КРИСТИАН ДАРИ
БОГДАН БРИНЗАРЕ
ФИЛИП ЧЕРЧЕЗ-ТОЗА
МИХАЙ БУСИКА

AJAX и PHP

РАЗРАБОТКА ДИНАМИЧЕСКИХ ВЕБ-ПРИЛОЖЕНИЙ

HIGH TECH

СИМВОЛ®

AJAX and PHP

Building Responsive Web Applications

Cristian Darie, Bogdan Brinzarea,
Filip Chereches-Tosa, Mihai Bucica

HIGH TECH

AJAX и PHP

Разработка динамических веб-приложений

Кристиан Дари, Богдан Бринзаре,
Филип Черчез-Тоза, Михай Бусика

СИМВОЛ®

Санкт-Петербург — Москва
2007

Серия «High tech»

Кристиан Дари, Богдан Бринзаре,
Филип Черчез-Тоза, Михай Бусика

AJAX и PHP: разработка динамических веб-приложений

Перевод А. Киселева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*О. Цилюрик*
Редактор	*В. Овчинников*
Художник	*В. Гренда*
Корректор	*И. Губченко*
Верстка	*Д. Орлова*

Дари К., Бринзаре Б., Черчез-Тоза Ф., Бусика М.
AJAX и PHP: разработка динамических веб-приложений. – СПб.: Символ-Плюс, 2007. – 336 с., ил.
ISBN 5-93286-077-4

Книга «AJAX и PHP: разработка динамических веб-приложений» – самый удобный и полезный ресурс, который поможет вам войти в захватывающий мир AJAX. Вы научитесь писать более эффективные веб-приложения на PHP за счет использования всего спектра возможностей технологий AJAX. Применение AJAX в связке с PHP и MySQL описывается на многочисленных примерах, которые читатель сможет использовать в собственных проектах. Рассмотрены следующие темы: верификация заполнения форм на стороне сервера; чат-приложение, основанное на технологии AJAX; реализация подсказок и функции автодополнения; построение диаграмм в реальном времени средствами SVG; настраиваемые и редактируемые таблицы на основе баз данных; реализация RSS-агрегатора; построение сортируемых списков с поддержкой механизма drag-and-drop. Издание предназначено тем, кто владеет базовыми знаниями PHP, XML, JavaScript и MySQL и хочет узнать все тонкости функционирования AJAX и взаимодействия составляющих этой технологии.

ISBN–13: 978–5–93286–077–9
ISBN–10: 5–93286–077–4
ISBN 1–904811–82–5 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 30.03.2007. Формат 70x100 1/16 . Печать офсетная.
Объем 21 печ. л. Доп. тираж 2000 экз. Заказ N
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Об авторах

Кристиан Дари (Cristian Darie) – программист, имеющий опыт работы с широким кругом современных технологий, автор многочисленных книг, включая популярную серию «Beginning E-Commerce». Компьютерами начал заниматься, как только научился нажимать на клавиши. Впервые вкус успеха в области программирования почувствовал, когда в возрасте 12 лет выиграл главный приз в конкурсе программистов. С тех пор Кристиан достиг многого, а сейчас он занимается изучением архитектур распределенных приложений в рамках своей кандидатской диссертации. Ему всегда нравилось получать отзывы о своих книгах, поэтому, если у вас выдастся свободная минутка, не стесняйтесь передать ему привет от себя. Связаться с Кристианом можно через его персональный веб-сайт по адресу: *www.cristiandarie.ro*.

Кристиан хотел бы выразить глубокую благодарность своим соавторам Богдану, Филиппу и Михаю, а также техническому редактору книги Джимми за их упорный труд, который они вложили в создание этой замечательной книги.

Богдан Бринзаре (Bogdan Brinzarea) имеет очень серьезную подготовку в области информатики. Он получил степень магистра на факультете автоматизации управления и вычислительной техники Бухарестского политехнического университета в Румынии и диплом аудитора факультета информатики политехнической школы Ecole Polytechnique в Париже.

В круг интересов Богдана входят вопросы разработки программного обеспечения для встраиваемых систем, распределенные и мобильные вычисления, а также новейшие веб-технологии. В настоящее время он работает специалистом по альтернативным каналам предоставления услуг в «Banca Romaneasca» (член банка «National Bank of Greece»), где отвечает за реализацию проекта предоставления банковских услуг через Интернет и координирует ряд других проектов, связанных с защитой приложений и применением новых технологий в области банковского дела.

Филип Черчез-Тоза (Filip Cherecheş-Toşa) – веб-программист, свято верящий в большое будущее веб-технологий. Карьеру начал в возрасте 9 лет, когда получил в подарок свой первый Commodore 64, оснащенный устройством внешней памяти на магнитной ленте.

У себя на родине, в Румынии, Филип руководит компанией eXigo (*www.exigo.org*), которая активно занимается разработкой веб-прило-

жений и веб-дизайном. В настоящее время учится в университете города Оради на факультете информатики и активно участвует в работе Румынского сообщества PHP-программистов (*www.phpromania.org*).

Михай Бусика (Mihai Bucica) начал заниматься программированием (а также участвовать и побеждать во многих конкурсах программистов) в возрасте 12 лет. Имея степень бакалавра информатики, выпускник факультета автоматизации управления и вычислительной техники Бухарестского политехнического университета Бусика занимается разработкой коммуникационного программного обеспечения для различных интернет-магазинов.

Несмотря на богатый опыт работы с многочисленными языками программирования и технологиями, Бусика предпочитает C++ и влюблен в слово LGPL. Михай также участвовал в создании книги «PHP 5 and MySQL E-Commerce». Связаться с ним можно через его персональный веб-сайт по адресу: *www.valentinbucica.ro*.

О рецензентах

Эмилиан Баланеску (Emilian Balanescu) – программист, имеющий опыт работы с множеством технологий, включая PHP, Java, .NET, PostgreSQL, MS SQL Server, MySQL и др. В настоящее время работает администратором беспроводной сети в компании accessNET International S.A. Romania, которая предоставляет услуги по обеспечению постоянного доступа к беспроводной сети общенационального масштаба, действующей в радиочастотном диапазоне. Его последняя разработка на данной должности – система управления сетью в реальном масштабе времени, созданная на базе идеологии AJAX (с использованием SNMP, Perl, PHP и PostgreSQL) и использующаяся для наладки, мониторинга и решения задач по диагностике и устранению неполадок. Связаться с ним можно по адресу *http://www.emilianbalanescu.ro*.

Пола Бадаску (Paula Badascu) обучается на третьем курсе Бухарестского политехнического университета, одного из самых известных технических университетов Румынии. Занимается изучением электроники, телекоммуникации и информационных технологий. В настоящее время Пола работает программистом-аналитиком в «NCH Advisors Romania», где занимается разработкой веб-приложений с использованием UML, OOP, PHP, SQL, JavaScript и CSS. Она внесла существенный вклад в анализ и разработку инфраструктуры, используемой для мониторинга состояния дел на фондовом рынке Румынии.

Предисловие

AJAX – это сложный феномен, который разные люди понимают по-разному. С точки зрения обычного пользователя AJAX – это дружественность и более высокая динамичность их любимых сайтов. А у веб-разработчиков AJAX ассоциируется с новыми знаниями и умениями, благодаря которым они могут создавать отличные веб-приложения с меньшими усилиями. В действительности все, что говорится об AJAX, звучит весьма неплохо.

В основе AJAX лежит сочетание различных технологий, которые позволяют уйти от необходимости повторно загружать страницы при переходе с одной страницы на другую. И это лишь первый шаг к тому, чтобы реализовать на веб-сайтах более развитые функциональные возможности, такие как проверка корректности данных в режиме реального времени, перетаскивание объектов на странице мышью и других, которые традиционно не были связаны с веб-приложениями. Компоненты, составляющие AJAX, сами по себе достаточно зрелые (например, объект XMLHttpRequest был разработан в Microsoft еще в 1999 г.), однако их роль в свете тенденций развития Всемирной паутины получает новое развитие, и этот сплав технологий еще очень изменится, прежде чем его можно будет применять для удовлетворения потребностей конечных пользователей. К моменту написания этой книги названию «AJAX» исполнился всего один год.

