройте несколько интернет-страниц и просмотрите отправленные и полученные данные. Скорее всего, большинство из них вам сейчас незнакомо, однако в этом нет ничего страшного. На данном этапе достаточно просто понимать, что происходит в процессе информационного обмена между браузером и веб-сервером.

Как скачать сайт утилитой Telnet

Чтобы закрепить полученные знания о протоколе HTTP, давайте познакомимся с еще одной утилитой, которая позволит трассировать (просматривать содержимое) HTTP-коммуникации. Эта утилита называется Telnet.

Строго говоря, TELNET (TErminaL NETwork) — это сетевой протокол для реализации текстового интерфейса по сети (в современной форме — при помощи транспорта TCP). Однако название "telnet" имеют также некоторые утилиты, реализующие клиентскую часть протокола.

HTTP как раз является текстовым протоколом. То есть HTTP-запросы и HTTP-ответы имеют понятный человеку вид, их можно читать в обычном текстовом ре-

дакторе. С помощью Telnet-клиента мы сможем подключиться к какому-нибудь веб-серверу, самостоятельно сформировать и отправить HTTP-запрос и увидеть содержимое HTTP-ответа.

Для запуска Telnet в ОС Windows выполните:

1. Нажмите кнопку **Пуск**.
2. В строке поиска введите `cmd` и нажмите клавишу <Enter>.
3. В консоли наберите `telnet` и нажмите клавишу <Enter>.

Перед вами появится окно программы Telnet-клиента (рис. 1.5).

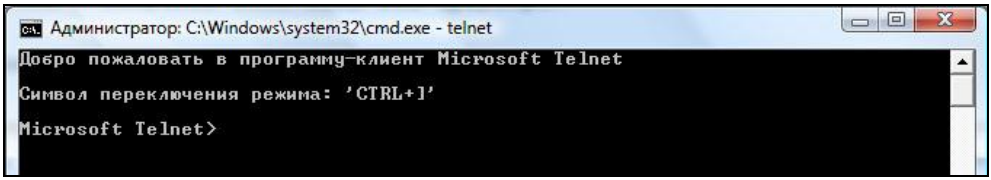


Рис. 1.5. Telnet-клиент Windows

Если же вы увидели окно с сообщением о том, что telnet является незнакомой командой для Windows, выполните следующие действия:

1. Откройте Панель управления.
2. Выберите **Программы и компоненты**.
3. Справа в окне найдите ссылку **Включение или отключение компонентов Windows** и щелкните по ней.
4. В окне **Компоненты Windows** в списке найдите **Клиент Telnet**, поставьте напротив флажок и нажмите кнопку **OK**.

Теперь, когда Telnet успешно включен, можно переходить к основным действиям. В тренировочных целях попробуем подключиться к веб-серверу Яндекса, который использует домен **ya.ru**. Однако знания одного лишь домена не достаточно для взаимодействия с веб-сервером.

Дело в том, что на одной машине в сети могут располагаться сразу несколько сервисов, которые ждут входящих сетевых соединений. Например, помимо веб-сервера на компьютере может быть запущена служба почтового сервера, FTP-сервер и т. п. Каким же образом сервер определяет, какая из программ должна обработать то или иное соединение? Для решения этой задачи используется понятие порта. *Порт* — это абстракция, характерная для проколов сетевого уровня TCP и UDP, которая представляет собой номер, выделяемый сетевому приложению для связи с приложениями на других хостах. Таким образом, за каждым сетевым приложением закреплен уникальный порт, т. е. некий номер, который фигурирует в передаваемых пакетах с данными и позволяет однозначно идентифицировать приложение, которому адресован каждый конкретный пакет.

Веб-серверы обычно используют порт с номером 80. Поэтому для соединения с Яндексом введите следующую строчку и нажмите клавишу <Enter> (рис. 1.6):

```
telnet ya.ru 80
```

Рис. 1.6. Telnet-соединение с веб-сервером

Если вы увидите пустой черный экран и мигающий курсор — все в порядке. Подключение прошло успешно, и сервер ждет вашего запроса.

