

PHP — ЭТО ПРОСТО

Начинаем с видеоуроков

Основы HTML и PHP

Принципы организации Интернета

Установка и настройка веб-сервера на локальном компьютере

Работа с файлами

Работа с базами данных

Объектно-ориентированное программирование на PHP

Шаблон проектирования веб-сайтов MVC

Профессиональная разработка веб-приложений на PHP



УДК 681.3.068
ББК 32.973.26-018.1
Л97

Ляпин, Д. А.

Л97 РНР — 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 РНР. Рассказано об использовании объектно-ориентированного подхода в программировании на РНР. 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 РНР-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	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Наталья Периакова</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Марины Дамбиевой</i>
Зав. производством	<i>Николай Тверских</i>

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

© Ляпин Д. А., Никитин А. В., 2012
© Оформление, издательство "БХВ-Петербург", 2012

Оглавление

- Предисловие..... 1**
- Введение..... 3**
- Урок 1. Основы протокола HTTP..... 5**
 - Что происходит между тем, как вы набрали адрес сайта и увидели страницу на экране? 5
 - URL и URI 6
 - Структура адреса сайта 7
 - Исходный код HTML-страницы 7
 - Схема клиент-серверного взаимодействия при работе браузера..... 8
 - Протокол HTTP..... 9
 - Заголовки HTTP 10
 - Как посмотреть данные HTTP-запроса и HTTP-ответа в браузере 10
 - Как скачать сайт утилитой Telnet 12
 - Резюме 15
 - Задания..... 15
- Урок 2. Подготовка рабочего места 16**
 - Установка пакета Денвер 16
 - Структура папок Денвера..... 19
 - Выбор текстового редактора..... 22
 - Первый PHP-сценарий..... 23
 - Резюме 23
 - Задания..... 24
- Урок 3. Введение в HTML..... 25**
 - Создание HTML-документа 25
 - Теги HTML 26
 - Структура HTML-страницы 27
 - Работа с текстом..... 29
 - Абзацы 29
 - Переносы строк 29

Заголовки	30
Оформление текста: полужирный, курсив, подчеркивание	30
Ссылки	30
Изображения	32
Таблицы	33
Создание таблиц.....	33
Параметры таблиц.....	34
Параметры ячеек	34
Особенности таблиц.....	35
XML и XHTML	35
Резюме	36
Задания.....	36
Урок 4. Основы PHP	38
Прежде чем приступить к изучению PHP.....	38
Базовый синтаксис.....	38
Переменные	40
Комментарии	41
Константы.....	41
Типы данных	43
Преобразование типов	45
Операторы и операции	45
Арифметические операции.....	45
Логические операции.....	47
Операции над строками	48
Логические операторы.....	49
Приоритетность операторов	50
Резюме	50
Задания.....	51
Урок 5. Ветвления и функции	53
Ветвление программы	53
Оператор <i>if</i>	54
Тернарный оператор условия.....	56
Оператор <i>switch</i>	57
Функции.....	59
Определение функции	59
Значения по умолчанию	61
Рекурсия.....	61
Область видимости и время жизни переменных	62
Резюме	64
Задания.....	65
Урок 6. Циклы и массивы.....	66
Циклы в PHP	66
Цикл <i>while</i>	66
Цикл <i>do..while</i>	67

