Первые несколько страниц — пустые

Так надо :-)

PHP. ТРЮКИ

100 советов и рекомендаций профессионалов

O'REILLY®

ПИТЕР®

ДжекД. Харрингтон

Краткое содержание

Оглавление

Об авторе

Джек Д. Харрингтон — программист, который уже 25 лет занимается разработкой приложений. За эти годы он создавал приложения на многих ведущих языках программирования и работал во многих средах разработки.

Он является автором трех книг. В книге «Генерирование кода в действии» (Code Generation in Action, Manning, 2002) описывается использование кода для автоматического создания программ. Некоторая часть разработок, описанных в книге «Генерирование кода в действии», также рассматривается в издании, которое вы держите в руках. В книге «Пакетная обработка данных. Трюки» (Podcasting Hacks, O'Reilly, 2005) он призывает своих читателей создавать информационные ресурсы с помощью пакетной обработки данных. Его третью книгу — «PHP. Трюки» (PHP Hacks, O'Reilly, 2006) - вы держите в руках.

Джек Д. Харрингтон также является автором более чем 30 статей на различные темы: PHP, создание приложений с помощью пакетной обработки данных, цифровая фотография и др. В одной статье, выпущенной издательством O'Reilly, Джек смело предположил, что с помощью PHP можно развивать крупные приложения, созданные для предприятий на Java или .NET. После выхода этой статьи развернулись баталии на новостном сайте Slashdot, которые не прекращаются даже сегодня.

Джек вместе со своей женой Лори и дочерью Меган живет в Сан-Франциско. Он работает в молодой компании Leverage Software, которая специализируется на создании программного обеспечения для сетевого использования. Перед этим он работал с компанией Macromedia.

В свободное от работы время Джек является заядлым путешественником, игроком в гольф, поваром, столяром и специалистом по оригами.

Соавторы

Следующие люди помогали автору в создании трюков, описываемых в этой книге.

Росс Шеннон (Ross Shannon) — студент из Дублина, в настоящее время учится на доктора философии по информатике в самом большом вузе Ирландии — Университетском колледже Дублина (University College Dublin). Большую часть времени Росс занимается веб-дизайном, что делает с большим удовольствием. Он является разработчиком обучающего сайта о языке HTML, который вы можете найти по адресу http://www.yourhtmlsource.com.

Мэттью Теренцио (Matthew Terenzio) — имеет более 10 лет стажа в разработке технологий и средств информации. Он является дипломированным специалистом

в создании интернет-приложений, был ведущим разработчиком многочисленных проектов для таких организаций, как The Berkman Center for Internet и Society at Harvard Law School.

В течение пяти лет Мэтт занимает должность старшего веб-разработчика сайтов GreenwichTime.com и StamfordAdvocate.com, а также внес свой вклад в создание нескольких многопрофильных новостных сайтов, включая NYNewsday.com и OrlandoSentinel.com.

Недавно Мэтт основал компанию BuddyBuilder LLC, которая выпустила множество сайтов на основе Web 2.0, среди которых BuddyBuilder.com, SkinnyFarm.com и Newsmarks.com.

Майкл Маллиган (Michael Mulligan) — разработчик программного обеспечения, который получил диплом в Корнелльском университете на факультете радиоэлектроники (College of Engineering at Cornell University) в 2005 году. Он работал на многих предприятиях, занимающихся разработкой программного обеспечения для компьютеров Apple. Свои главные исследования Майкл проводит в области дистанционного обучения. Его электронный адрес — mtm26@cornell.edu.

Летом 2005 года Майкл женился, вместе со своей прекрасной женой Дифти Девабоз (Dhipthi Devabose) он живет во Флориде. Сейчас все свое время он отдает разработке программного обеспечения для компании Lockheed Martin. Кроме того, он является автором интернет-ресурса myPhoto (http://agent0068.dyndns.org/~mike/projects/myPhoto/). В свободное время Майкл любит готовить и играть со своим щенком Сиеной.

Дрю Нельсон (Dru Nelson) занимается интернет-проектами с 1988 года. Начав в организации ISP (Internet service provider) во Флориде, он переехал в Сан-Франциско и участвовал в основании таких ресурсов, как Four11 («Почта Yahoo!»), Diva, eGroups («Группы Yahoo!»), Danger и Blue6. Сейчас он работает в Plaxo.com, разрабатывает программное обеспечение для Win32. У Дрю есть блог, который вы найдете по адресу http://www.xxeo.com/.

Тайлер Митчелл (Tyler Mitchell) — географ и первооткрыватель. Он является автором книги «Иллюстрированная веб-картография» (Web Mapping Illustrated, O'Reilly, 2005). Тайлер работает в области Geographic Information Systems (GIS) менеджером в компании Timberline Forest Inventory Consultants и живет в прекрасной канадской провинции Британская Колумбия. Он также является постоянным советчиком, модератором и участником семинаров в конференциях GIS.

Питер Лэвин (Peter Lavin) начинал как веб-разработчик в Торонто. Он публиковался в некоторых журналах и интернет-ресурсах, включая такие, как UnixReview.com и Dr.Dobb's Journal. Недавно Питер написал книгу об использовании объектно-ориентированного PHP, которая была издана компанией No Starch Press. Более полную информацию вы можете получить на сайте http://softcoded.com/.

Предисловие

PHP заслуженно считается одним из ведущих языков написания веб-сценариев и используется везде, начиная с небольших сценариев-утилит и заканчивая объектно-ориентированными корпоративными приложениями. В этой книге раскрывается весь спектр такого рода задач и предлагаются трюки, которые можно использовать во всех областях, как в HTML и Ajax, так и для генерации кода и организации очереди сообщений на основе баз данных.

Мы написали тексты программ и выбрали материалы самых передовых авторов из среды веб-разработки, программирования, графики и мультимедиа. В книге подробно рассматривается динамический HTML, который позволяет клиентам пользоваться интерактивными возможностями веб-страницы, не обновляя ее в браузере. Кроме того, вы научитесь генерировать флэш-ролики на лету, даже узнаете, как использовать PHP для доступа к базам данных, узнаете о веб-сервисах и многом другом.

Эта книга является чем-то большим, чем просто готовые решения. В ней предлагаются идеи и приемы, которые вы можете использовать в своих приложениях. И зачем тогда останавливаться на достигнутом? Мы предлагаем вам использовать предоставленные здесь идеи и развивать их. «Хакайте» наши трюки, развивая ваши сценарии и классы как можно эффективнее.

Почему PHP-трюки?

Термин *«хакинг»* в прессе пользуется плохой репутацией. Он применяется для обозначения такого типа людей, которые взламывают системы или причиняют какой-либо вред, используя компьютер как оружие. Однако среди людей, которые занимаются разработкой приложений, термин *«хак»* означает «быстрое-но-грязное» решение поставленной задачи или хитрый способ что-либо сделать. И слово *«хакер»* зачастую применяется как комплимент, означающий, что у человека есть *творческий* подход и техническая подкованность в решении поставленных задач. Описанный набор трюков («хаков») — попытка реабилитировать это слово, показать, что есть случаи, когда люди «хакают» в хороших целях, и раскрыть для непосвященных творческую сторону хакеров. Наблюдение за тем, как другие подходят к решению проблем, зачастую является наиболее быстрым способом обучиться какой-либо новой технологии.

Как пользоваться данной книгой

Данную книгу можно читать, как многим нравится, от корки до корки, но все описанные трюки не зависят друг от друга, так что вы можете спокойно пролистывать целые разделы и переходить к тем главам, которые являются, на ваш взгляд,

наиболее интересными. Если в каком-то случае окажется, что есть информация, с которой вам надо было ознакомиться заранее, то в этом помогут ссылки, по которым вы сможете найти нужный трюк.

Структура книги

Эта книга разделена на несколько тематических глав.