Теперь давайте попробуем получить от сервера главную страницу сайта **ya.ru**. Для этого нам потребуется передать стартовую строку HTTP-соединения, а также необходимые HTTP-заголовки.

Для начала введите стартовую строку и нажмите клавишу <Enter>:

```
GET / HTTP/1.1
```

Пусть вас не смущает, что ничего не отображается на экране, это особенность Telnet-клиента в Windows. Из данной стартовой строки видно, что мы используем для HTTP-запроса метод GET, хотим получить корневой элемент сайта, о чем говорит URI-путь "/", а также используем HTTP-протокол версии 1.1.

Теперь введите HTTP-заголовок:

```
Host: www.ya.ru
```

Несмотря на то, что самому Telnet-клиенту мы уже указали, что соединение должно быть установлено с хостом **ya.ru**, требования HTTP-протокола обязывают задавать заголовок `Host` в каждом HTTP-запросе. Иначе в ответ мы получим ошибку "Bad request".

Теперь *дважды* нажмите клавишу <Enter>. Двойное нажатие требуется из-за того, что помимо заголовков сообщения мы также можем задать его тело. Но в нашем случае тело сообщения пустое, и мы просто еще один раз нажимаем клавишу <Enter>.

После второго нажатия Telnet-клиент выполнит HTTP-запрос к указанному серверу и получит HTTP-ответ. В нашем случае вы должны увидеть на экране результат, похожий на тот, что представлен на рис. 1.7.

Как видно из рисунка, HTTP-ответ веб-сервера состоит из стартовой строки, семи HTTP-заголовков и HTML-кода страницы.

Если вы все сделали правильно, то стартовая строка HTTP-ответа будет иметь вид:

```
HTTP/1.1 200 OK
```

`200` — это код состояния HTTP-ответа, `OK` — пояснение кода состояния. Данная строчка говорит о том, что веб-сервер успешно выполнил обработку нашего HTTP-запроса.

Таким образом, мы только что научились запрашивать и получать привычные нам веб-страницы не с помощью браузера, а через Telnet-клиент, который отображает данные HTTP-ответа веб-сервера.

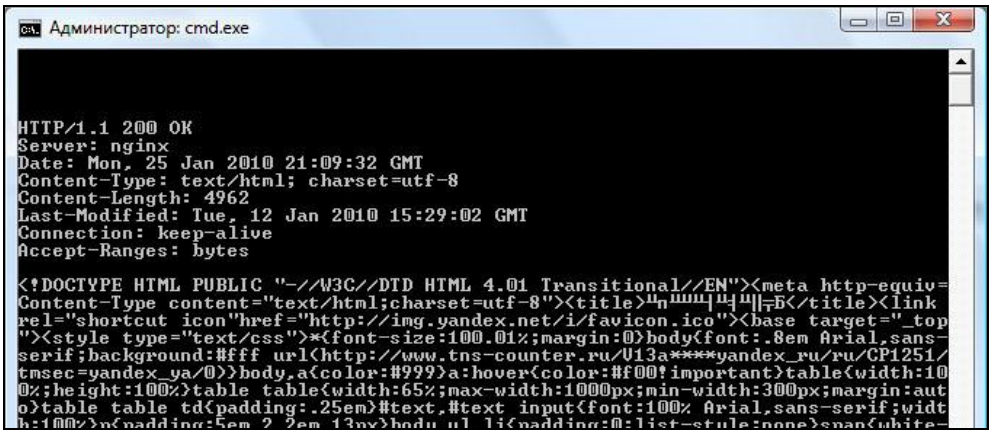


Рис. 1.7. HTTP-ответ от хоста ya.ru

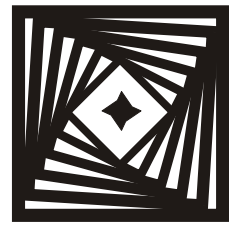
Резюме

После прочтения данного урока и повторения практических примеров вы должны четко понимать, как взаимодействуют браузер и веб-сервер при запросе той или иной интернет-страницы. Если вы еще не выполнили действия, которые мы описали по ходу этого урока, обязательно сделайте их. Только повторенные на собственной практике знания можно считать окончательно усвоенными. А знания этого урока — это важнейший фундамент всего дальнейшего изучения не только PHP, но и других технологий и языков веб-программирования.