Цикл <i>for</i>	68
Бесконечный цикл, операторы выхода из цикла и прерывания итерации цикла	69
Массивы	70
Что такое массив	70
Обход массивов в цикле	71
Функции для работы с массивами	72
<i>count()</i>	72
<i>in_array()</i>	72
<i>sort()</i>	73
<i>explode()</i> и <i>implode()</i>	73
Многомерные массивы	73
Предопределенные массивы	74
Резюме	75
Задания	75
Урок 7. Запросы HTTP, параметры URL и формы HTML	77
Типы запросов HTTP	77
URL и параметры запроса	78
Обработка параметров URL	78
Обработка отправки HTML-формы	81
Резюме	83
Задания	84
Урок 8. Cookies и сессии	85
Что такое cookies и с чем их едят	85
Как это работает?	86
Манипулируем cookies средствами PHP	87
Что такое сессии PHP и как они работают?	88
Практическое использование cookies и сессий. Авторизация на сайте	89
Резюме	93
Задания	94
Урок 9. Работа с файлами	95
Особенности работы с файлами в PHP	95
Два режима работы с файлом	95
Функции для работы с файлами	96
Журнал посещений сайта	100
Загрузка файлов на сервер	104
Функции для работы с каталогами	107
Получение списка файлов и подпапок в каталоге	108
Резюме	109
Задания	110
Урок 10. Работа с базой данных	111
Для чего нужна база данных?	111
Отличие базы данных от СУБД	112
Реляционная база данных	112

Язык SQL	113
Вставка строк	115
Удаление строк	115
Изменение строк	115
Выборка строк	115
Средства PHP для работы с MySQL	116
Подключение к БД	116
Выбор БД	117
Выполнение SQL-запросов	117
Получение результата выборки в виде массива	117
Получение результата выборки в виде ассоциативного массива	118
Получение числа строк в выборке	118
Получение числа измененных строк	119
Получение информации об ошибках	119
Резюме	120
Задание	120
Урок 11. Архитектура сайта	121
Что такое архитектура программы	121
Так что же такое архитектура программы?	121
Уровни абстракции	122
Архитектура MVC	123
Реализация MVC	124
Модель	127
Представление	128
Контроллер	130
Резюме	131
Задание	131
Урок 12. Введение в объектно-ориентированное программирование	132
Что такое объектно-ориентированное программирование	132
Понятие класса	133
Понятие объекта	134
Методы	135
Инкапсуляция	136
Наследование	136
Полиморфизм	139
Спецификаторы (модификаторы) доступа	141
Конструкторы	142
Статические члены классов	143
Абстрактные классы и методы	144
Резюме	145
Задание	146
Урок 13. Совместное использование принципов MVC и ООП	147
Описание задачи	147
Исходные положения	149

Точка входа 149

Иерархия контроллеров..... 151

 Класс *Controller*..... 153

 Класс *C_Base* 153

 Контроллеры *C_View* и *C_Edit*..... 154

Цикл обработки запроса 155

Задание..... 156

Резюме 156

Заключение 157

Описание компакт-диска 159

Литература 163

Предметный указатель 165

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.