Разумеется, AJAX не надо считать панацеей от всех бед, ведь это лишь одна из созданных в последнее время технологий. Как и любая другая технология, AJAX может применяться неправильно или не там где надо. Кроме того, AJAX порождает и свои собственные проблемы – вам придется бороться с несовместимостью броузеров, а страницы, реализованные на основе AJAX, не работают без поддержки JavaScript, их нельзя поместить в закладки, а поисковые системы не всегда в состоянии разобраться с их содержимым. И вообще AJAX вызывает восторг далеко не у всех. Одни стремятся выстроить каркас приложения на основе JavaScript, другие вообще предпочитают обходиться без него. Однако большинство из вас согласится, что, как правило, истина лежит посередине.

В книге «AJAX и PHP: разработка динамических веб-приложений» мы избрали наиболее прагматичный и надежный подход к обучению, основанный на решении практических задач, с которыми, на наш взгляд, любой веб-разработчик рано или поздно столкнется. Мы пока-

жем, как избежать самых распространенных ошибок, как писать высокопроизводительный код AJAX и как создать функциональность, которую без труда можно интегрировать в работающие и будущие веб-приложения без пересборки проектов. Знания, полученные из этой книги, можно сразу применить на практике и внедрить их в свои веб-приложения, написанные на языке PHP.

Мы надеемся, что эта книга окажется для вас полезной и поможет в работе над проектами. Самые последние сведения, касающиеся ее, можно найти на сайте *http://ajaxphp.packtpub.com*.

Здесь выложены дополнительные главы и ресурсы, с которыми мы рекомендуем ознакомиться.

Краткое содержание книги

Глава 1 *«AJAX и будущее веб-приложений»* представляет собой начальный экскурс в мир AJAX и его возможности, которые открываются перед веб-разработчиками и компаниями. В этой главе вы создадите свою первую веб-страницу на основе AJAX.

В главе 2 *«Клиентские технологии на основе JavaScript»* рассказывается о технологиях, применяемых при создании клиентских частей веб-приложений AJAX, основанных на JavaScript, DOM, объекте XMLHttpRequest и XML. Данная глава не претендует на роль всеобъемлющего справочного руководства по всем этим технологиям, однако после ее прочтения вы сможете построить на их основе крепкий фундамент своих будущих веб-приложений.

Глава 3 *«Технологии, применяемые на стороне сервера: PHP и MySQL»* завершает теоретическую часть книги и рассказывает о том, как создаются сценарии на стороне сервера, предназначенные для взаимодействия с клиентской частью AJAX. Здесь рассмотрены различные методы решения самых распространенных задач, включая проблемы безопасности JavaScript и обработку ошибок.

Глава 4 *«Верификация заполнения форм в AJAX»* проведет вас через создание современной, динамической и безопасной системы проверки правильности заполнения форм, которая реализует как проверку в режиме реального времени, так и проверку на стороне сервера после отправки формы.

Глава 5 *«Чат AJAX»* представит пример простейшего работающего веб-приложения, предназначенного для организации прямого общения по сети, в котором задействуется только код AJAX и не применяются Java-апплеты, Flash или какие-либо другие специализированные библиотеки, широко применяемые сейчас при разработке подобных приложений.

Глава 6 *«Подсказки и функция автодополнения в AJAX»* продемонстрирует пример реализации функциональности, напоминающей под-

сказки Google, которая поможет вам быстро отыскать требуемую функцию языка PHP и перенаправит на официальную справочную страничку по этой функции.

Глава 7 *«Построение диаграмм в реальном времени средствами SVG и AJAX»* расскажет, как реализовать создание диаграмм в режиме реального времени на основе решений, предлагаемых AJAX и SVG. SVG (Scalable Vector Graphics, масштабируемая векторная графика) – это язык создания графических изображений, который может применяться для вычерчивания фигур и вывода текста.

Глава 8 *«Таблицы в AJAX»* научит, как использовать преимущества AJAX для представления данных в табличной форме. Здесь вы также научитесь производить разбор XML-документов с помощью XSLT для извлечения табличных данных.

В главе 9 *«Чтение лент новостей в AJAX»* показан пример простейшего веб-приложения, считывающего информацию из лент новостей, реализованного на базе XML, XSLT и SimpleXML (библиотеки языка PHP).

Глава 10 *«Технология drag-and-drop в AJAX»* продемонстрирует использование платформы script.aculo.us для построения простейших списков элементов с возможностью перетаскивания мышью.

Приложение А *«Подготовка рабочего окружения»* показывает, как установить и сконфигурировать необходимое программное обеспечение, в состав которого входят: Apache, PHP, MySQL, phpMyAdmin. Примеры этой книги работают, только если рабочее окружение и базы данных настроены так, как показано в этом приложении.

На сайте *http://ajaxphp.packtpub.com* можно найти все примеры, приведенные в тексте книги.

Что потребуется для работы с этой книгой

PHP 5, веб-сервер и сервер баз данных. Примеры программ тестировались в различных окружениях, но главным образом с Apache 2 в качестве веб-сервера и MySQL 4.1 и MySQL 5 в качестве серверов баз данных.

Можно выбрать другой веб-сервер или продукт для работы с базами данных, но в этом случае мы не ручаемся за работоспособность процедур, описываемых в книге. Очень важно, чтобы версия PHP была не ниже 5, поскольку не все объектно-ориентированные особенности, задействованные нами, поддерживаются в более ранних версиях.

Дополнительные сведения по настройке рабочего окружения вы найдете в приложении А. Если в системе уже установлено все необходимое программное обеспечение, то вам останется только прочитать заключительную часть приложения, в которой рассказывается, как создать базу данных, фигурирующую в примерах этой книги.

Соглашения

Вам встретятся различные способы оформления текста, призванные выделить различные типы информации. Вот несколько примеров такого выделения и описание их назначения.

Исходные тексты программ могут оформляться тремя способами. Ключевые слова в тексте выделяются шрифтом, например: «Мы можем подключать содержимое других файлов с помощью директивы `include`».

Блоки кода форматируются так:

```
// функция обращается к серверу с помощью объекта XMLHttpRequest
function process()
{
    // получить имя, введенное пользователем при заполнении формы
    name = document.getElementById("myName").value;
    // запустить сценарий quickstart.php на сервере
    xmlHttp.open("GET", "quickstart.php?name=" + name, false);
    // выполнить синхронный запрос сервера
    xmlHttp.send(null);
    // прочитать ответ
    handleServerResponse();
}
```

Для того чтобы привлечь ваше внимание к отдельным строкам исходного кода в блоке, они выделяются жирным шрифтом:

```
// функция обращается к серверу с помощью объекта XMLHttpRequest
function process()
{
    // получить имя, введенное пользователем при заполнении формы
    name = document.getElementById("myName").value;
    // запустить сценарий quickstart.php на сервере
    xmlHttp.open("GET", "quickstart.php?name=" + name, false);
    // выполнить синхронный запрос сервера
    xmlHttp.send(null);
    // прочитать ответ
    handleServerResponse();
}
```

Текст, который должен вводиться в командной строке, форматируется так:

```
./configure --prefix=/usr/local/apache2 --enable-so --enable-ssl --withssl
--enable-auth-digest
```

Новые термины и **важные понятия** выделяются жирным шрифтом. Текст, который вы увидите на экране, в меню или в диалогах, в тексте книги заключается в кавычки, например: «щелкните по кнопке «Далее», чтобы перейти к следующему экрану».

Примечание

Предупреждения или важные примечания будут выглядеть так.

Совет

А так будут оформляться советы и подсказки.

Обратная связь с читателями

Мы всегда рады получать отзывы от наших читателей. Дайте нам знать, что вы думаете об этой книге, что вам понравилось или, наоборот, что не понравилось. Отзывы читателей имеют для нас очень большое значение, т. к. они помогают издавать действительно необходимые книги.