Также хотим обратить ваше внимание на огромную роль самообразования для любого программиста. Каким бы не был объем книги, она не сможет включить в себя всех знаний по теме веб-разработки. Поэтому возьмите себе за правило самостоятельно более глубоко изучать интересующие вас моменты из каждого урока. Возможно, в этом уроке вас заинтересовало значение различных HTTP-заголовков, которые вы могли наблюдать в Telnet-клиенте или Firebug. Не ленитесь, найдите информацию о них в Интернете, это вовсе не сложно. Как показывает практика, наибольших успехов в программировании добиваются те, кто с удовольствием занимается самообразованием.

Задания

1. Скачайте и установите для своего браузера плагин, позволяющий просматривать данные HTTP-запросов и ответов, например Firebug. Выполните несколько вызовов веб-страниц с включенным плагином. Изучите отправленные и принятые данные.
2. Скачайте какую-нибудь веб-страницу с помощью Telnet-клиента. Откройте ту же страницу в браузере и просмотрите ее исходный код. Совпадает ли он с тем, что вы увидели в Telnet-клиенте. Почему? Постарайтесь обосновать ответ.



УРОК 2

Подготовка рабочего места

В этом небольшом уроке мы наконец-то закончим подготовку к изучению PHP-программирования и перейдем от слов к делу, написав свой первый PHP-сценарий. Но прежде чем это сделать, нам необходимо установить пакет программ, которые обеспечат удобную разработку на протяжении всего курса обучения.

Как вы уже знаете из первого урока, для функционирования PHP-сценариев необходим веб-сервер и PHP-интерпретатор. Давайте займемся установкой этих программ на локальный компьютер, чтобы мы смогли разрабатывать и тестировать сайты без необходимости их размещения в Интернете. Запомните, что любой разработчик сначала работает с локальной версией сайта или сценария и лишь после успешного тестирования выкладывает новые версии сценариев на реальный сайт.

При серьезной разработке принято организовывать сразу три площадки: для разработки, для тестирования и "боевую", т. е. реально функционирующую систему. Однако для наших целей хватит и одной площадки — локального компьютера, где мы и будем проводить все эксперименты с нашими PHP-сценариями.

Установка пакета Денвер

После изучения *урока 1* вы должны знать, что для полноценной разработки нам потребуется установить как минимум веб-сервер и PHP-интерпретатор. А также, что очень желательно, какую-нибудь СУБД (систему управления базами данных), чтобы мы могли пользоваться функционалом работы с базой данных в своих сценариях.

Мы же предлагаем заменить разрозненную установку вышеперечисленных программ установкой всего одной программы, а точнее пакета программ под названием Денвер. Здесь нет никакого чуда, Денвер — это просто программа-инсталлятор и настройщик всех необходимых инструментов для локальной веб-разработки. Он незаменим на первых этапах обучения веб-программированию, когда ученик еще не обладает глубоким пониманием механизмов функционирования веб-сервера и его компонентов.

Бытует мнение, что Денвер плох тем, что не позволяет провести тонкую настройку компонентов веб-сервера. Это мнение ошибочно, т. к. все что делает Денвер — это

инсталлирует необходимые компоненты и настраивает их взаимодействие. Вы всегда сможете внести необходимые корректировки в настройки любой из составляющих системы, будь то веб-сервер, PHP-интерпретатор или СУБД.