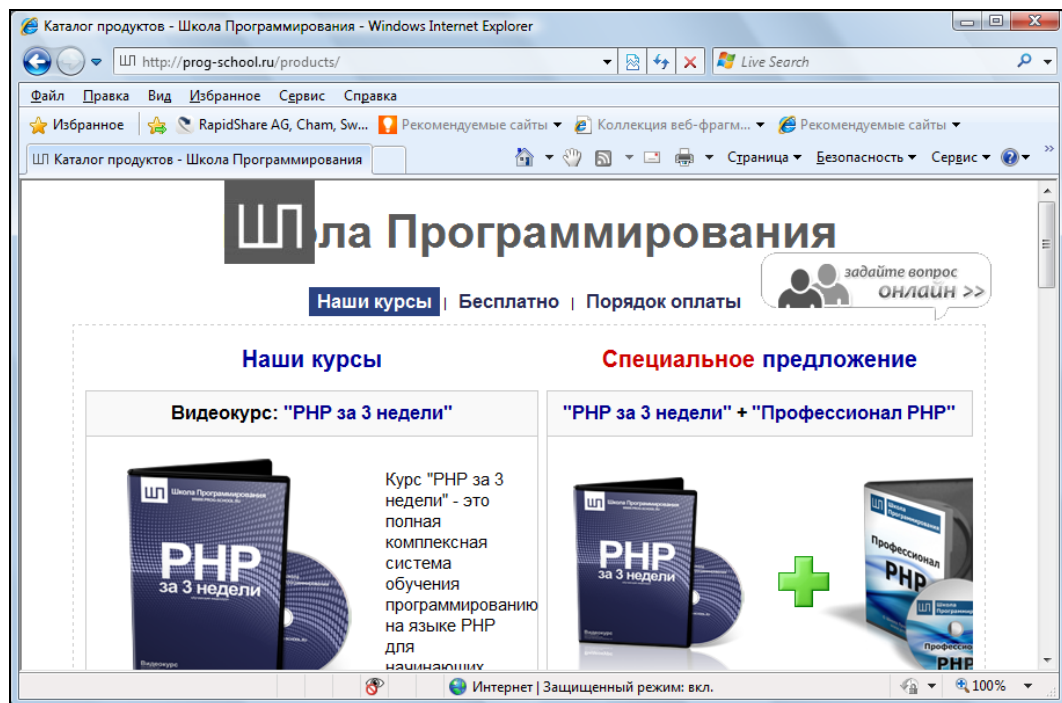


Рис. 1.1. Отображение в браузере страницы **<http://prog-school.ru/catalog>**

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 придерживаться 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.

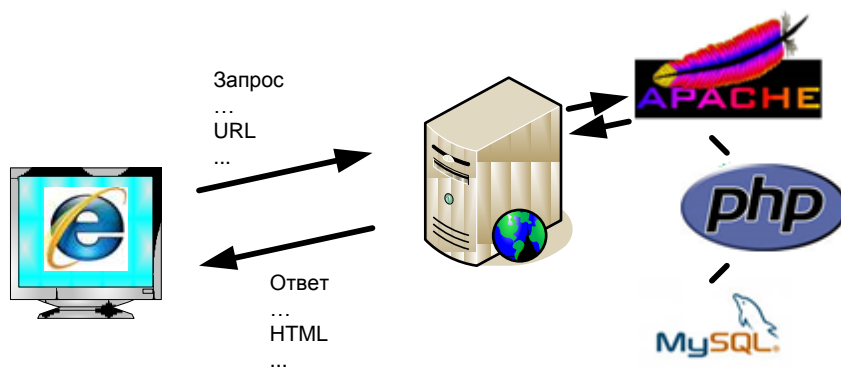


Рис 1.2. Схема взаимодействия браузера и веб-сервера

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 иницируют 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 содержание 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).