Свои отзывы отправляйте по адресу *feedback@packtpub.com*, при этом в поле «Тема» не забудьте указать название книги.

Заполнив и отправив форму SUGGEST A TITLE на сайте *www.packtpub.com*, можно предложить тему для новой книги. Предложение можно отправить и по электронной почте на адрес *suggest@packtpub.com*.

Если вы обладаете необходимыми знаниями и хотите написать или передать для публикации свою книгу, обращайтесь по адресу *www.packtpub.com/authors*.

Поддержка покупателей

Покупателям книг, выпущенных издательством Packt, мы делаем дополнительное предложение, чтобы они могли извлечь максимум пользы из своей покупки.

Загрузка исходных текстов примеров из этой книги

Чтобы загрузить исходные тексты примеров или дополнительные ресурсы, имеющие отношение к этой книге, надо зайти на страницу *http://www.packtpub.com/support* и выбрать название книги из списка. После того как вы сделаете выбор, откроется список файлов, доступных для скачивания.

Примечание

Скачиваемые файлы содержат инструкции по их применению.

Список опечаток

Мы приложили максимум усилий, чтобы избежать опечаток и неточностей, но ошибки все равно могут встретиться. Если вы найдете ошибку в тексте или в исходном коде примера и сообщите нам об этом, мы будем вам весьма признательны. Тем самым вы убережете других читате-

лей от разочарований и поможете улучшить последующие издания этой книги. Сообщения об ошибках можно оставить по адресу *http://www.packtpub.com/support*, где надо щелкнуть по ссылке `Submit Errata` и ввести подробное описание найденной ошибки. Если ваше замечание подтвердится, оно будет принято и добавлено в список известных опечаток. Вы можете просмотреть список известных опечаток, щелкнув по названию книги на странице *http://www.packtpub.com/support*.

Вопросы

Вопросы, так или иначе связанные с книгой, можно задать, написав нам по адресу *questions@packtpub.com*, и мы приложим все усилия, чтобы ответить на них.

1

AJAX и будущее веб-приложений

«Компьютер, нарисуй робота!», – сказал мой маленький кузен компьютеру, когда впервые увидел его. (Поскольку я научил компьютер не реагировать на приказы посторонних, то он, само собой, проигнорировал эту команду.) Если вы похожи на меня, то первая мысль, которая могла прийти вам в голову, была бы: «как глупо» или «как забавно», и это было бы ошибкой. Наш обученный и сформировавшийся мозг до определенной степени знает, как работать с компьютером. Люди обучаются приспосабливать компьютеры под себя, чтобы компенсировать отсутствие у компьютеров возможности понимать человеческий язык. (С другой стороны, возможности людей приспосабливаться также весьма ограничены, но это уже другая история.)

Этот забавный случай наглядно показывает, что в работе с компьютером люди руководствуются инстинктом. В идеале той речевой команды должно было бы хватить, чтобы компьютер порадовал моего кузена. В последние годы был достигнут значительный успех в развитии технологий, повышающих дружественность компьютеров по отношению к пользователю, но впереди еще долгий путь, который нам предстоит пройти, прежде чем мы получим по-настоящему интеллектуальные компьютеры. А пока люди должны обучаться взаимодействию с компьютерами, причем некоторые даже влюбляются в черный экран с крошечным приглашением к вводу команды на нем.

Неслучайно многие навыки работы с компьютером вырабатываются программами с пользовательским интерфейсом, который обеспечивает интуитивно понятное (и удобное) взаимодействие с человеком. Этим, наверное, можно объяснить популярность правой кнопки мыши, восхищение такими функциональными возможностями, как перетаскивание объектов мышью, или тем, что, введя слово в простое текстовое поле, можно выполнить его поиск по всему Интернету за 0,1 секунды (или что-то около того). Индустрия программного обеспечения (по крайней мере, та ее часть, которая приносит прибыль) видит,

анализирует и изучает ситуацию. Сейчас рынок заполнен программами со сверкающими кнопками, значками, окошками и разнообразными мастерами, и люди *платят за них большие деньги*.

Главное, что поняли в индустрии ПО, – это что *его удобство в использовании и доступность* представляют собой эквивалент мощного двигателя в красном спортивном автомобиле. Это замечательно, когда точка зрения бизнеса совпадает с точкой зрения обычного пользователя, потому что прибыль более или менее пропорциональна тому, насколько удовлетворен пользователь.

Мы планируем быть очень практичными и краткими в этой книге, но прежде чем обратиться к вашему любимому занятию (написанию кода), отвлечемся немного, чтобы вспомнить, что и почему мы делаем. Мы влюблены в звук, извлекаемый при нажатии на клавиши, поэтому мы легко забываем, что самая главная причина существования технологий заключается в служении людям и в том, чтобы делать их жизнь интереснее, а работу – эффективнее.

Для создания конечных приложений очень важно понять образ мышления человека. Мы еще далеки от этой цели, но уже понимаем, что конечные пользователи нуждаются в интуитивно понятных интерфейсах. Их мало интересует, с какой операционной системой они работают, пока функциональные возможности, которые они имеют, совпадают с тем, что они *ожидают*. Это очень важный момент, о котором надо помнить, тогда как многие программисты, работая с конечными пользователями, продолжают мыслить чисто техническими категориями (хотя в типичной группе разработчиков программист не общается с пользователем напрямую). Если вы не согласны с этим утверждением, попробуйте вспомнить, сколько раз вы произнесли слова *база данных* в разговоре с неспециалистом.

В результате наблюдений за привычками и потребностями людей, работающих с компьютерными системами, родился термин **юзабилити** (удобство в использовании), обозначающий *искусство* удовлетворять ожидания пользователей от интерфейса, понимать характер их работы и соответственно создавать приложения.

Исторически сложилось, что методы улучшения юзабилити применялись в основном к **обычным приложениям** по той простой причине, что они требовали наличия инструментальных средств, не доступных для **веб-приложений**. Однако по мере развития Интернета рос и потенциал технологий, позволяющих сделать это.

Современные сетевые технологии не только дают возможность повышать качество работы в Интернете, но и позволяют совершенствовать приложения, разрабатываемые специально для интрасетей. Наличие сайтов, дружественных к пользователям, очень важно для сетевого бизнеса, поскольку Интернет никогда не спит, а клиенты зачастую уходят на другой сайт только потому, что он выглядит лучше или *кажется* бо-

лее динамичным. В то же самое время возможность создавать дружественные веб-интерфейсы позволяет внедрять в интрасети программные решения, которые ранее создавались как обычные приложения.

Создание дружественного ПО всегда было проще в случае обычных приложений, чем веб-приложений, просто потому, что Интернет проектировался как средство доставки текста, изображений и несложной функциональности. За последние несколько лет проблема дружественности обострилась в связи с ростом объемов услуг, поставляемых через Интернет.

Для решения этой проблемы были придуманы (и они продолжают появляться) новые технологии, которые позволяют создавать яркие, удобные и мощные веб-приложения. В качестве примеров можно привести получившие широкую известность **Java-апплеты** и **Macromedia Flash**, которые требуют установки в броузеры дополнительных библиотек.