- **Глава 1. Установка и начало работы.** В этой главе будет рассказано об установке PHP и MySQL, а также о том, как пользоваться крайне полезной библиотекой PEAR.
- **Глава 2. Веб-дизайн.** Прочитав эту главу, вы узнаете о приемах использования HTML вместе с PHP для создания интерактивно-притягательного интерфейса.
- **Глава 3. DHTML.** В этой главе используется мощная комбинация HTML, CSS и JavaScript, известная как динамический HTML (DHTML), рассматриваемая в работе вместе с PHP для демонстрации ее возможностей в браузере.
- **Глава 4. Графика.** Эта глава освещает широкие возможности методов отображения данных в графической форме.
- **Глава 5. Базы данных и XML.** Базы данных являются крайне важной составляющей частью PHP-приложений. В этой главе будет показано, как создавать объекты гибких баз данных и даже автоматически добавлять свои собственные слои баз данных при помощи генерации кода.
- **Глава 6. Дизайн приложений.** Эта глава посвящена более детальному обсуждению техники быстрой и надежной разработки приложений.
- **Глава 7. Шаблоны.** Программисты, работающие на C++, C# и Java, многие годы использовали дизайнерские шаблоны. Можно ли их так же использовать в PHP? Будьте уверены в этом. В этой главе показывается, как использовать некоторые из *дизайнерских шаблонов* из книги Эдисона Уэсли «Дизайнерские шаблоны» для создания более качественных PHP-приложений.
- **Глава 8. Тестирование.** Вы не спите по ночам, думая о том, функционирует ли ваше PHP-приложение? Эта глава посвящена технологиям тестирования, которые найдут ошибки за вас, а также будут постоянно следить за работой вашего сайта.
- **Глава 9. Альтернативные пользовательские интерфейсы.** В этой главе показано, как пользоваться различными пользовательскими интерфейсами для работы с PHP-приложением. Вы можете запускать приложение с Рабочего стола, используя мобильный телефон или программы для немедленной передачи текстовых сообщений.
- **Глава 10. Забавные возможности.** Прочитав эту главу, вы узнаете, как все написанное ранее разместить в Интернете и, используя различные любопытные возможности Сети, отследить многопользовательские игры. Здесь также описывается, как использовать в приложениях карты Google и многое другое.

Общие проблемы

В связи с тем что часто возникают некоторого роды проблемы с PHP-приложениями, в данной книге описаны некоторые меры по их устранению.

Плохая модель базы данных

Большинство PHP-приложений работают с реляционными базами данных, в основном MySQL. Для большинства разработчиков, изучавших традиционные языки программирования, моделирование базы данных не является чем-то само собой разумеющимся и простым в разработке. Первым шагом при доведении программы до ума является проверка качества моделирования базы данных (см. трюк 34).

Недостаточно полное использование возможностей баз данных

В PHP есть несколько различных возможностей доступа к базам данных, и недостаточно полное их использование может стать причиной серьезных проблем безопасности. Исправление уровня доступа к базе данных можно начать с перехода к PEAR DB или PDO (см. трюк 35). После этого вы можете оценить, использовать SQL (см. трюк 41), команды SELECT (см. трюк 42) или CRUD (см. трюк 37).

Интегрирование текста программы в страницу

Следующая проблема, с которой я сталкиваюсь довольно часто, — встраивание текста программы прямо в страницу. В частности, доступ к базе данных встраивается прямо в исходный текст страницы. В трюке для генерации кода программы, создающем участки кода с использованием команды SELECT (см. трюк 42) и CRUD (см. трюк 37), показан пример правильной двухуровневой модели в PHP. Трюк с использованием динамического SQL-объекта также показывает, как избежать встраивания в страницу кода для доступа к SQL.

Обработка данных во время создания страницы

Еще одной ошибкой, с которой я сталкивался, является попытка программы выполнить большое количество операций на веб-сервере во время загрузки страницы. Примером может служить приложение, которому требуется отправить большое количество сообщений по электронной почте в ответ на запрос пользователя. Часто это встречается на страницах, ожидающих какого-либо действия от пользователя, что заставляет его в итоге ждать загрузки страницы, в то время как система отправляет кипу электронной почты. Одним из лучших решений данной проблемы является использование очереди сообщений (см. трюк 50).

Отсутствие тестирования

Я практически никогда не встречал в приложениях текста программы, не предназначенного для тестирования. Однако одним из лучших способов чувствовать себя комфортно, возвращаясь ночью домой, является использование автоматизированного тестирования. Это особенно актуально в тех случаях, когда ваша работа заключается в написании интернет-приложения, работающего 24 часа в сутки

семь дней в неделю. В этой книге содержится информация о модуле, предназначенном для тестирования (см. трюк 79), и о том, как генерировать такое тестирование автоматически (см. трюк 80). Здесь вы также найдете текст программы для проверки сайта при помощи роботов (см. трюк 83) и с использованием модуля автоматизации в Internet Explorer (см. трюк 82), который может даже проверить текст вашего приложения, написанного на JavaScript.

Обеспечение большей безопасности пользователей

Многие используют один и тот же пароль для доступа к большинству своих учетных записей. Если в вашем приложении пароли хранятся в открытом виде и оно, возможно, содержит уязвимости, то вы рискуете позволить всем и каждому узнать пароль вашего пользователя. Используйте MD5 для шифрования паролей пользователей (см. трюк 59), а также используйте разграничение доступа (см. трюк 58), чтобы быть уверенным в том, что пользователи не имеют доступа к тому, к чему не положено.

Более активное использование шаблонов

Я допускаю, что *дизайнерским шаблонам* придается слишком большое значение. Но в них содержится большое количество полезностей, которых не хватает PHP-приложениям. Вся глава 7 посвящена тому, как практично и эффективно использовать дизайнерские шаблоны для создания качественных PHP-приложений. Это всего лишь несколько идей о том, как улучшить ваше уже существующее интернет-приложение, чтобы оно стало более надежным и безопасным. Но как насчет создания продвинутых приложений?

Создание отличного интернет-приложения

Возможности браузеров, плагинов для браузеров, DHTML и Ajax настолько велики, что создание непродвинутого, неудобного в использовании и не обладающего самыми современными возможностями приложения практически нереально. Вот только несколько идей, взятых из различных частей книги.

- **Использование Рабочего стола.** Верите вы или нет, вы можете использовать PHP при создании приложений для Рабочего стола (см. трюк 87), причем в нем будет использоваться тот же программный код, реализующий функциональность приложения, что и в приложении, запущенном на вашем веб-сервере. Еще лучше то, что этот программный код может быть портирован на Mac OS X, Windows и Linux с минимальными изменениями (а зачастую и вообще без них).
- **Использование карт.** Этот метод в последнее время становится все более популярным. Есть два простых способа реализовать его на PHP: с использованием MapServer (см. трюк 86) или карт Google (см. трюк 95).
- **Использование динамической графики.** Гистограммы и отображение графической информации всегда были очень популярны, и в PHP для их использова-

ния заложены огромные возможности. Можно использовать обычный HTML (см. трюк 8), SVG (см. трюк 28), динамический HTML (см. трюк 22) и библиотеку GD (см. трюк 31). Здесь также содержится информация о создании объектно-ориентированной надстройки на базе графической библиотеки (см. трюк 29).

J **Работа с пользовательскими приложениями.** Еще один способ создания более конкурентоспособных приложений — активное общение с вашими пользователями. RSS (см. трюк 88) является одним из наиболее распространенных способов реализации такого рода задумок. Я даже использую RSS на моей портативной приставке PlayStation (см. трюк 90). Вы же можете пользоваться более традиционными способами, например электронной почтой (см. трюк 10). Я также добавил кое-какую информацию о том, как генерировать RTF-документы в приложении Microsoft Word (см. трюк 48), таблицы Microsoft Excel (см. трюк 49), а также как использовать документы Microsoft Word (см. трюк 47) или Microsoft Excel (см. трюк 45) как объекты для входной информации.

- **Улучшение веб-интерфейса.** Я также добавил некоторые идеи по созданию динамических меню (см. трюк 17), простых ссылочных структур сайтов (см. трюк 4), вкладок (см. трюк 6), интерфейсов с использованием обложек (см. трюк 3), всплывающих окон (см. трюк 12), раскрывающихся вкладок (см. трюк 16). Кроме того, вы узнаете, как эффективнее использовать технологию Drag & Drop (см. трюк 13), календари (см. трюк 25), графики (см. трюк 24) и многое другое, что сделает ваш веб-интерфейс лучшим.

Это всего лишь несколько идей со страниц данной книги. Загляните в нее дальше и станьте одним из ведущих PHP-разработчиков.

Соглашения, использованные в данной книге

Здесь приводится список соглашений, использованных в данной книге.

Шрифт **для названий**

Применяется для отображения URL, а также названий папок и выводимой на экран информации.

Шрифт для команд

Используется для имен файлов, названий путей, имен переменных и команд. Для примера путь будет выглядеть так: /Developer/Applications.

`Шрифт с постоянной шириной`

Применяется для отображения примеров исходного кода и содержимого файлов.

`Полужирный шрифт с постоянной шириной`

Используется для выделения кода, добавленного в старый код.

`Курсив с постоянной шириной`

Используется в примерах кода и таблиц для отображения тех участков кода, которые нужно заменить вашим собственным.

Вам следует обращать особое внимание на специальные заметки, выделенные основным текстом при помощи следующих рисунков.

ПРИМЕЧАНИЕ

Это подсказка, пожелание, заметка общего типа. Она содержит полезную прикладную информацию по рассматриваемой теме.

ВНИМАНИЕ

Это предостережение или указание, говорящее о том, что вам необходимо быть внимательным. Оно часто указывает на то, что ваши деньги или ваша частная информация могут оказаться под угрозой.

Изображение в форме термометра, присутствующее сразу после каждого трюка, отображает его сложность.