Пакет Денвер позволяет единовременно установить на своем локальном компьютере такие необходимые программы, как веб-сервер Apache, интерпретатор PHP и СУБД MySQL. Стоит заметить, что это наиболее популярная связка программ, которая используется для работы миллионов реально существующих сайтов.

Итак, скачать этот замечательный установщик пакета программ можно по адресу: **http://www.denwer.ru/** (рис. 2.1).

Рис. 2.1. Сайт **denwer.ru**

При нажатии кнопки **Скачать Денвер** вам предложат выбрать версию PHP. Советуем выбирать самую последнюю версию дистрибутива (рис. 2.2).

Рис. 2.2. Выбор версии PHP

После этого нажмите кнопку **Скачать** и в появившейся форме введите свои контактные данные.

По указанному адресу электронной почты вы получите письмо со ссылкой для скачивания дистрибутива. Скачайте предложенный файл, сохраните в любое удобное место на локальном компьютере и запустите его. Сначала вы увидите окно распаковки архива (рис. 2.3), а после окно с предложением начать установку Денвера (рис. 2.4).

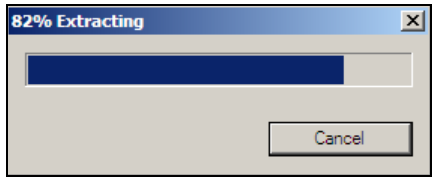


Рис. 2.3. Распаковка архива Денвера

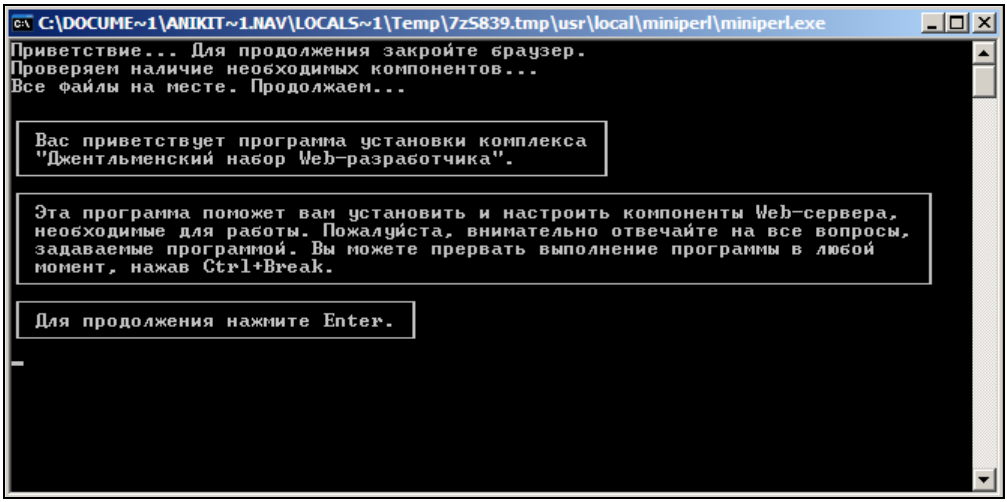


Рис. 2.4. Окно установки Денвера

Последовательно ответьте на вопросы установщика. Рекомендуем при вашей первой установке Денвера по возможности оставлять значения по умолчанию. Если в процессе инсталляции у вас возникли какие-либо сложности, советуем обратиться к видеоуроку с нашего сайта, в котором подробно описан процесс установки:

http://prog-school.ru/2010/02/kak-ustanovit-veb-server-i-napisat-pervyj-php-skript/

После успешного завершения установки вы должны увидеть окно браузера (рис. 2.5).

На своем рабочем столе вы найдете три ярлыка (рис. 2.6).

Это своеобразный пульт управления вашим веб-сервером. Ярлык **Start Denwer** позволяет запустить все программы веб-сервера, **Stop Denwer** — завершить эти программы, **Restart Denwer** — перезапустить их.