Предоставление функциональности через Интернет

Веб-приложения – это приложения, функциональные возможности которых обеспечиваются сервером и доставляются конечным пользователям по сети, такой как Интернет или интранет. Конечные пользователи запускают веб-приложения с помощью **тонкого клиента** (веб-броузера), который знает, как обрабатывать и отображать данные, полученные от сервера. В противоположность им, обычные приложения являют собой пример **толстого клиента**, который выполняет большую часть работы.[1]

Прогрессирующее развитие веб-приложений позволяет надеяться, что в один прекрасный день они будут выглядеть и вести себя как их более зрелые (и более мощные) родственники – обычные приложения. Поведение любого ПО, предназначенного для взаимодействия с людьми,

[1] Здесь авторы сознательно несколько упрощают картину, сводя ее к двум градациям: «толстый» и «тонкий» клиенты. На самом деле таких крупных градаций три (таков и хронологический порядок их развития): а) «монолитные» приложения, выполняющие всю работу автономно (говорить здесь о клиентской части бессмысленно, если не считать сервером такого приложения операционную систему, но это уже масло масленое; самый известный пример этого класса – MS Office); б) «толстый» клиент, пользующийся сервисами локального сервера (самые очевидные примеры – это приложения работы с базами данных, например все бухгалтерские системы); в) «тонкий» клиент, взаимодействующий с удаленным сервером, реализующим всю функциональность системы, по специализированному сетевому протоколу (почти всегда под таким протоколом имеют в виду HTTP, но таким же протоколом является графический протокол X и другие). – *Примеч. науч. ред.*

стало еще важнее, чем было до сих пор, потому что сейчас уровень подготовки пользователей варьируется в гораздо более широких пределах, чем раньше, когда они были технически грамотнее. Теперь, например, вы должны предоставить Синди, начальнику отдела сбыта, хорошо оформленные отчеты и обеспечить Дейва, сотрудника того же отдела, удобными формами ввода данных.

Поскольку удовлетворение нужд конечного пользователя – самое важное требование, необходимо создавать такие приложения, которые будут удовлетворять всех пользователей, взаимодействующих с ними. Что касается веб-приложений, их процесс движения к совершенству будет считаться завершенным тогда, когда интерфейс и поведение приложений не будут свидетельствовать о том, где они исполняются – на локальном ли компьютере или на удаленном сервере, поставляя результаты своей работы через сеть. Предоставление удобных интерфейсов через сеть всегда было делом проблематичным просто потому, что некоторые функциональные возможности, которыми, как правило, обладают обычные приложения, например перетаскивание объектов мышью или одновременное выполнение множества задач в одном и том же окне, были невозможны.

Еще одна трудность, с которой постоянно сталкиваются разработчики веб-приложений, – это **стандартизация**. Сейчас необходимо, чтобы все, что доступно через Интернет, было проверено на совместимость, по крайней мере, с двумя или тремя броузерами с целью гарантировать, что все посетители смогут извлечь максимум пользы из вашего сайта.

Преимущества веб-приложений

Да, попытки предоставить функциональные возможности через Интернет сопряжены с массой трудностей. Но почему тогда не попробовать сделать то же самое посредством обычных приложений? Это, конечно, верно, но даже проблемы с дружественностью веб-приложений не помешали им приобрести небывалую популярность, потому что они предлагают массу важных преимуществ, которые отсутствуют в обычных приложениях.

- **Установка веб-приложений проще и обходится дешевле.** Благодаря использованию веб-приложений предприятия могут снизить затраты на содержание отделов ИТ, которые отвечают за установку программного обеспечения на компьютеры пользователей. В этом случае пользователю необходимы лишь компьютер с броузером и соединение с Интернетом или корпоративной сетью.
- **Обновление веб-приложений проще и обходится дешевле.** Стоимость обслуживания ПО всегда имела большое значение. Обновление ПО в чем-то схоже с его установкой, поэтому упомянутые выше преимущества имеются и в этой ситуации. Как только приложение будет обновлено на сервере, все сразу же смогут работать с новой версией.

- **Веб-приложения предъявляют более гибкие требования к конечному пользователю.** Вам достаточно будет установить веб-приложение на сервере, работающем под управлением любой современной ОС, и вы сможете пользоваться им через Интернет или интранет на любом компьютере, работающим под управлением Mac OS, Windows, Linux или какой-либо другой ОС. Если приложение не будет содержать ошибок, оно одинаково хорошо будет работать в любом современном веб-броузере, будь то Internet Explorer, Mozilla Firefox, Opera или Safari.
- **Веб-приложения облегчают организацию централизованного хранения данных.** Если необходимо обращаться к одним и тем же данным из разных мест, то намного проще организовать их хранение в одном месте, вместо того чтобы разбрасывать по разным базам данных. Благодаря этому отпадет необходимость выполнения операций синхронизации и повысится степень их защищенности.

Далее в этой книге мы будем рассматривать применение современных веб-технологий для создания качественных веб-приложений, максимально задействующих возможности, предлагаемые Всемирной паутиной. Но прежде чем углубиться в детали, мы предпримем *краткий* экскурс в историю развития.

Разработка веб-сайтов до 1990 года

Интернет появился много раньше, но все-таки именно в 1991 г. был изобретен **протокол передачи гипертекстовой информации (HyperText Transfer Protocol – HTTP)**, который до сих пор применяется для передачи данных через Интернет. В своей начальной версии он делал лишь чуть больше, чем просто открывал/закрывал соединение. Последние версии HTTP (версия 1.0 появилась в 1996 г., а версия 1.1 – в 1999) стали тем протоколом, который мы знаем и применяем.

HTTP и HTML

Протокол HTTP поддерживается всеми веб-броузерами и прекрасно справляется с возложенными на него обязанностями, которые заключаются в доставке данных через Интернет. Всякий раз, когда вы в своем любимом броузере запрашиваете веб-страницу, это предполагает использование протокола HTTP. Например, если в адресной строке броузера Firefox ввести *www.mozilla.org*, то по умолчанию он предположит, что вы имели в виду *http://www.mozilla.org*.

Стандартный тип документа в Интернете, который веб-броузеры могут понять, проанализировать и отобразить, – это документ, написанный на **языке разметки гипертекста (HyperText Markup Language – HTML).** HTML – это язык описания содержимого и форматирования документа, который в основном состоит из статического текста и изображений. HTML не был предназначен для создания сложных веб-приложений

с изменяющимся содержимым или дружественными интерфейсами. Для того чтобы через HTTP получить другую страницу HTML, ее надо полностью загрузить, а сама она должна существовать в указанном месте в виде статического документа еще до выполнения запроса. Очевидно, что эти ограничения не способствуют созданию чего-нибудь интересного.

Тем не менее HTTP и HTML по-прежнему остаются удачной парой, которая понятна как веб-серверам, так и веб-клиентам (броузерам). Как мы уже знаем, они составляют фундамент современного Интернета. На рис. 1.1 показан простейший пример взаимодействия клиента и сервера, когда пользователь запрашивает веб-страницу из Интернета с помощью протокола HTTP.

Рис. 1.1. *Простой запрос HTTP*

Примечание

Необходимо запомнить три вещи:

1. Взаимодействия по протоколу HTTP всегда осуществляются между веб-клиентом (ПО, выполняющим запрос, таким как веб-броузер) и веб-сервером (ПО, отвечающим на запрос, таким как Apache или IIS). С этого момента везде в книге мы будем писать просто «клиент», подразумевая веб-клиент, и просто «сервер», подразумевая веб-сервер.
2. Пользователь – человек использующий программу-клиент.
3. Несмотря на то, что протокол HTTP (и его безопасная версия **HTTPS**) едва ли не самый важный, тем не менее это не единственный протокол, применяемый в Интернете. Различные типы веб-серверов используют различные протоколы для решения разнообразных задач, как правило, не связанных с простым просмотром сети. Чаще всего в этой книге мы будем говорить о протоколе HTTP, и, говоря «веб-запрос», мы будем подразумевать запрос с использованием протокола HTTP, в противном случае мы будем явно указывать название протокола.

Безусловно, комбинация HTTP–HTML предоставляет весьма ограниченные возможности, поскольку пользователи могут получать из Интернета только статическую информацию (страницы HTML). Чтобы восполнить нехватку функциональных возможностей, были разработаны различные дополнительные технологии.

Несмотря на то, что все веб-запросы, о которых мы будем говорить далее, по-прежнему применяют для передачи информации протокол HTTP, сама эта информация может генерироваться веб-сервером динамически (скажем, извлекаться из базы данных). Кроме того, она может содержать не только текст на языке HTML, позволяя клиенту не только просматривать содержимое статических страниц, но и производить некоторые дополнительные действия.