Использование примеров кода

Эта книга предназначена для того, чтобы помочь вам выполнить вашу работу. В целом вы можете использовать исходные коды, встречающиеся в данной книге, в ваших программах и документации. Нет необходимости связываться с нами для получения разрешения на их использование, кроме тех случаев, когда вы пользуетесь существенными участками кода. Например, выборка нескольких фрагментов исходного кода для написания программы не является нарушением. Но продажа или распространение компакт-дисков с примерами из книги издательства O'Reilly *является нарушением*. Использование ссылок и цитат из данной книги не требует какого-либо специального разрешения. Но для включения значительных участков кода в вашу программную документацию вам *необходимо* получить разрешение.

Мы ценим, но не требуем ссылок на авторство. Это обычно подразумевает под собой указание названия, автора, издателя, а также ISBN. Например: «PHP. Трюки», автор Джек Д. Херрингтон. Copyright 2006 O'Reilly Media, Inc., 0-596-10139-2.

Если вы не уверены в разрешении использования исходных кодов из книги в той или иной ситуации, то без проблем можете связаться с нами по адресу permissions@oreilly.com.

Как с нами связаться

Мы протестировали и проверили информацию в данной книге настолько тщательно, насколько это было в наших силах, но, возможно, вы заметите, что произошли некоторые изменения (или даже обнаружите, что мы сделали ошибки!). Будучи читателем данной книги, вы можете помочь нам сделать будущие изда-

ния лучше, выслав нам информацию об ошибках. Пожалуйста, дайте нам знать о каких-либо ошибках, неточностях, недостатках, вводящих в заблуждение или сбивающих с толку утверждениях, а также о каких-либо опечатках в данной книге.

Пожалуйста, высылайте нам также соображения о том, как сделать эту книгу более полезной для вас. Мы внимательно изучим ваши замечания и попытаемся учесть их в будущих изданиях. Вы можете писать нам по следующему адресу.

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

(800) 998-9938 (в U.S. или Canada)

(707) 829-0515 (международный/местный)

(707) 829-0104 (факс)

Чтобы задать какие-либо технические вопросы или высказать замечания по данной книге, присылайте нам письма по адресу bookquestions@oreilly.com.

На сайте «PHP-трюков» перечислены примеры, опечатки, а также планы по поводу будущих изданий. Вы можете найти эту информацию по адресу http://www.oreilly.com/catalog/phphks.

Для получения большей информации о данной книге, а также о других книгах посетите сайт O'Reilly: http://www.oreilly.com

Сами разработали трюк?

Если хотите предложить какой-либо новый, разработанный вами трюк, который можно будет добавить в последующие книги, то можете просмотреть книгу трюков в Интернете по адресу http://hacks.oreilly.com

Safari Enabled

Когда вы видите рисунок Safari® Enabled на обложке вашей любимой технической книги, это значит, что вы можете получить доступ к этой книге в Интернете при помощи электронной библиотеки O'Reilly Safari.

Safari предлагает более продвинутую систему, чем электронные книги. Это виртуальная библиотека, которая позволяет вам проводить поиск по тысячам лучших технических книг, копировать фрагменты примеров исходного кода, загружать целые главы, а также находить быстрые ответы на ваши вопросы, когда вам нужна наиболее точная и современная информация по данной теме. Воспользуйтесь бесплатно этой библиотекой по адресу http://safari.oreilly.com.

От издательства

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты dgurski@minsk.piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

Все исходные коды, приведенные в книге, вы можете найти по адресу http://www.piter.com/download.

На сайте издательства http://www.piter.com вы найдете подробную информацию о наших книгах.

Установка и начало работы

Трюки 1-2

Прежде чем вы начнете использовать PHP-трюки, вам необходимо либо установить PHP, либо получить учетную запись на машине, на которой он уже установлен. В этой главе рассказано о том, как установить PHP, некоторые важные дополнения к нему, а также СУБД MySQL, используемую в качестве источника данных в PHP-приложениях. Здесь также рассказывается об установке модулей PEAR с открытым кодом, которые вы можете свободно использовать в ваших приложениях на PHP.

Установка PHP

Установка языка PHP под Windows, Mas OS X и Linux для веб-серверов Apache и Internet Information Server.

Первое, что вам необходимо сделать для начала работы с данной книгой, — установить PHP, что для большинства операционных систем довольно просто выполнить. Установка PHP начинается с посещения сайта PHP (http://www.php.net/) и загрузки либо исходных текстов PHP, либо исполняемых файлов, а также документации.

Установка PHP для Windows

Для установки PHP под Windows вам необходимо загрузить исполняемые файлы PHP версии 5. Для большей простоты используйте установку в формате MSI, а в качестве пути установки укажите директорию c:\php5. После успешной установки вы можете запускать интерпретатор PHP из командной строки под Windows:

```
C:\> php -v
PHP 5.0.4 (cli) (built: Mar 31 2005 02:45:00)
Copyright © 1997-2004 The PHP Group
Zend Engine v2.0.4-dev, Copyright (c) 1998-2004 Zend Technologies
```

Если исполняемый PHP-файл не найден, то вам необходимо добавить путь c:\php5\bi n в переменные среды. В Панели управления выберите пункт Система, перейдите на вкладку Дополнительно и нажмите кнопку Переменные среды. Отредактируйте переменную Path, добавив к ее значению c:\php5\bin.

ПРИМЕЧАНИЕ

Необходимо закрыть все окна с командной строкой, а затем открыть новое, чтобы убедиться, что изменения были приняты.

Доступ к PHP через командную строку — это, конечно, хорошо, но хочется, чтобы PHP был установлен и интегрирован прямо на веб-сервер. В Windows есть два способа сделать это. Первый — установить веб-сервер Apache и настроить PHP для него, второй — установить веб-сервер Internet Information Services (IIS) и интегрировать в него PHP.

В обоих случаях вам понадобится скопировать файл `php.ini` в директорию с Windows (`c:\windows`). Отредактируйте его, изменив строку `extension_dir`, чтобы она выглядела так:

```
extension_dir ="c:\php5\ext"
```

Далее раскомментируйте строки вида:

```
extension=php_mysql.dll
```

Эта строка позволяет получать доступ к базе данных MySQL.

ПРИМЕЧАНИЕ

Возможно, вам понадобится раскомментировать несколько других библиотек в этом файле, чтобы получить к ним доступ. Изучите для этого документацию PHP.

Теперь вновь перейдите к сайту PHP (http://www.php.net/) и загрузите PECL-модули. Сохраните эти DLL-файлы в директории `c:\php5\ext` (той самой директории, ссылку на которую вы прописали в файле php.ini). Эти дополнительные модули необходимы для получения доступа к базе данных SQL или для работы с графикой (рано или поздно вам понадобится и то, и другое).

Установка PHP для Apache

Перейдите на сайт Apache (http://www.apache) и загрузите Apache версии 1.3, который уже скомпилирован для Windows. Он поставляется в виде MSI-инсталлятора, и это самый простой способ установить Apache. После установки необходимо отредактировать файл http.conf в директории conf (`c:\Program Files\Apache Group\Apache\conf`, если Apache был установлен по указанному по умолчанию пути).

Добавьте следующие строки в конце файла httpd.conf:

```
LoadModule php5_module "c:/php5/php5apache.dll"
AddModule mod_php5.c
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```

Далее запустите Apache, используя файл `apache.exe`:

```
C:\Program FilesXApache Group\Apache>
apache Apache/1.3.33 (Win32) PHP/5.0.4 running...
```

Директория, в которой хранятся документы, носит название **htdocs** (полный путь - :: \Program Fi les\Apache Group\Apache\htdocs). Для проверки создайте файл test.php в директории **htdocs** и поместите в него такой текст:

```
<?php phpinfoO:
>
```

Используя для перехода к странице браузер, можно наблюдать что-то похожее на рис. 1.1.

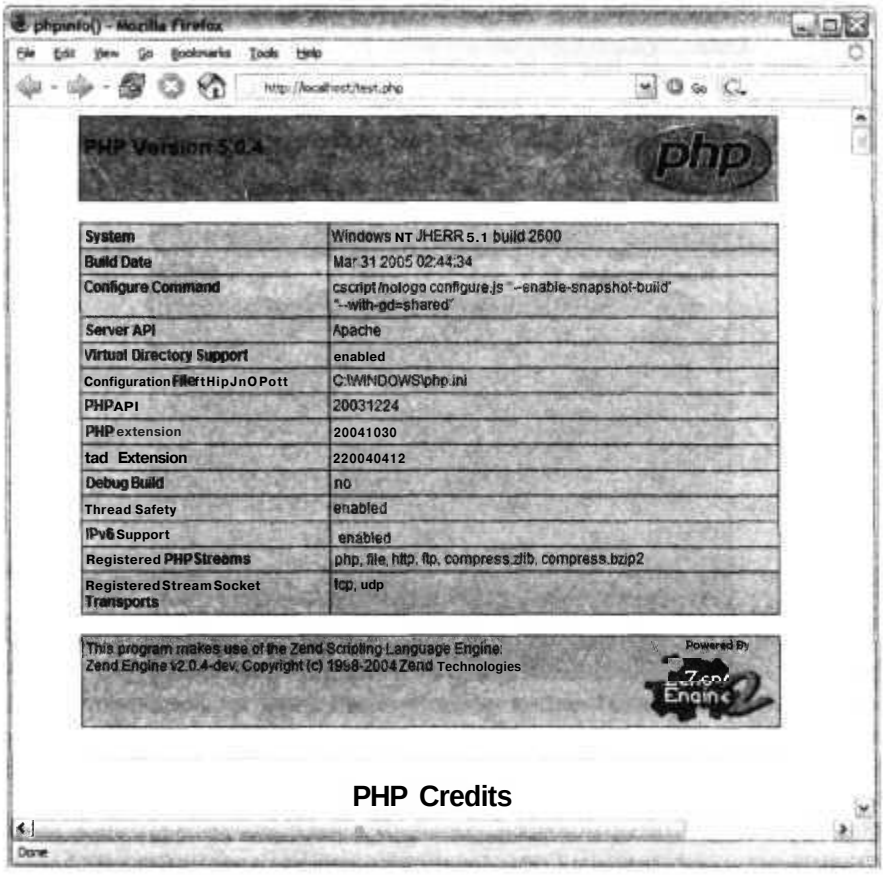


Рис. 1.1. Тестовая страница PHP для Apache/Windows

Из указанной директории вы можете запускать все трюки, описанные в данной книге.

Установка PHP для IIS

После установки PHP в директорию c:\php5 вы можете интегрировать его в IIS при помощи библиотеки php5isapi.dll. Для начала откройте панель управления IIS. Далее создайте новую виртуальную директорию, как это показано на рис. 1.2.

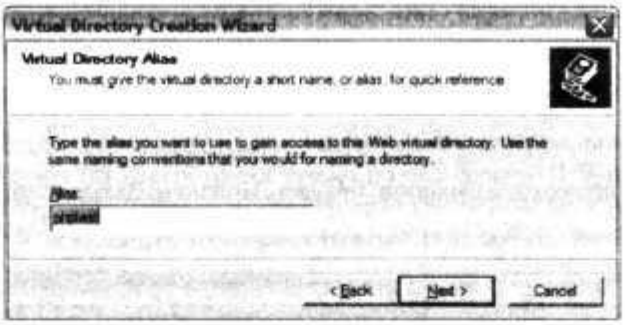


Рис. 1.2. Создание виртуальной директории

Проследите за тем, чтобы были корректно установлены права на выполнение (рис. 1.3).

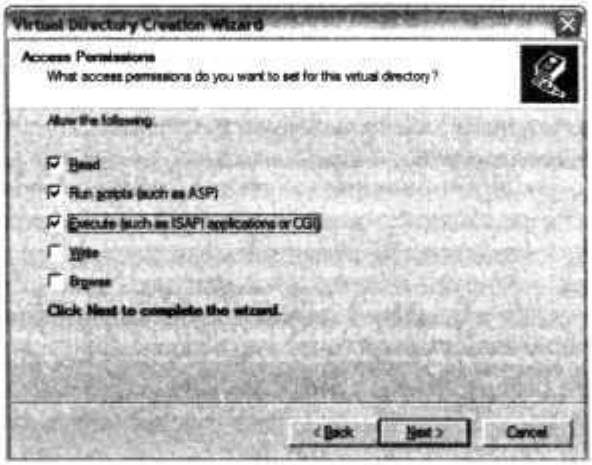


Рис. 1.3. Установка разрешения на выполнение для виртуальной директории

Далее щелкните правой кнопкой мыши на виртуальной директории и в контекстном меню выберите пункт **Properties.** В открывшемся окне нажмите кнопку Configuration. В результате откроется окно **Application Mappings,** в котором вы можете поставить в соответствие файлам с расширением PHP библиотеку php5i `sapi.dll`. Пример такого окна показан на рис. 1.4.

Нажмите кнопку **Add** для сохранения конфигурации и сделайте исполняемым файл `c:\php5\php5isapi.dll`.

ПРИМЕЧАНИЕ

Если при создании конфигурации вы пользуетесь кнопкой Browse, то в раскрывающемся списке измените тип файлов на DLL, чтобы нужный вам файл отображался.

Измените расширение на PHP. Результат показан на рис. 1.5.

Рис. 1.4. Окно Application Mappings для подключения PHP-файлов

Рис. 1.5. Настройка конфигурации для PHP 5

Для подтверждения всех изменений нажмите кнопку OK. Затем перейдите к папке с документами, которую вы указали при создании виртуальной директории. Создайте новый файл с названием test.php со следующим содержанием:

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

Затем перейдите в вашем браузере к **localhost (см.** рис. **1.1).**

Установка PHP для Mac OS X

PHP уже установлен для всех версий OS X. Все, что вам надо, — запустить его. Для начала получите права Superuser при помощи команды sudo:

```
X sudo tcsh
```

При статусе Superaser в командной строке вы можете изменять системные файлы. Измените содержимое файла httpd.conf в директории /etc/httpd, используя

любой текстовый редактор на выбор (vi, Emacs и т. д.). Найдите и закомментируйте следующую строку:

```
LoadModule php4_module          libexec/httpd/libphp4.so
```

Далее раскомментируйте строку:

```
AddModule mod_php4.c
```

Затем сохраните изменения и перезапустите встроенный сервер Apache:

```
% apachectl restart
```

Для документов веб-сервера Apache для Mac OS X установлена по умолчанию директория `/Library/WebServer/Documents`. Для проверки работоспособности PHP создайте в этой директории следующий тестовый сценарий:

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

И наконец, перейдите в браузере к тестовой странице, в которой должно отразиться окно статуса PHP (рис. 1.6).

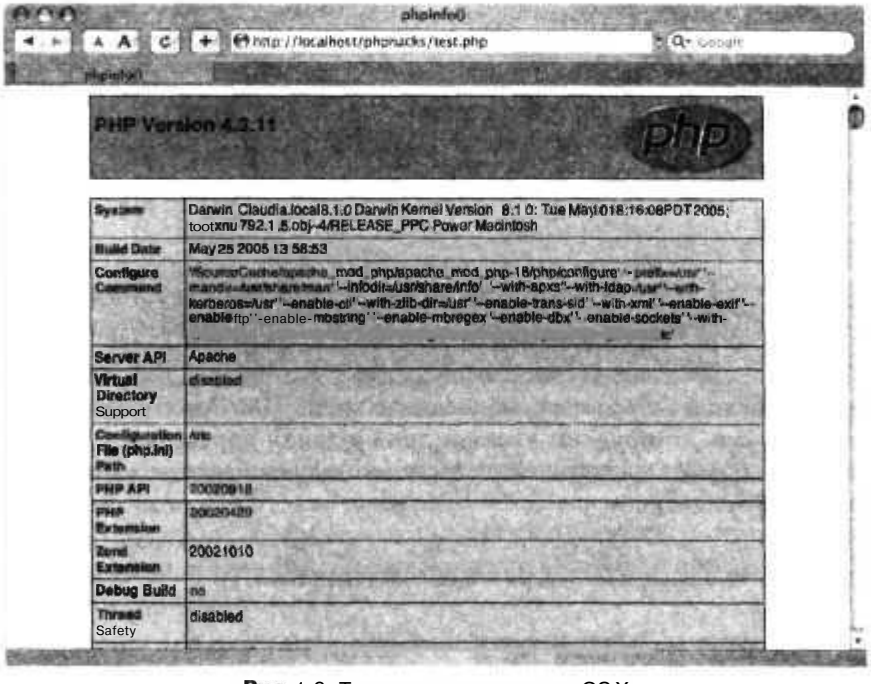


Рис. 1.6. Тестовая страница для OS X

Но на самом деле не все так просто. Для Mac OS X по умолчанию установлен PHP версии 4 с очень ограниченным набором модулей. В частности, отсутствуют *все* модули для работы с графикой! Чтобы установить PHP версии 5, вы можете либо загрузить его исходные тексты и затем скомпилировать, либо скачать уже готовую скомпилированную версию.

ПРИМЕЧАНИЕ

Я рекомендую использовать уже скомпилированную версию, так как это намного проще. Если вы будете компилировать исходные коды PHP, то вам также придется загрузить и скомпилировать множество дополнительных библиотек, используемых PHP. Этот процесс может занять слишком много времени.

На сайте Marc Liyanage есть готовый пакет PHP 5 для OS X с набором отличных предустановленных библиотек (http://www.entropy.ch/software/macosx/php/). Чтобы установить этот пакет, просто скачайте инсталлятор и запустите его.

После установки PHP 5 вам придется переместить исполняемые файлы PHP 4 из места их расположения по умолчанию. Для этого пользуйтесь указанными командами, перемещающими PHP и PEAR в директории php4 и pear4:

```
X sudo mv /usr/bin/php /usr/bin/php4
X sudo mv /usr/bin/pear /usr/bin/pear4
```

Теперь запросите версию интерпретатора PHP, чтобы убедиться, что был установлен PHP версии 5:

```
X php -v
PHP 5.0.4 (cli) (built: Apr 4 2005 17:32:28)
Copyright (c) 1997-2004 The PHP Group
Zend Engine v2.0.4-dev. Copyright (c) 1998-2004 Zend Technologies
```

Для проверки работоспособности веб-сервера Apache вернитесь к запуску тестовой страницы. На рис 1.7 вы можете видеть проверку того, запущен ли PHP 5.

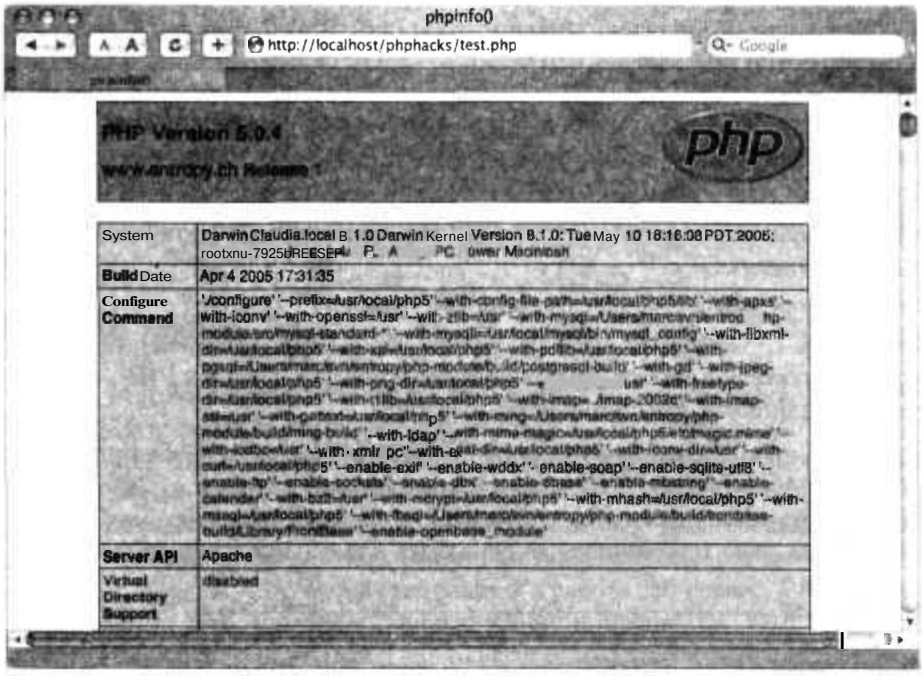


Рис. 1.7. Тестовая страница после установки PHP 5

Установка PHP для Linux

Установка PHP для Linux начинается с проверки наличия уже установленного PHP (в большинстве случаев он уже есть). Сперва вам необходимо проверить наличие веб-сервера Apache. Находятся ли на машине страницы, к которым она предоставляет доступ? Если нет, то проверьте наличие исполняемого файла httpd:

```
my-host$ find / -name httpd
```

Если исполняемый файл найден, то проверьте, запускается ли он автоматически при загрузке. Если веб-сервер Apache не установлен, то первое, что вам необходимо для установки PHP, — веб-сервер. Посетите официальный сайт Apache (http://apache.com), загрузите и установите сервер.

ПРИМЕЧАНИЕ

Я настоятельно рекомендую устанавливать сервер версии 1.3, а не 2.0. Большинство хостингов в Интернете предоставляют Apache 1.3 как стабильно работающий и проверенный временем сервер. Apache 2, будучи разработанным недавно, поддерживает потоки, чего нет в PHP.

После того как был установлен Apache, необходимо проверить наличие установленного PHP. Создайте файл с названием index.php и поместите его в директорию Apache, предназначенную для документов. Содержимое файла должно быть таким:

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

Загрузите в браузере файл index.php. Если у вас на экране отобразилось что-то похожее на рис. 1.7, значит, у вас уже есть готовая рабочая версия PHP. Если отображается только текст файла index.php, то PHP либо не установлен, либо не запущен.

Проверьте ваш файл настроек httpd.conf для Apache. Ищите в нем строки, похожие на следующую:

```
# LoadModule php4_module          libexec/httpd/libphp4.so
```

Если найдете, раскомментируйте эти строки, убрав символ # в начале строки. Если в файле отсутствуют строки, имеющие какое-либо отношение к PHP, то вам придется его установить, используя исходные коды.

Установка с использованием исходных кодов означает, что необходимо скачать с сайта http://www.php.net/ TGZ-файл, в котором они находятся. Далее следуйте инструкциям, размещенным на сайте PHP. Операционная система Linux у вас уже запущена, так что установка должна быть для вас пустяковым делом.

ПРИМЕЧАНИЕ

Я рекомендую устанавливать PHP 5, так как это наиболее современная версия, обладающая необходимыми возможностями для написания мощных приложений.

После того как вы установите PHP, вы сможете перейти к странице index.php, созданной ранее, и увидеть результат его работы (см. рис. 1.7).

Проверка установки ISP

Чтобы проверить особенности установки PHP для ISP, вам необходимо создать тестовую страницу для вашего сервера ISP и попытаться ее загрузить. Содержимое тестовой страницы должно быть следующим:

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

Когда этот файл будет размещен на сервере, у вас должна быть возможность перейти к нему при помощи браузера, и на экране должно отразиться окно, схожее с изображенным на рис. 1.7. С его помощью вы можете получить полную информацию о том, как был скомпилирован интерпретатор PHP, а также полный список установленных модулей.

Основная проблема, с которой приходится сталкиваться, — недостаточный набор возможностей для доступа к базе данных и к средствам для работы с графикой. Вам необходимо проверить доступность этих модулей для вашей учетной записи на ISP. Если эти библиотеки у вас не установлены, то отправьте запрос на их установку (вам не должны сильно препятствовать, это стандартные библиотеки для PHP, полезные для всех программистов PHP).

Если у вас еще нет доступа к ISP, то первым делом проверьте наличие этих модулей, прежде чем попытаетесь зарегистрироваться. Небольшой обзор хостингов, произведенный в процессе написания данной книги, показал, что многие из них поддерживают как PHP 4, так и PHP 5. Однако большинство из них делают это как расширение для CGI, что намного медленнее, чем если бы он был интегрирован прямо в веб-сервер Apache. Если работа с PHP 5 для вас очень важна, то удостоверьтесь, что хостинг поддерживает PHP 5 как отдельный плагин для Apache, а не использует для этого CGI.

Установка MySQL

PHP — это всего лишь одна из частей, входящих в так называемую архитектуру LAMP. LAMP состоит из Linux, Apache, MySQL и PHP (Python, Perl или Ruby). Архитектура LAMP очень популярна из-за легкости в установке и обучении, из-за стабильности в работе и, что является наиболее важным, из-за того, что она бесплатна. Каждая из составляющих LAMP вносит свой важный вклад в ее работу: Linux — операционная система, на базе которой запускаются все компоненты архитектуры, Apache — это очень стабильный в работе веб-сервер, PHP — легкий в использовании язык написания сценариев, ну и, наконец, СУБД MySQL, используемая для хранения данных. Поскольку для многих сложных интернет-приложений требуется хранилище для некоторых структур данных, большинство ISP

под UNIX предоставляют Apache, PHP и MySQL. Это означает, что не только процесс разработки будет очень простым, но и ваше приложение можно будет запустить практически везде.

Установка MySQL крайне проста. Готовые инсталляторы доступны для Windows, Mac OS X и некоторых дистрибутивов Linux. Это самый простой способ быстро запустить MySQL.

В дополнение ко всему исходные коды компилируются очень просто на большинстве платформ UNIX. Чтобы скомпилировать MySQL, используя его исходные коды, сперва загрузите их последнюю версию в TGZ-файле с официального сайта MySQL (http://www.mysql.com/). Распакуйте файлы и следуйте инструкциям, описывающим процесс компиляции и установки. Для этого вам понадобятся права Superuser и доступ к командной строке.

Управление базами данных

Как только СУБД MySQL будет установлена, вам, наверное, захочется создать базу данных, в которой будут храниться таблицы, используемые в вашем веб-приложении. Для создания новой базы данных используйте следующую команду:

```
% mysqladmin --user=root-password=password create dbname
```

Необходимо будет задать поля username и password, а вместо dbname введите имя вашей базы данных.

Большинство трюков в данной книге для своей работы создают базы данных. Этим базам даются различные имена, чтобы избегать наложения их друг на друга. В идеале каждое PHP-приложение должно использовать свою базу данных MySQL.

Удалить базу данных тоже очень просто:

```
% mysqladmin --user=root --password=password drop foo
Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.