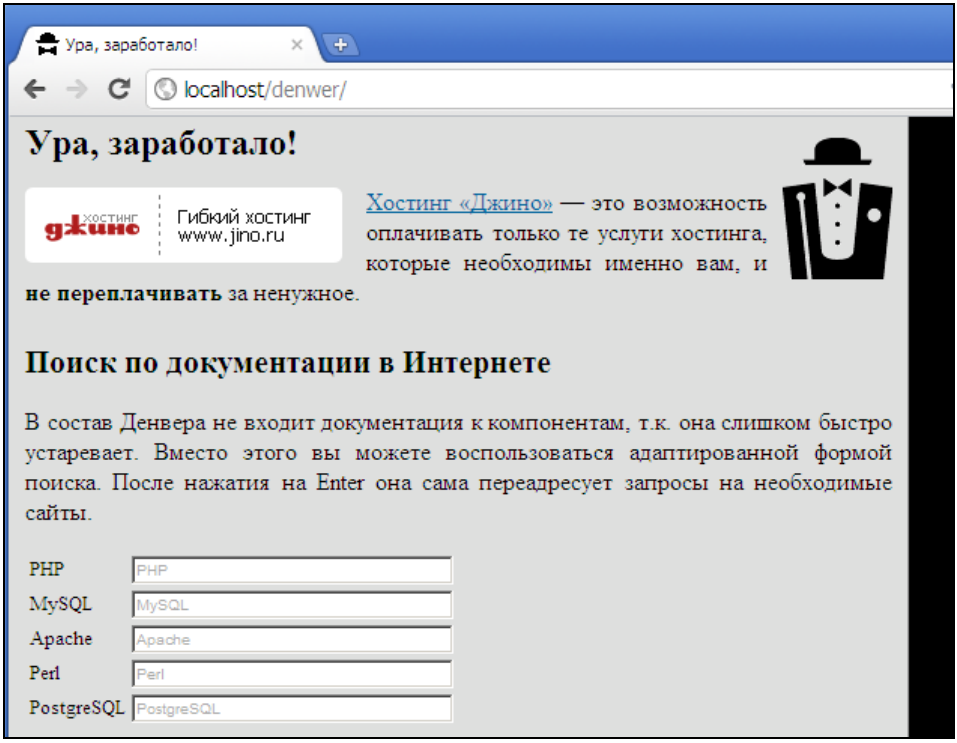


Рис. 2.5. Завершение установки Денвера

Рис. 2.6. Ярлыки Денвера

Структура папок Денвера

Прежде чем приступить к изучению архитектуры каталогов нашего веб-сервера, перейдите на рабочий стол и дважды щелкните по ярлыку **Start Denwer**. Вы должны увидеть следующее окно, подобное представленному на рис. 2.7.

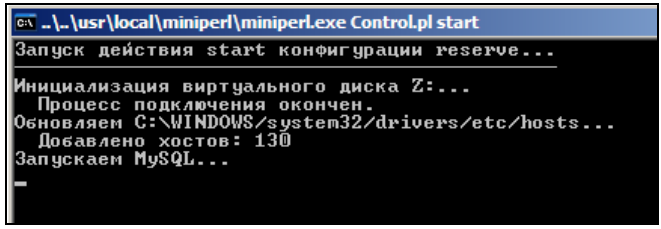


Рис. 2.7. Запуск Денвера

Теперь давайте разберемся, как устроена архитектура каталогов Денвера и куда нам в будущем потребуется выкладывать свои файлы сценариев, чтобы увидеть их работу на локальном компьютере.

Обратите внимание, что после запуска Денвер должен был создать виртуальный диск на вашем компьютере. При установке вы задавали его букву. По умолчанию это была буква "Z". Зайдите в окно **Компьютер** и убедитесь, что такой диск появился (рис. 2.8). Откройте содержимое этого диска. Вы должны увидеть перечень каталогов, как показано на рис. 2.9.