Технологии, расширяющие возможности Интернета, образуют две основные категории:

- **Клиентские технологии**, позволяющие веб-клиенту не только отображать статические документы, но и делать более интересные вещи. Как правило, эти технологии дополняют, а не замещают язык HTML.
- **Серверные технологии**, предоставляющие возможность перенести логику приложения на сторону сервера с целью создавать веб-страницы на лету.

PHP и другие серверные технологии

Серверные веб-технологии позволяют веб-серверу не просто возвращать запрошенные HTML-файлы, но и выполнять дополнительные действия, например сложные вычисления, запускать объектно-ориентированные программы, работать с базами данных и многое другое.

Только представьте себе, какой объем данных должен обработать Amazon, чтобы сгенерировать индивидуальные рекомендации для каждого посетителя. Или Google, который выполняет поиск информации по гигантской базе данных, чтобы удовлетворить запрос. Да, технологии обработки данных на стороне сервера стали механизмом, запустившим революцию во Всемирной паутине, и причиной, по которой Интернет в нынешнем его виде имеет такой успех.

Очень важно понимать, что, независимо от происходящего на стороне сервера, клиент должен получить ответ на том языке, который ему понятен (что вполне очевидно), например, на языке HTML, который имеет массу ограничений, о чем мы уже говорили.

PHP – это одна из технологий, применяемых для реализации логики приложений на стороне сервера. Введение в PHP будет дано в главе 3. Мы будем использовать PHP в процессе создания и изучения примеров **AJAX**. Тем не менее вы должны знать, что у PHP много конкурентов, таких как **ASP.NET (Active Server Pages** – активные серверные страницы, веб-технология, разработанная в Microsoft), **Java Server Page (JSP** – серверные страницы на языке Java), **Perl**, **ColdFusion**, **Ruby on Rails** и других. Каждая из этих технологий предоставляет свои возможности по реализации функциональности на стороне сервера.

PHP – это не только серверная технология, но и язык, на котором программисты могут писать сценарии. На рис. 1.2 показан пример запро-

са страницы index.php. На этот раз вместо передачи клиенту содержимого файла index.php сервер исполняет сценарий index.php и возвращает полученные результаты. Эти результаты должны быть размечены по правилам HTML или любого другого языка, понятного клиенту.

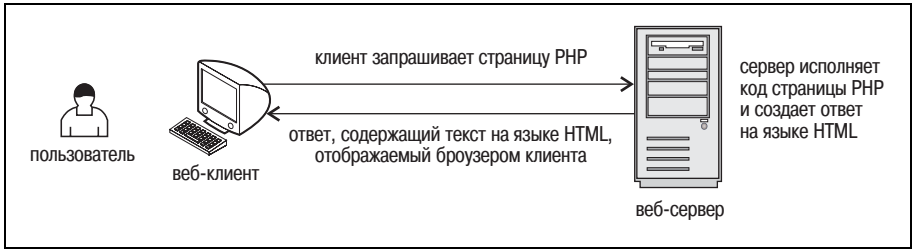

Рис. 1.2. *Клиент запрашивает страницу PHP*

Обычно на стороне сервера также должен размещаться **сервер баз данных**, который будет управлять данными. В примерах из этой книги будет фигурировать **MySQL**, но концепция в целом применима к любому другому серверу баз данных. Основы работы с базами данных и PHP вы изучите в главе 3.

Однако даже при использовании PHP, который может выполнять сложные запросы к базе данных, зависящие от ситуации, броузер пользователя по-прежнему будет статически отображать скучные, не вызывающие интереса веб-документы.

Потребность в более широких функциональных возможностях удовлетворяется отдельной категорией веб-технологий, которые называются клиентскими технологиями. Современные броузеры способны анализировать не только простой HTML. Давайте посмотрим как.

JavaScript и другие клиентские технологии

Различные клиентские технологии отличаются друг от друга в первую очередь способом, которым они загружаются и исполняются веб-клиентом. **JavaScript** – это язык сценариев, код которых в виде простого текста может быть внедрен прямо в страницы на языке HTML. Страница HTML, запрашиваемая клиентом, может содержать сценарии JavaScript. Все современные броузеры поддерживают JavaScript и не требуют от пользователя установки дополнительных компонентов в систему.

JavaScript – это самостоятельный язык программирования (теоретически он не связан с разработкой веб-приложений). Он поддерживается большинством веб-клиентов на любой платформе и обладает некоторыми объектно-ориентированными возможностями. Язык JavaScript относится к интерпретирующим и потому не годится для разработки приложений с интенсивными вычислениями или драйверов устройств и должен целиком доставляться броузеру клиента для последующей интерпретации. Кроме того, он испытывает определенные

проблемы с безопасностью, но при использовании в составе веб-страниц прекрасно справляется с возложенными на него задачами.

Благодаря JavaScript разработчики наконец получили возможность создавать веб-страницы с визуальными эффектами и способностью проверять правильность заполнения форм, избавив тем самым пользователей от необходимости повторно загружать всю страницу (с потерей всех введенных ранее данных, кстати), если они забыли указать какую-либо информацию (например, пароль или номер кредитной карточки) или в случае ошибки. Однако несмотря на имеющийся потенциал, JavaScript никогда не применялся должным образом, чтобы сделать веб-интерфейс действительно дружественным к пользователю, как в обычных приложениях.

Среди других клиентских технологий, наделенных функциональными возможностями, можно назвать Java-апплеты и Macromedia Flash. Java-апплеты пишутся на весьма популярном и мощном языке программирования Java и исполняются **виртуальной Java-машиной (Virtual Java Machine – JVM)**, которую необходимо отдельно устанавливать в систему. Без сомнения, Java-апплеты позволяют создавать более сложные проекты, но применительно к веб-приложениям они уже потеряли свою былую популярность, поскольку потребляют значительное количество системных ресурсов. Иногда даже сам запуск их может занимать значительное время, и вообще они слишком тяжеловесны для простых и нетребовательных веб-приложений.

Технология Macromedia Flash обладает весьма широкими возможностями по созданию графических и анимационных эффектов и фактически стала стандартом для разработки подобных веб-приложений. Macromedia Flash также требует установки *дополнительных компонентов* броузера. Технологии Flash становятся все более мощными, и каждый год появляются все новые и новые ее разновидности.

Комбинируя HTML с серверными и клиентскими веб-технологиями, можно строить очень мощные веб-приложения.

Чего не хватает?

Итак, в нашем распоряжении есть практически все, зачем же нам нужны новые технологии? Чего еще не хватает?

Как говорилось в начале главы, любая технология существует для того, чтобы удовлетворять потребности рынка. А какая-то часть рынка всегда хочет предоставлять в распоряжение веб-клиентов более широкие функциональные возможности, без использования Flash, Java-апплетов или других технологий, которые либо выглядят слишком ярко и броско, либо слишком тяжеловесны для решения простых задач. В подобных случаях разработчики обычно создают веб-сайты и веб-приложения на основе комбинации HTML, JavaScript и PHP (или какой-нибудь другой серверной технологии). В данной ситуации типичный за-

прос выглядит так, как показано на рис. 1.3, где изображен запрос HTTP и ответ, состоящий из HTML и внедренного в него сценария на языке JavaScript, созданный программно с помощью PHP.

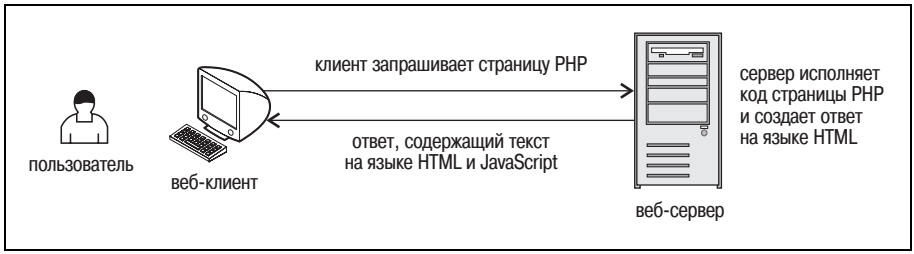

***Рис. 1.3.** HTTP, HTML, PHP и JavaScript в действии*

Подобное решение таит в себе неприятности, состоящие в том, что, когда клиенту необходимо получить новую порцию данных, он вынужден отправлять серверу новый запрос HTTP и повторно загружать страницу целиком, приостанавливая на время деятельность пользователя. **Повторная загрузка страницы** – это неизбежное зло в данной ситуации, а помочь нам победить это зло призван AJAX.