Do you really want to drop the 'foo' database [y/N] y
Database "foo" dropped
%
```

В этом примере я удалил базу с именем foo. По умолчанию MySQL выдает запрос, действительно ли вы хотите удалить базу данных. Вы можете избежать этого, используя параметр -f:

```
% mysqladmin --user=root--password=password drop -f foo
Database "foo" dropped
```

ПРИМЕЧАНИЕ

Этот параметр практически незаменим при использовании автоматического обновления баз данных.

Следующим шагом после создания базы данных будет добавление в нее таблиц и данных. Самый простой способ сделать это — воспользоваться клиентским ПО

mysql для работы с файлом схемы данных SQL. Например:

```
mysqladmin -user=root--password=password create btest
mysql --user=root--password=passwordbtest < books.sql
```

Первая команда создает базу данных btest, а вторая переносит ее вместе с таблицами и данными в файл books.sql.

ПРИМЕЧАНИЕ

Есть множество способов переноса структуры и данных, но я считаю, что описанный выше является самым простым.

Если вам не нравится работать с командной строкой, то можете управлять базой данных, используя приложение phpMyAdmin (http://www.phpmyadmin.net). Оно имеет дружественный интерфейс и позволяет вам добавлять или удалять базы, создавать и изменять таблицы, получать доступ к данным и даже добавлять или изменять данные через веб-интерфейс.

Смотрите также

3 «Установка модулей PEAR» (см. трюк 2).

Установка модулей PEAR

Используйте обширные исходные коды PEAR для добавления крутых функций в ваше PHP-приложение.

Библиотека PEAR представляет собой набор добавленных различными пользователями модулей PHP, которые структурированы таким образом, чтобы их можно было просто загрузить, установить и получить доступ к их исходным кодам. Библиотека PEAR является настолько важной, что при установке PHP уже поставляется как одна из ее составных частей.

Чтобы выяснить, что содержится в библиотеке PEAR, посетите сайт http://pear.php.net/. Там вы можете найти полный список модулей и попытаться отыскать какой-либо специфический модуль. Если он будет найден и вам захочется его установить, просто запустите PEAR в командной строке.

В Windows это выглядит так:

```
C:\> pear install DB
downloading DB-1.7.6.tgz ...
Starting to download DB-1.7.6.tgz (124.807 bytes)
.....done: 124,807 bytes
install ok: DB 1.7.6
```

В описанном примере я устанавливаю модуль PEAR, который называется DB (см. трюк 35), — объектно-ориентированную надстройку для баз данных, часто используемую в данной книге.

ПРИМЕЧАНИЕ

Возможно, в Windows вам понадобится удостовериться в наличии файла pear.bat, расположенного в директории bin. Она, в свою очередь, находится в папке, в которую ранее был установлен PHP. Кроме того, директория, куда в дальнейшем будут устанавливаться модули PEAR, по умолчанию не создается. В этом случае вам придется воспользоваться Проводником Windows или командной строкой, чтобы ее создать. Если вы установили PHP в c:\php5, то для PEAR должна быть создана папка c:\php5\pear ответственно. Возможно, вам также понадобится добавить путь к файлу с модулями c:\windows\php.ini file.

В системах на базе UNIX, включая Mac OS X, запустить PEAR так же просто:

```
% sudo pear install HTTP_Client
downloading HTTP_Client-1.0.0.tgz ...
Starting to download HTTP_Client-1.0.0.tgz (6.396 bytes)
.....done: 6.396 bytes
install ok: HTTP_Client 1.0.0
%
```

Здесь я устанавливаю модуль PEAR под названием HTTP_Client (см. трюк 84). Придется воспользоваться командой sudo, так как модуль PEAR необходимо установить в систему.

Чтобы получить полный список модулей PEAR, воспользуйтесь командой list-all:

```
% pear list-all
All packages:
=============
Package                 Latest          Local
APC                     3.0.3
Cache           1.5.4           1.5.4
Cache_Lite      1.4.1
apd                     1.0.1
memcache        1.4
parsekit        1.0
...
```

ПРИМЕЧАНИЕ

Поскольку команда list-all не вносит в систему никаких изменений, права Superuser для ее выполнения не обязательны.

Некоторые модули PEAR будут помечены как unstable. Это означает, что на данный момент они находятся в разработке. При попытке их установить будет выдано сообщение об ошибке:

```
% sudo pear install Services_Amazon
No release with state equal to: 'stable' found for 'Services_Amazon'
```

Здесь мы видим, что модуль Amazon Web Services настолько новый и, возможно, нестабильный, что он помечен как alpha или beta. Вам придется воспользоваться параметром -f, чтобы установить модуль:

```
% sudo pear install -f Services_Amazon
Warning: Services_Amazon is state 'beta' which is less stable than state
       'stable'
downloading Services_Amazon-0.2.0.tgz ...
```

```
Starting to download Services_Amazon-0.2.0.tgz (8.086 bytes)
. . . . .done: 8.086 bytes
install ok: Services_Amazon 0.2.0
```

Второй вариант — указать конкретную версию модуля:

```
# sudo pear install Services_Amazon-0.2.0
downloading Services_Amazon-0.2.0.tgz ...
Starting to download Services_Amazon-0.2.0.tgz (8.086 bytes)
. . . . .done: 8.086 bytes
install ok: Services_Amazon 0.2.0
```

В этом случае проверка на стабильность проводиться не будет. Этот способ применяют, когда более новая версия работает с ошибками и вы хотите вернуться к более старой.

Вы можете вывести полный список уже установленных на вашем компьютере модулей PEAR при помощи команды `list`:

```
% pear list
Installed packages:
=====================
Package Version State
Archive_Tar 1.1 stable
Benchmark 1.2.3 stable
Cache 1.5.4 stable
Console_Getopt 1.2 stable
DB 1.7.6 stable
HTML_Template_IT 1.1 stable
HTTP 1.3.6 stable
HTTP_Client 1.0.0 stable
HTTP_Request 1.2.4 stable
Image_Barcode 1.0.4 stable
Log 1.8.7 stable
Net_Curl 0.2 stable
Net_SmartIRC 1.0.0 stable
Net_Socket 1.0.6 stable
Net_URL 1.0.14 stable
Net_UserAgent_Detect 2.0.1 stable
PEAR 1.3.5 stable
PHPUnit 1.2.3 stable
PHPUnit2 2.2.1 stable
SOAP 0.9.1 beta
Services_Amazon 0.2.0 beta
Services_Google 0.1.1 alpha
Services_Weather 1.3.1 stable
Services_Yahoo 0.1.0 alpha
XML_Parser 1.2.6 stable
XML_RPC 1.2.2 stable
XML_RSS 0.9.2 stable
XML_Serializer 0.16.0 beta
XML_Tree 1.1 stable
XML_Util 1.1.1 stable
```

ВНИМАНИЕ

Не путайте команды list и list-all. Первая перечисляет уже установленные модули, а вторая — все доступные.

Для максимально эффективного использования PHP очень важно научиться грамотно и широко пользоваться возможностями PEAR. Библиотеки, встроенные в PHP, достаточно хороши, но дополнительные модули PEAR делают PHP действительно мощной средой для разработки приложений.

Установка модулей PEAR в ISP. Поскольку у вас нет прав Superuser на машине, на которой установлен ISP, вам придется пошевелить мозгами, чтобы придумать способ, как установить модули PEAR. Во-первых, необходимо определить директорию, куда они будут устанавливаться. Для этого создайте директорию на машине с ISP. Затем воспользуйтесь командой iniset, чтобы добавить эту директорию в качестве пути, по которому хранятся подключаемые модули, как это показано ниже:

```
<?php
ini_sett 'include_path',
im_get( 'include_path' ).PATH_SEPARATOR."/users/jherr/mylibs" );
?>
```

ПРИМЕЧАНИЕ

Этот код необходимо вставить в вашу страницу на PHP или в общий заголовок, который будет подключаться на каждой из страниц.

Таким образом, вы добавляете директорию /users/jherr/mylibs к списку путей, которыми будут пользоваться директивы include и require при поиске. Это необходимо сделать до того, как вы попытаетесь воспользоваться директивами requi re или include для доступа к какому-либо из установленных модулей PEAR.

После того как вы задали директорию для библиотеки и внесли изменения в путь поиска необходимых для подключения модулей, можете загружать модули, которые хотите установить, с сайта http://pear.php.net/. Распаковывайте и помещайте файлы с исходными кодами в директорию для библиотеки, которую вы только что указали (в этом примере — /users/jherr/myl ibs).

ГЛАВА 2

Веб-дизайн

Трюки 3-10

В этой главе представлены трюки для разработки пользовательского интерфейса, который вы можете создать при помощи HTML с использованием PHP. Трюки посвящены созданию элементов вкладок, полей ввода и вывода информации в целях улучшения вашего веб-интерфейса, созданию таких элементов, как ссылочные структуры сайтов, и построению легковесных диаграмм. В эту главу даже включен трюк, показывающий, как отправлять вашим клиентам электронные письма в HTML-формате.

Создание интерфейсов с использованием обложек

Используйте CSS, чтобы пользователь имел возможность выбора внешнего вида вашего веб-приложения.

Вы когда-нибудь встречались с пользователем, которому просто *необходимо,* чтобы каждый блок, который он читает, отображался с использованием его собственной цветовой схемы? Или вы сами являетесь представителем такого рода пользователей? К счастью, благодаря поддержке современными браузерами CSS работать с такими клиентами стало намного проще.

С помощью CSS можно определить используемые шрифты, цвета, размеры и даже положение элементов страницы вне зависимости от HTML-кода. Вы можете кардинально изменить внешний вид HTML-страницы, просто переопределив используемую ей таблицу стилей CSS. В этом трюке показано, как организовать возможность изменения пользователем стиля CSS, а также дается несколько советов по созданию настраиваемых интерфейсов.

Код

Для начала сохраните исходный код примера 2.1 как файл `index.php`.

Пример 2.1. Простой вариант главной страницы с настраиваемыми стилями CSS

```
<html>
<head>
```

```
<?php
$style - "default":
if ( $_GET["style"] )
  $style - $_GET["style"];
$files - array( );
$dh - opendir( "styles" );
while( $file = @readdir( $dh ) )
{
  if( preg_match( '7[.]css$/". $file ) )
  {
    $file = preg_replace( "/[.]css$/", "". $file );
    $files []= $file;
  }
}
?>
<style type="text/css" media="all">@import url(styles/<?php echo($style); ?>.css);</style>
</head>
<body>
<table width="800">
  <tr>
  <td width="200" class="menu" valign="top">
    <div class="menu-active"><a href="home.php">Home</a></div>
    <div class="menu-inactive"><a href="faq.php">FAQ</a></div>
    <div class="menu-inactive"><a href="contact.php">Contact</a></div>
  </td>
  <td width="600" valign="top">

    <table class="box">
      <tr>
      <td class="box-title">
        Important information
      </td>
      </tr>
      <tr>
      <td class="box-content">
        Lots of information about important events and stuff.
      </td>
      </tr>
    </table>

  </td>
  </tr>
</table>
<form>
  Style: <select name="style">
  <?php foreach( $files as $file ) { ?>
  <option value="<?php echo($file); ?>"
  <?php echo( $file – Sstyle ? "selected" : "" ); ?>
  ><?php echo($file); ?></option>
  <?php } ?>
  </select>
  <input type="submit" value="Select" />
</form>
</body>
</html>
```

Далее сохраните пример 2.2 (таблица стилей CSS) в файл `styles/default.css`.

Пример 2.2. Простая таблица стилей **CSS,** использующая красно-белую цветовую схему

```
body { font-family: arial. verdana; font-size: small: margin: 0px; }
.box { background: red; }
.box-title { text-align: center: color: white; font-weight: bold: }
.box-content { background: white: font-size: xx-small; padding:10px;}
.menu { margin: 5px; }
.menu-active { margin: 2px: padding:5px: background: black: }
.menu-active a { text-decoration: none: color: white; font-weight: bold: }
.menu-inactive { margin: 2px; padding:5px; background: #ccc: }
.menu-inactive a { text-decoration: none; }
```

Чтобы обеспечить возможность выбора для пользователя, сохраните пример 2.3 в файл styles/black_and_white.css.

Пример 2.3. Таблица стилей CSS для того же документа HTML, но с черно-белой цветовой схемой

```
body { font-family: arial. verdana: font-size: small; margin: Opx; }
.box { background: #eee: border: 1px solid black: }
.box-title { background: white: text-align: center: font-weight: bold; }
.box-content { background: white; font-size: xx-small: padding:10px;}
.menu { margin: 5px; }
.menu-active { margin: 2px: padding:5px; background: black; }
menu-active a { text-decoration: none: color: white; font-weight: bold; }
.menu-inactive { margin: 2px; padding:5px; background: #ccc; }
.menu-inactive a { text-decoration: none; }
```

Запуск трюка

Загрузите файлы на ваш PHP-сервер и перейдите к странице index.php. Код страницы автоматически выберет установленную по умолчанию обложку, если не была выбрана какая-либо другая. Эта обложка — красно-белая цветовая схема с разными цветами границы и заголовка в информационной таблице — показана на рис. 2.1.

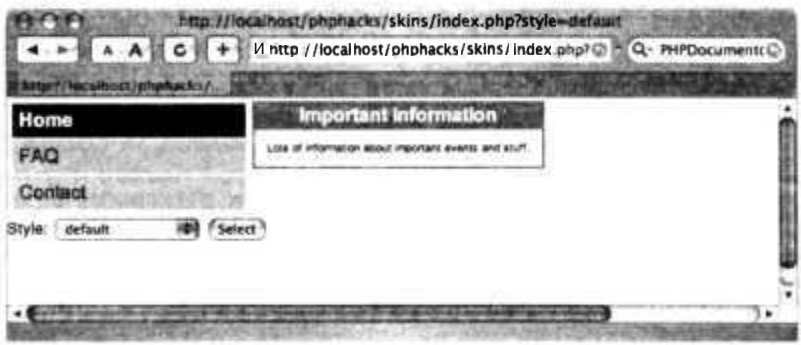


Рис. 2.1. Обложка по умолчанию

Теперь выберите в поле выбора черно-белую обложку и нажмите кнопку Select. Страница должна перезагрузиться со слегка измененной цветовой схемой, как это показано на рис. 2.2.

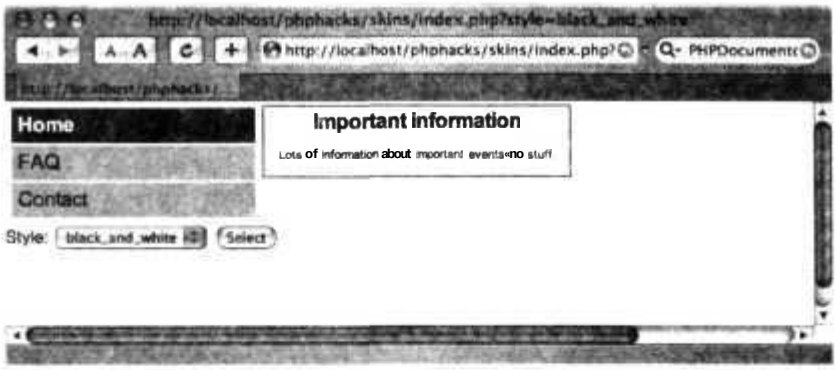


Рис. 2.2. Черно-белая обложка

Этот простой пример не удовлетворит вкус каждого пользователя (или просто скажутся недостаточно богатые возможности в выборе цветов). Тем не менее, добавив дополнительные таблицы стилей или даже позволив пользователям загружать свои собственные, вы можете создать среду, полностью настраиваемую для каждого пользователя.

Однако самым важным является вовсе не исходный код страницы. Он всего лишь позволяет организовать выбор CSS-файла, а затем применить соответствующий стиль (директива `@import` в теге `<style>`). Самое приятное, когда при помощи CSS происходят изменения цветов, шрифтов и даже разметки страницы.

Вот несколько подсказок для удачного применения в веб-дизайне интерфейсов с изменяемыми обложками.

- **Используйте разметку CSS.** CSS позволяет вам указывать место для размещения элементов DIV, используемых для разметки на странице, при помощи абсолютного или относительного положения или при помощи элементов с плавающим расположением.
- **Управляйте шрифтами при помощи CSS.** Используйте только CSS для управления шрифтами, их размерами и стилями на странице. Избегайте использования тега <font>, а вместо него работайте с такими тегами, как <p>, `<div>` и , используя атрибут `class`, определяющий, какой CSS-стиль будет применен.
- **Разрабатывайте документацию к вашим стилям CSS.** Используйте хорошо документированную обложку, применяемую по умолчанию, чтобы ее можно было применять в качестве шаблона для создания других обложек. CSS поддерживает комментарии, и вам следует использовать их для того, чтобы определять, какие классы применяются к тем или другим элементам страницы.
- **Используйте ID в ваших элементах разметки.** Используйте атрибут id в тегах , чтобы разбивать вашу разметку на секции. Разработчики обложек в дальнейшем могут пользоваться этим, чтобы сосредоточиться на изменениях в определенной области экрана. Например, вы можете решить, что якорные

теги в секции навигации должны использовать атрибуты id, в отличие от всех остальных тегов страницы.

- **Учитесь у других.** Программное обеспечение для блоков, в частности Six Apart's Movable Type **(http://sixapa.rt.com/movabletype),** было изначально разработано для создания обложек. Установите его, чтобы понять, как разработчики Six Apart создают свои шаблоны CSS, которые легко могут использовать даже не искушенные в дизайне люди для изменения внешнего вида своих блоков.

Если вы серьезно заинтересовались созданием обложек, то вам также следовало бы организовать обмен обложками у вас на сайте, выделив его на странице объявлений либо в разделе обмена файлами. Это подтолкнет людей к дальнейшей разработке дизайна и экспериментированию над новыми свойствами обложек. Отправной точкой для этого может стать медиа-центр загрузок и выгрузок (см. трюк 97).

Смотрите также

- «Возможность использования вашими клиентами контроля над форматированием при помощи XSL» (см. трюк 7).
- ❑ «Создание медиацентра загрузок и выгрузок» (см. трюк 97).

Создание элементов внутренней ссылочной структуры сайта

Используйте элементы внутренней ссылочной структуры сайта, чтобы пользователи ориентировались, в каком разделе они сейчас находятся.

Элементы внутренней ссылочной структуры сайта — это список ссылок в верхней части страницы, показывающий текущее расположение в иерархии страницы.

ПРИМЕЧАНИЕ

Элементы внутренней ссылочной структуры сайта — это не набор страниц, которые уже посетил пользователь, как можно подумать исходя из термина. У пользователя уже есть кнопка Back, чтобы вернуться назад.

Элементы внутренней ссылочной структуры сайта позволяют пользователям переходить немного выше по иерархии сайта для поиска более важной информации. Для примера ссылочная структура может выглядеть так: Home /Platforms / Portables / PSP. Пользователь может легко вернуться назад на страницу **Portables,** отображающую список всех мобильных игровых консолей, на страницу **Platforms,** отображающую различные игровые консоли, или на домашнюю страницу. Все это можно сделать с помощью всего лишь одного щелчка кнопкой мыши.

На рис. 2.3 показана используемая Yahoo! структура для отображения текущего местоположения. В данном конкретном случае я нахожусь в подразделе **Buddy,** в категории Comedy, расположенном в области Titles раздела **Movies and Films.** Как пользователь, я могу вернуться назад к любому интересующему меня уровню.

Рис. 2.3. Элементы внутренней ссылочной структуры Yahoo!

Принято, что последний элемент в списке — текущая страница, и он никак не отображается. Первым элементом является главная страница сайта.

Код

Чтобы добавить ссылочную структуру к вашему сайту, сохраните код примера 2.4 в файл showpage.php.

Пример 2.4. Создание внутренней ссылочной структуры сайта

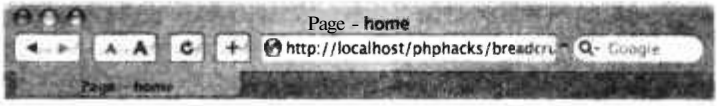
```
<?php
$id - $_GET['id'];
if ( strlen( Sid ) < 1 )
  Sid = "home";