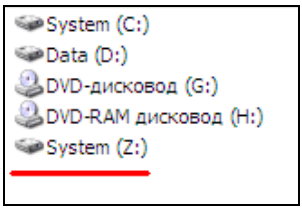


Рис. 2.8. Виртуальный диск веб-сервера

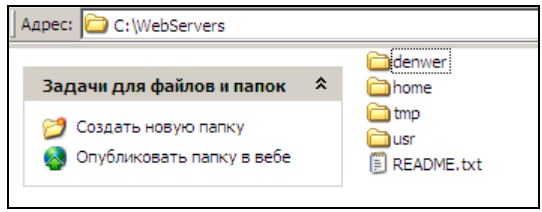


Рис. 2.9. Содержимое диска веб-сервера

Обратите внимание, что созданный диск не является физическим, это лишь виртуальное представление обычной папки на компьютере, необходимое для функционирования веб-сервера.

При установке Денвер просил выбрать вас папку на жестком диске, где будут располагаться все компоненты веб-сервера. По умолчанию был предложен каталог C:\WebServers (рис. 2.10). Давайте зайдем в этот каталог либо в ту папку, которую вы указали в качестве требуемого пути в ходе установки. В ней вы найдете все то же содержимое нашего виртуального диска.

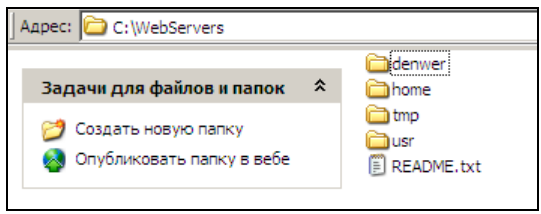


Рис. 2.10. Содержимое папки C:\WebServers

Вам должно быть ясно, что каталог C:\WebServers и виртуальный диск Z: — это фактически одно и то же, а не две копии одной и той же папки.

Мы не будем углубляться в изучение архитектуры папок, в которых находятся файлы веб-сервера Apache, интерпретатора PHP, СУБД MySQL и т. д. На начальном этапе нас интересует принцип создания сайтов на локальном компьютере.

Главный интерес для нас представляет папка home. В ней мы и будем располагать будущие сайты. Для каждого нового сайта будем использовать отдельный каталог внутри папки home. При этом обратиться к новому сайту в браузере мы сможем по адресу **http://<*название_подкаталога_в_папке_home*>.**

Обратите внимание, что на конце адреса нет привычного ".ru" или ".com". Это отличие наших локальных сайтов. Для обращения к ним мы не должны добавлять в конце адреса дополнительные домены (например, ".ru").

Предположим, что мы завели в папке home подкаталог с именем helloworld. Это значит, что к страницам нашего нового локального сайта мы будем обращаться следующим образом: **http://helloworld/*<путь_внутри_сайта>*.**

Все сценарии и статичные HTML-страницы локального сайта требуется располагать внутри подкаталога www нашего локального сайта. Это связано с настройками маршрутизации Денвера. На данном этапе достаточно просто запомнить это правило.

Таким образом, если мы хотим создать станицу index.html для нашего сайта helloworld, то на жестком диске мы должны расположить ее по адресу C:\WebServers\home\helloworld\www (если не менялись пути по умолчанию). А обратиться в браузере к такой странице можно по адресу **http://helloworld/index.html**.

Предлагаем вам прямо сейчас создать предложенную структуру каталогов. В каталог C:\WebServers\home\helloworld\www\ поместите файл index.html, состоящий из одной строчки:

```
Hello, world!
```

После выполнения предложенных действий откройте браузер и перейдите по адресу **http://helloworld/index.html** и... убедитесь, что ничего не работает (рис. 2.11).