Что такое AJAX

Название AJAX – это акроним, раскрывающийся как **Asynchronous JavaScript and XML** и означающий **асинхронный JavaScript и XML**. Если это название, на ваш взгляд, мало о чем говорит, мы согласимся с вами. Проще говоря, можно считать, что AJAX – это «JavaScript с расширенными правами», поскольку по своей сути эта технология представляет собой сценарии на языке JavaScript, которые по мере необходимости в фоновом режиме выполняют запросы к серверу и получают дополнительные данные, обновляя отдельные части страницы и тем самым исключая необходимость повторной ее загрузки целиком. На рис. 1.4 наглядно показано, что происходит, когда посетитель запрашивает веб-страницу, созданную с применением технологии AJAX.

С точки зрения перспективы AJAX обладает лучшей сбалансированностью между функциональностью, реализуемой на стороне клиента, и функциональностью, реализуемой на стороне сервера, при выполнении действий, затребованных пользователем. До этого места функциональность клиента и функциональность сервера рассматривались как отдельные части, которые работают независимо друг от друга в ответ на действия, предпринимаемые пользователем. AJAX предлагает новое решение – распределить нагрузку между клиентом и сервером, разрешив им общаться между собой, *пока пользователь работает* со страницей.

Чтобы пояснить вышесказанное на простом примере, рассмотрим вебформу, куда пользователь должен внести некоторые сведения (напри-

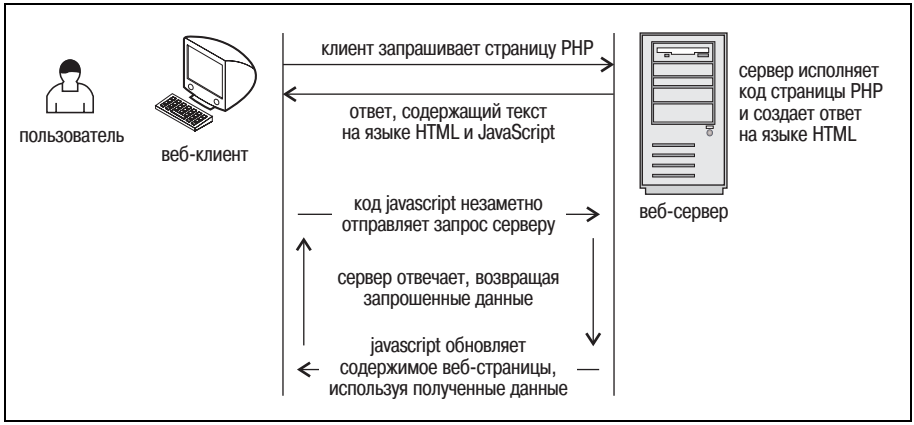

Рис. 1.4. *Типичный вызов AJAX*

мер, имя, адрес электронной почты, пароль, номер кредитной карточки и прочее). Прежде чем эта форма поступит в распоряжение вашего приложения, она должна быть проверена. В случае отказа от применения AJAX в нашем распоряжении имеются два способа проверки. Первый из них заключается в том, чтобы позволить ему или ей заполнить форму и *отправить* ее, после чего приложение проверит правильность заполнения на стороне сервера. В этом случае пользователь будет *терять время*, ожидая загрузки новой страницы. Другой вариант – выполнить проверку на стороне клиента, но это не всегда возможно, если для этого потребуется передать клиенту слишком большой объем данных (только представьте себе, что необходимо проверить правильность ввода названия города в соответствии с введенным названием страны).

Если же AJAX применяется, то веб-приложение может проверять данные, отправляя запросы серверу в фоновом режиме, по мере того как пользователь вводит их. Например, после того как пользователь выберет название страны, веб-броузер может запросить у сервера список городов этой страны, не отвлекая пользователя от его занятия. Пример проверки правильности заполнения формы с помощью технологии AJAX вы найдете в главе 4.

Рассказывать, где с успехом применяется AJAX, можно бесконечно. Более точное представление о возможностях, открываемых технологией AJAX, можно получить, взглянув на этот список реальных и достаточно популярных примеров:

- **Подсказки Google (Google Suggest)** помогут вам при работе с поисковой системой **Google**. Они представляют собой достаточно эффектное зрелище (просто зайдите по адресу *http://www.google.com/webhp?complete=1*). Аналогичная функциональность предлагается и **Yahoo! Instant Search**, которая доступна по адресу *http://instant.search.yahoo.com/*. (Как добиться этого, мы расскажем в главе 6.)

- **GMail (*http://www.gmail.com*).** Почтовая служба GMail пользуется заслуженной популярностью и не нуждается в представлении. Другие почтовые службы с веб-интерфейсом, такие как **Yahoo! Mail** и **Hotmail**, также наделены функциональностью технологии AJAX.
- **Google Maps (*http://maps.google.com*), Yahoo Maps (*http://maps.yahoo.com*)** и **Windows Live Local (*http://local.live.com*).**
- Прочие службы, такие как *http://www.writely.com* и *http://www.basecamphq.com*.

В этой книге вы встретите и другие примеры.

Примечание

AJAX, как и любая другая технология, может применяться *неправильно* или не *по назначению*. Если на вашем сайте применяется AJAX, то это еще не значит, что он станет лучше. Выигрыш, получаемый от технологии, зависит от того, насколько правильно и к месту вы ее используете.

Таким образом, технология AJAX служит для создания более гибких и интерактивных веб-приложений. Она позволяет выполнять асинхронные обращения к серверу, не прерывая работы пользователя и незаметно для него. AJAX – это инструмент, который может применяться разработчиками для создания веб-приложений, более интеллектуально взаимодействующих с человеком.

Технологии, из которых состоитAJAX, уже реализованы во всех современных веб-броузерах, таких как Mozilla Firefox, Internet Explorer или Opera. Таким образом, клиент не требует установки каких-либо дополнительных модулей, чтобы иметь возможность взаимодействия с веб-сайтами, построенными на основе AJAX. В состав AJAX входят следующие компоненты:

- JavaScript – основной ингредиент AJAX, позволяющий реализовать функциональность на стороне клиента. В ваших функциях JavaScript для манипулирования отдельными частями страницы HTML часто задействуется **объектная модель документа (Document Object Model – DOM)**.
- Объект **XMLHttpRequest** позволяет из JavaScript организовать асинхронный доступ к серверу, благодаря чему пользователь имеет возможность продолжать работу со страницей, в то время как она выполняет некоторые действия. Под доступом к серверу подразумеваются простые запросы HTTP на получение файлов или сценариев, размещенных на сервере. Запросы HTTP просты в исполнении и не вызывают каких-либо трудностей в случае применения брандмауэров.
- Серверные технологии, которые необходимы для обслуживания запросов, поступающих от JavaScript, со стороны клиента. В этой книге для выполнения действий на стороне сервера мы будем обращаться к PHP.

Для организации взаимодействия клиент-сервер необходимо иметь возможность передавать данные и понимать, что за данные были переданы. Передача данных – это самое простое. Сценарий на стороне клиента, обладающий доступом к серверу (посредством объекта `XMLHttpRequest`), может передавать серверу пары имя-значение с помощью методов **GET** или **POST**. Эти данные легко могут быть прочитаны с помощью любого сценария на стороне сервера.

Сценарий на стороне сервера просто отправляет свой ответ по протоколу HTTP, но, в отличие от обычного веб-сервера, ответ должен иметь такой формат, который легко может быть разобран кодом JavaScript на стороне клиента. Мы рекомендуем формат XML, который имеет свои преимущества, заключающиеся в том, что, во-первых, он получил широкое распространение и, во-вторых, существует большое количество библиотек, облегчающих работу с XML документами. Но при желании можно выбрать любой другой формат (данные могут передаваться даже в виде простого текста). Одна из известных альтернатив XML – **JavaScript Object Notation (JSON** – представление объектов в JavaScript).