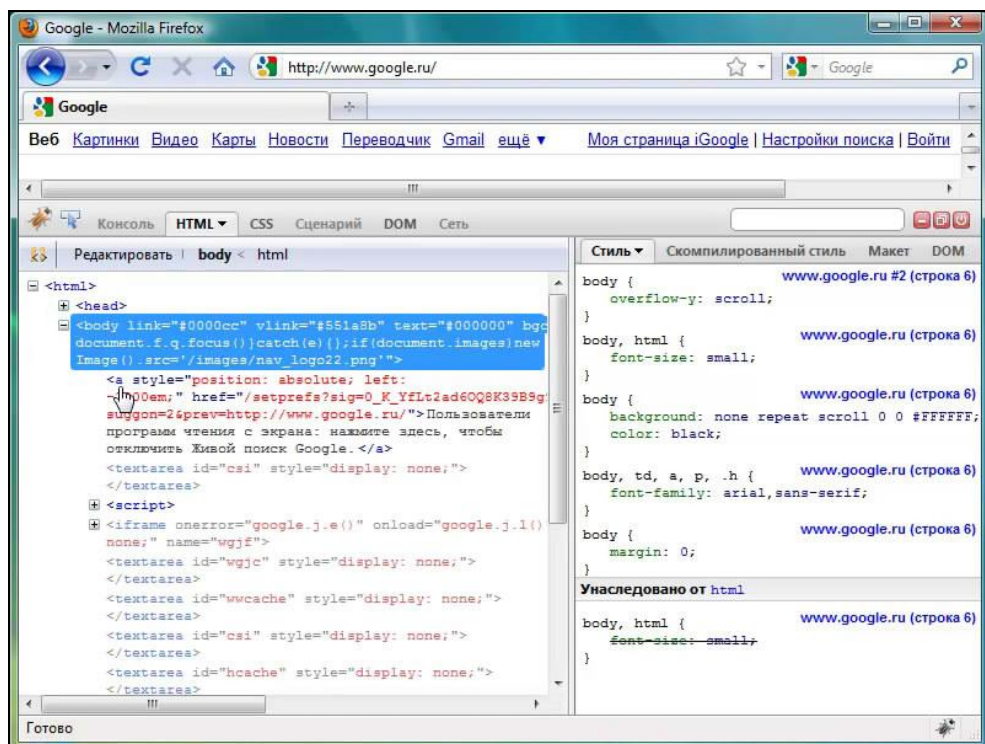


Рис. 1.3. Плагин Firebug

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.

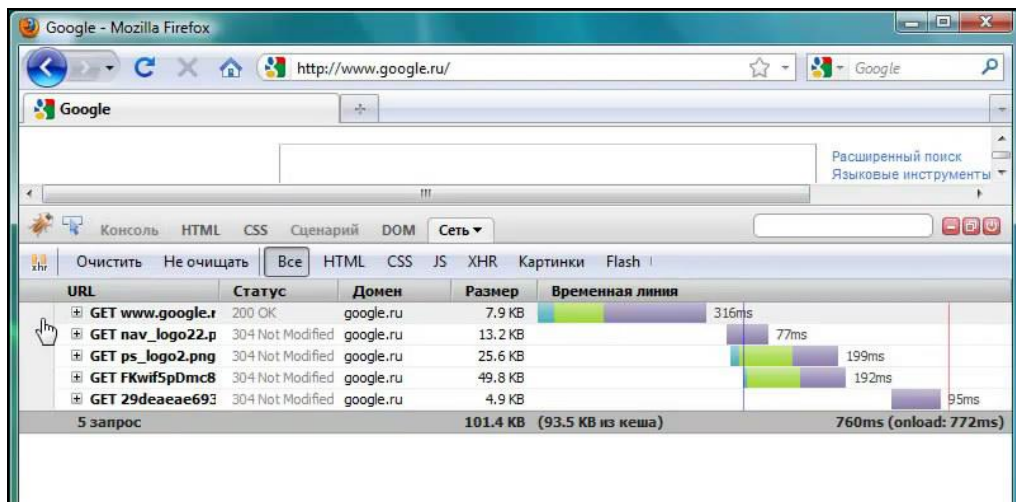


Рис. 1.4. Плагин Firebug, вкладка **Сеть**

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 (TErминаL 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).

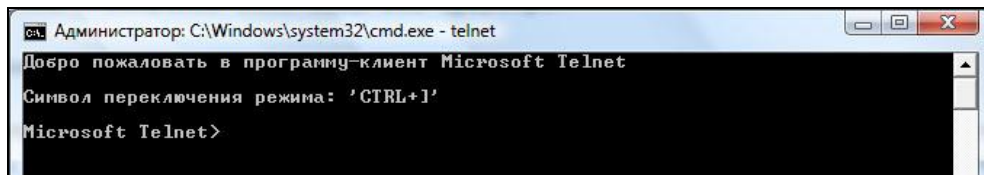


Рис. 1.5. Telnet-клиент Windows

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 **ОК**.

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 протоколов 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
```

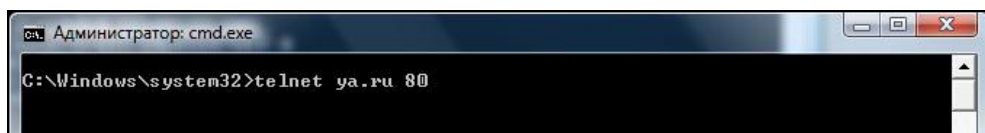



Рис. 1.6. Telnet-соединение с веб-сервером

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.