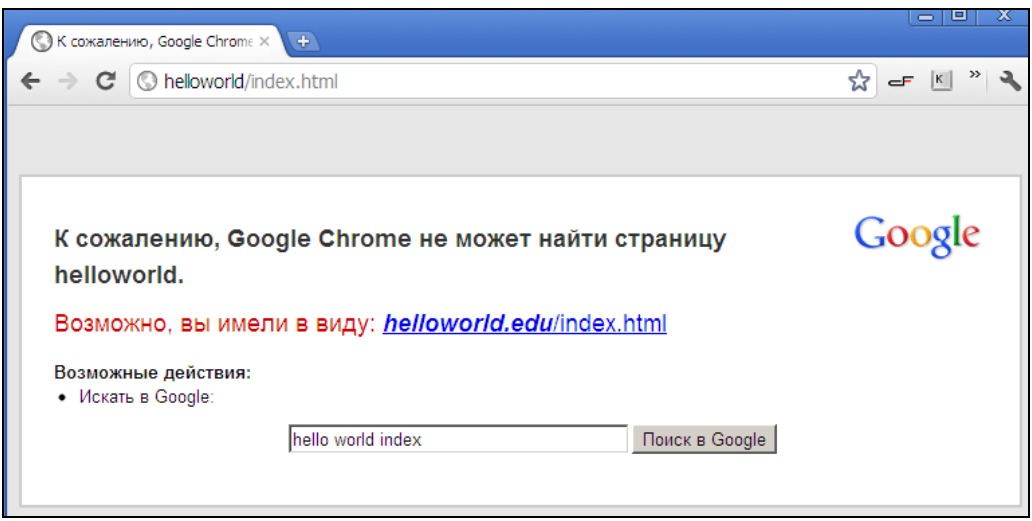


Рис. 2.11. Ошибка при попытке вызова страницы

Дело в том, что веб-сервер еще "не знает" о том, что вы создали папку с новым сайтом. Чтобы Apache "узнал" об этом, перейдите на рабочий стол и дважды щелкните по ярлыку **Restart Denwer**. После перезагрузки веб-сервера попробуйте снова. На этот раз вы должны увидеть правильный результат (рис. 2.12).

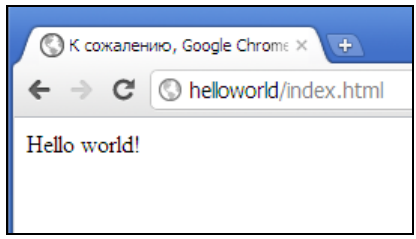


Рис. 2.12. Тестовая HTML-страница

Подводя промежуточный итог, отметим то, что вам следует запомнить:

- все сайты создаются внутри каталога home вашего веб-сервера;
- на каждый сайт отводится отдельная папка, имя которой совпадает с названием сайта в адресной строке браузера;
- для вызова локальных сайтов не нужно приписывать в конце адреса домены первого уровня (".ru", ".com" и т. п.);
- все файлы сайта должны располагаться в каталоге www;
- после создания нового сайта требуется перезапустить веб-сервер.

Выбор текстового редактора

Прежде чем написать свой первый PHP-сценарий, у вас должен возникнуть вопрос: какую программу использовать для написания кода?

Вообще говоря, набирать код вы можете и в стандартном редакторе Блокнот, никто вам этого не запретит. Но разработчики уже давно придумали более удобные инструменты для программирования. В то же время на первом этапе не следует погружаться в изучение громоздких "навороченных" редакторов, т. к. 95% их функциональности вам все равно не пригодится.

Для написания первых PHP-сценариев мы рекомендуем пользоваться бесплатной программой Notepad++. Она имеет целый ряд преимуществ, которые позволяют смело советовать ее в качестве редактора кода:

- она бесплатна;
- ее легко найти и установить. Достаточно воспользоваться ссылкой **http://notepad-plus-plus.org/download/**;
- поддерживается синтаксис PHP. Это значит, что редактор будет подсвечивать различными цветами конструкции языка, что очень удобно и наглядно;
- наконец, она проста. И это является несомненным достоинством программы. Так как хороший редактор должен быть в первую очередь *незаметен* во время написания кода и не должен отвлекать на себя драгоценного внимания программиста.

Установите Notepad++. Для этого перейдите по ссылке **http://notepad-plus-plus.org/download/** и скачайте самую последнюю версию программы.