Мы предполагаем, что вы уже знакомы со всеми составными частями AJAX, кроме, разве что, объекта `XMLHttpRequest`, который известен не так широко. Более подробно мы рассмотрим, как эти части работают и взаимодействуют между собой, в главах 2 и 3. А пока, до конца этой главы, мы сфокусируем свое внимание на рассмотрении общих принципов, а также, к большой радости самых нетерпеливых читателей, напишем программу с поддержкой AJAX.

Примечание

В AJAX нет ни одного нового или революционного компонента, как можно было бы подумать из-за шумихи, поднятой вокруг этой технологии, – все компоненты AJAX существуют, по крайней мере, с 1998 года. Название AJAX родилось в 2005 году в статье Джесси Джеймса Гаррета (Jesse James Garret, *http://www.adaptivepath.com/publications/essays/archives/000385.php*) и получило известность после того, как AJAX стал применяться компанией Google во многих собственных приложениях.

Зато с AJAX связано то новое обстоятельство, что впервые рынок аккумулировал достаточно энергии, чтобы поддержать стандартизацию и сфокусировать эту энергию на дальнейшем развитии. Как следствие, было разработано множество библиотек AJAX и появилось много веб-сайтов с поддержкой AJAX. Благодаря проекту Atlas, Microsoft также способствует развитию AJAX.

Применение технологии AJAX для создания новых веб-приложений дает нам следующие преимущества:

- Она позволяет создавать более динамичные и более качественные веб-сайты и веб-приложения.
- Высокая популярность способствует появлению шаблонных решений, которые помогут разработчикам не изобретать велосипед при решении наиболее распространенных задач.

- Она задействует уже существующие технологии.
- Позволяет разработчикам применять наработанные навыки.
- Функциональные возможности AJAX прекрасно интегрируются функциональностью, предоставляемой веб-броузерами (например, навигацией по странице, приведением размеров страницы к определенным значениям и т. д.).

Наиболее общие случаи применения AJAX:

- Проверка правильности заполнения формы с привлечением возможностей сервера, что очень удобно, когда невозможно заранее передать клиенту все данные, которые могут потребоваться в процессе проверки. Пример, в котором показано, как проверяется правильность заполнения формы, вы найдете в главе 4.
- Разработка простых чатов, которые не требуют наличия внешних библиотек, таких как виртуальная Java-машина или Flash. Подобную программу мы создадим в главе 5.
- Добавление функциональности, аналогичной подсказкам Google, пример мы разберем в главе 6.
- Более эффективное использование других технологий. В главе 7 мы реализуем приложение, которое динамически создает диаграммы с использованием **SVG (Scalable Vector Graphics** – масштабируемая векторная графика), а в главе 10 с помощью внешней библиотеки AJAX реализуем простейший список, поддерживающий возможность перетаскивания объектов мышью.
- Создание динамических **таблиц данных**, которые на лету обновляют базы данных на сервере. Подобное приложение мы создадим в главе 8.
- Разработка приложений, которые требуют обновления информации в режиме реального времени, считывая ее из различных внешних источников. В главе 9 мы создадим простейший **RSS**-агрегатор.

С AJAX связаны следующие потенциальные трудности:

- Поскольку адрес страницы в процессе ее работы не изменяется, добавить в закладки ссылку на страницу AJAX будет не так-то просто. В случае приложений AJAX установка закладки имеет иное значение, зависящее от конкретного приложения, т. е. обычно требуется сохранить текущее состояние (представьте себе, что работаете с обычной программой, ссылку на которую нельзя поместить в закладки).
- Поисковые системы могут оказаться не в состоянии проиндексировать все части вашего сайта, созданного на основе AJAX.
- Нажатие на кнопку «Назад» в броузерах не приводит к тому же результату, как в классических веб-приложениях, поскольку все действия пользователь выполняет в одной и той же странице.
- На стороне клиента JavaScript может быть отключен, что сделает приложения AJAX нефункциональными, поэтому неплохо бы пре-

дусмотреть на своем сайте альтернативные варианты страниц, чтобы не потерять потенциальных клиентов.

Наконец, прежде чем вы перейдете к написанию вашей первой программы с поддержкой AJAX, мы приведем ряд ссылок, которые помогут вам в вашем путешествии по увлекательному миру AJAX:

- *http://ajaxblog.com* – блог, посвященный AJAX.
- *http://www.fiftyfourleven.com/resources/programming/xmlhttprequest* – обширная коллекция статей об AJAX.
- *http://www.ajaxian.com* – сайт (с поддержкой AJAX) Бена Галбрайта (Ben Galbraith) и Диона Алмейра (Dion Almaer), авторов «Pragmatic AJAX».
- *http://www.ajaxmatters.com* – информационный сайт об AJAX, содержащий массу полезных ссылок.
- *http://ajaxpatterns.org* – рассказывает о шаблонах проектирования AJAX.
- *http://www.ajaxinfo.com* – ресурс содержит статьи об AJAX и полезные ссылки.
- *http://dev.fiaminga.com* – содержит массу ссылок на различные ресурсы и руководства, имеющие отношение к AJAX.
- *http://ajaxguru.blogspot.com* – популярный блог, посвященный AJAX.
- *http://www.sitepoint.com/article/remote-scripting-ajax* – замечательная статья Камерона Адамса (Cameron Adams) «Usable Interactivity with Remote Scripting».
- *http://developer.mozilla.org/en/docs/AJAX* – страница проекта Mozilla, посвященная AJAX.
- *http://en.wikipedia.org/wiki/AJAX* – страница об AJAX на Wikipedia.

По нашему мнению, этот список далеко не полный. Если вам потребуются сведения о других ресурсах, посвященных AJAX, в вашем распоряжении всегда имеется Google. В следующих главах мы дадим еще немало ссылок, но они будут касаться определенных технологий, о которых мы будем рассказывать.

Создание простого приложения на основе AJAX и PHP

А теперь приступим к написанию кода! На следующих страницах мы соберем простое приложение AJAX.

Примечание

Это упражнение для самых нетерпеливых читателей, которые стремятся приступить к программированию как можно скорее, однако вы уже должны быть знакомы с JavaScript, PHP и XML. Если это не так или в какой-то момент вы почувствуете, что это упражнение слишком сложно для вас, переходите к главе 2.

> В главах 2 и 3 мы поближе познакомимся с технологией AJAX и его методиками, и вам все станет понятно.

Здесь вы создадите простое веб-приложение под названием **quickstart**, применяя технологию AJAX. Оно запрашивает у пользователя имя и по мере ввода сервер возвращает клиенту полученные от него данные. На рис. 1.5 показана начальная страница `index.html`, загруженная пользователем. (Обратите внимание, страница `index.html` загружается по умолчанию, при обращении к веб-каталогу `quickstart`, даже если имя файла не указано явно.)

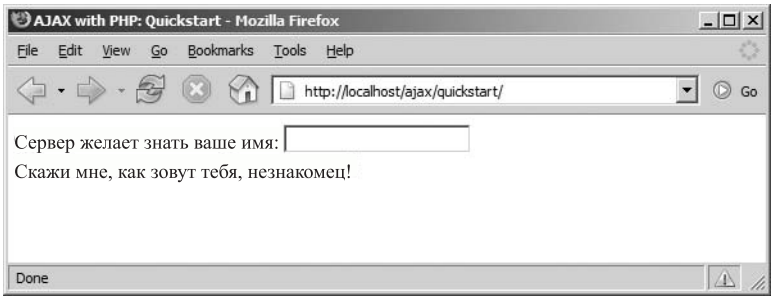

Рис. 1.5. *Начальная страница вашего приложения Quickstart*

Пока пользователь вводит свое имя, через регулярные интервалы времени производятся обращения к серверу, чтобы он мог опознать текущее имя. Сервер вызывается автоматически, примерно один раз в секунду, и поэтому на форме отсутствует кнопка «Передать» (Submit), с помощью которой производится передача данных серверу в классических веб-приложениях. (Такой метод плохо подходит для реализации механизмов авторизации пользователей, но он прекрасно демонстрирует некоторые возможности AJAX.)

В зависимости от того, какое имя было введено, сообщения, получаемые от сервера, могут отличаться (рис. 1.6).

Проверить работу этого приложения вы сможете, зайдя стандартным броузером по адресу *http://ajaxphp.packtpub.com/ajax/quickstart*.

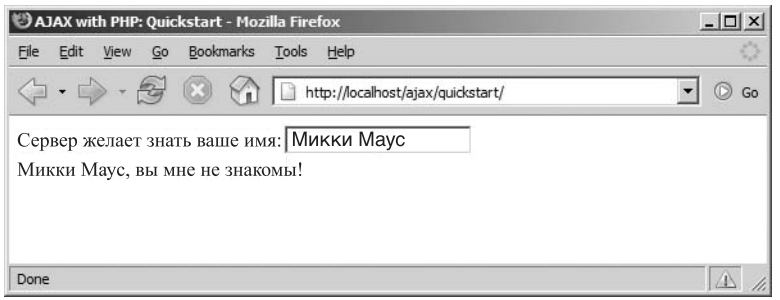

Рис. 1.6. *Пользователь получает ответ от сервера*

На первый взгляд в этом приложении нет ничего экстраординарного. Мы нарочно сохранили простоту первого примера, чтобы вам было проще разобраться в нем. Примечательно в этом приложении то, что отображаемое сообщение автоматически поступает от сервера, не прерывая действий пользователя. (Сообщение появляется сразу же, как только пользователь начинает вводить свое имя.) **Для того чтобы отобразить сообщение, приложение не производит повторную загрузку всей страницы, даже несмотря на то, что за получением текста сообщения необходимо обращаться к серверу.** Без помощи AJAX достигнуть такого эффекта очень сложно.

Приложение состоит из трех файлов:

1. `index.html` – файл HTML, который изначально запрашивает пользователь.
2. `quickstart.js` – файл, содержащий код JavaScript, который загружается клиентом вместе с `index.html`. Этот файл отвечает за выполнение асинхронных обращений к серверу, когда становится необходимой его функциональность.
3. `quickstart.php` – сценарий PHP, постоянно находящийся на сервере. Этот сценарий вызывается со стороны клиента из кода JavaScript, находящегося в файле `quickstart.js`.

На рис. 1.7 показана последовательность действий, выполняемых приложением.

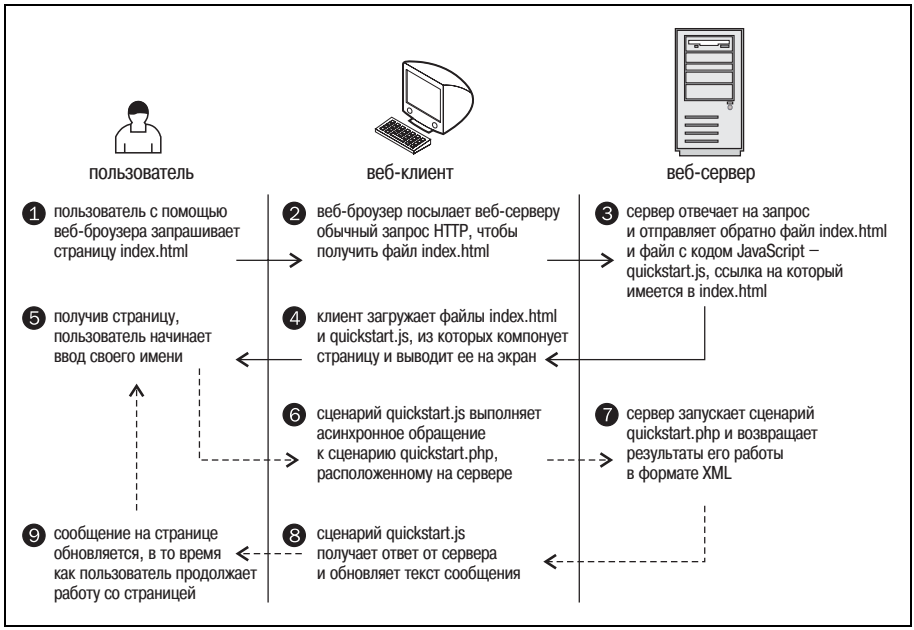

***Рис. 1.7.** Диаграмма, описывающая процессы, протекающие внутри приложения Quickstart*

Шаги с 1 по 5 – это типичный запрос HTTP. После того как запрос будет послан, пользователю придется подождать, пока загрузится страница. В случае типичных веб-приложений (не использующих технологию AJAX) такая загрузка страницы происходит всякий раз, когда клиент пытается получить новые данные от сервера.

Шаги с 5 по 9 демонстрируют запрос AJAX, а если точнее, то целую серию асинхронных запросов HTTP. Доступ к серверу производится в фоновом режиме, с помощью объекта `XMLHttpRequest`. В это время пользователь может продолжать нормальную работу со страницей, как если бы это было обычное приложение. Чтобы получить новые данные от сервера и обновить текст сообщения на странице, не надо загружать страницу повторно.

Теперь пора реализовать этот код на вашей машине. Прежде чем двинуться дальше, убедитесь, что ваше рабочее окружение настроено так, как описано в приложении A, где рассказано, как установить и настроить PHP и Apache и как настроить базу данных, фигурирующую в примерах из этой книги. (Для реализации примера quickstart база данных не нужна.)

Примечание

Все примеры в этой книге предполагают, что ваше рабочее окружение настроено так, как описано в приложении A. Если это не так, то вам скорее всего придется вносить некоторые изменения, например, изменить названия каталогов и тому подобное.

Время действовать – Quickstart AJAX

1. В приложении A вы найдете инструкции по установке и настройке веб-сервера и созданию каталога, доступного через Интернет, с именем `ajax`, в котором будут размещаться все исходные тексты примеров из этой книги. В каталоге `ajax` создайте каталог `quickstart`.
2. В каталоге `quckstart` создайте файл с именем `index.html` и добавьте в него следующий код:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>AJAX и PHP: Quickstart</title>
        <script type="text/javascript" src="quickstart.js"></script>
    </head>
    <body onload='process()'>
        Сервер желает узнать ваше имя:
        <input type="text" id="myName" />
        <div id="divMessage" />
    </body>
</html>
```

3. **Создайте новый файл с именем** `quickstart.js` **и добавьте в него следующий код:**

```
// запомнить ссылку на объект XMLHttpRequest
var xmlHttp = createXmlHttpRequestObject();

// создать объект XMLHttpRequest
function createXmlHttpRequestObject()
{
    // для хранения ссылки на объект XMLHttpRequest
    var xmlHttp;
    // если сценарий запущен под управлением Internet Explorer
    if(window.ActiveXObject)
    {
        try
        {
            xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (e)
        {
            xmlHttp = false;
        }
    }
    // если сценарий запущен под управлением Mozilla или другого броузера
    else
    {
        try
        {
            xmlHttp = new XMLHttpRequest();
        }
        catch (e)
        {
            xmlHttp = false;
        }
    }
    // вернуть созданный объект или вывести сообщение об ошибке
    if (!xmlHttp)
        alert("Ошибка создания объекта XMLHttpRequest.");
    else
        return xmlHttp;
}

// выполнить асинхронный запрос HTTP с помощью объекта XMLHttpRequest
function process()
{
    // работа возможна только если объект xmlHttp не занят
    if (xmlHttp.readyState == 4 || xmlHttp.readyState == 0)
    {
        // получить имя, введенное пользователем в форму
        name = encodeURIComponent(document.getElementById("myName").value);
        // обратиться к сценарию quickstart.php на сервере
        xmlHttp.open("GET", "quickstart.php?name=" + name, true);
```