Spages = array(
  home => array( id=>"home", parent=>"", title=>"Home",
```

```
          url=>"showpage.php?id=home" ),
  users => array( id=>"users", parent=>"home", title=>"Users",
           url=>"showpage.php?id=users" ),
  jack => array( id=>"jack", parent=>"users", title=>"Jack",
          url=>"showpage.php?id=jack" )
);

function breadcrumbs( $id, $pages )
{
  $bcl = array( );
  $pageid = $id;
  while( strlen( $pageid ) > 0 )
  {
    $bcl[] = $pageid;
    $pageid = $pages[ $pageid ]['parent'];
  }
  for( $i = count( $bcl ) - 1; $i >= 0; $i-- )
  {
    $page = $pages[$bcl[$i]];
    if ( $i > 0 )
    {
      echo( "<a href=\"" );
      echo( $page['url'] );
      echo( "\">" );
    }
    echo( $page['title'] );
    if ( $i > 0 )
    {
      echo( "</a> | " );
    }
  }
}
?>
<html>
<head>
<title>Page - <?php echo( $id ); ?></title>
</head>
<body>
Breadcrumbs: <?php breadcrumbs( $id, $pages ); ?><br/>
Page name: <?php echo( $id ); ?>
</body>
</html>
```

Исходный текст на самом деле прост. Он начинается с объявления списка страниц. Этот список имеет вид хэш-таблицы, в которой в качестве ключевого поля используется ID страницы. Далее функция управления ссылочной структурой, используя ID текущей страницы и полный список страниц и перемещаясь по списку назад от текущей страницы, создает ссылочную структуру. После этого HTML-код отображает текущую страницу и созданную для нее ссылочную структуру.

Запуск трюка

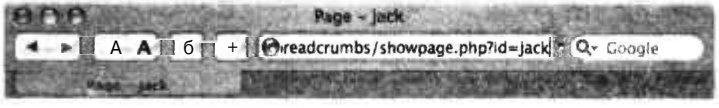
Загрузите этот исходный код на ваш сервер и перейдите к странице showpage.php. По умолчанию будет загружена домашняя страница (рис. 2.4V



Breadcrumbs: Home
Page name: home

Рис. 2.4. Домашняя страница

Пока ничего впечатляющего не видно, так как на домашней странице ссылочная структура пуста. Запросите другую страницу, добавив `?id=jack` к URL. В результате получится окно, похожее на изображенное на рис. 2.5.



Breadcrumbs: Home | Users | Jack
Page name: jack

Рис. 2.5. Страница со ссылочной структурой

Как вы можете видеть из рис. 2.5, имеется внутренняя ссылочная структура сайта, содержащая домашнюю страницу и страницу пользователя со статическим текстом на ней.

Улучшаем трюк

Было бы намного проще отслеживать список страниц, если бы он был представлен в XML-формате. Вот пример того, как мог бы выглядеть такого рода XML-файл:

```
<pages>
  <page id="home" parent="" title="Home" url="showpage.php?id=home" />
  <page id="users" parent="home" title="Users" url="showpage.php?id=users"/>
  <page id="jack" parent="users" title="Dack" url="showpage.php?id=jack" />
</pages>
```

Чтобы его обработать, вы можете использовать встроенные в PHP функции или некоторые стандартные выражения (см. трюк 38).

Смотрите также

LJ «Создание всплывающих подсказок» (см. трюк 12).

- «Создание динамических меню навигации» (см. трюк 17).

ТРЮК №5

Создание рамок при помощи HTML

Используйте HTML и основные функции работы с графикой для создания привлекательных рамок для ваших веб-страниц.

Иногда бывает намного полезнее разместить содержимое вашего сайта в рамках, чтобы пользователям было удобнее в нем ориентироваться. Таким образом вы можете привлечь внимание к какой-то особенной части содержимого, создать интерфейс наподобие газеты или просто выразить ваши претензии на художественность в стиле кубизма. Сценарии в этом трюке позволяют с легкостью поместить любое содержимое в рамку.

Код

Сохраните код из примера 2.5 в файл boxltest.php.

Пример 2.5. Пробная страница с содержанием в рамках

```
<html>
<head>
<? include( "boxi.php" ):
add_box_styles();
?>
</head>
<body>
<div style="width:200px;">
<? start_box( "News" ): ?>
Today's news is that there is no news. Which is probably a good thing since
the news can be fairly distressing at times.<br/><br/>
<a href="morenews.html">more...</a>
<? end_box( ); ?>
</div>
</body>
</html>
```

Для создания мини-приложения на PHP сохраните исходный код из примера 2.6 в файл box1.php.

Пример 2.6. Добавление PHP и CSS

```
<?
function add_box_styles( ) { ?>
<style type="text/css">
.box
{
  font-family: arial, verdana, sans-serif;
  font-size: x-small;
  background: #ccc;
}
.box-title
{
  font-size: small;
  font-weight: bold;
  color: white;
```

```
  background: #777;
  padding: 5px;
  text-align: center;
}
 box-content
{
  background: white;
  padding: 5px;
}
</style>
<? }

function start_box( $name ) { ?>
<table class="box" cellspacing="2" cellpadding="0">
<tr><td class="box-title"><? print( $name ) ?></td></tr>
<tr><td class="box-content">
<? }

function end_box( ) { ?>
</td></tr></table>
<? } ?>
```

Самое важное, что можно не просто использовать теги CSS, а настроить рамку по вашему желанию, выделив любую информацию из нее. Вы можете даже скомбинировать это с интерфейсом выбора стиля CSS (см. трюк 3) и оставить за пользователем выбор стиля рамок!

Запуск трюка

Запустите в вашем браузере сценарий box1test.php. Вы увидите окно, похожее на изображенное на рис. 2.6.

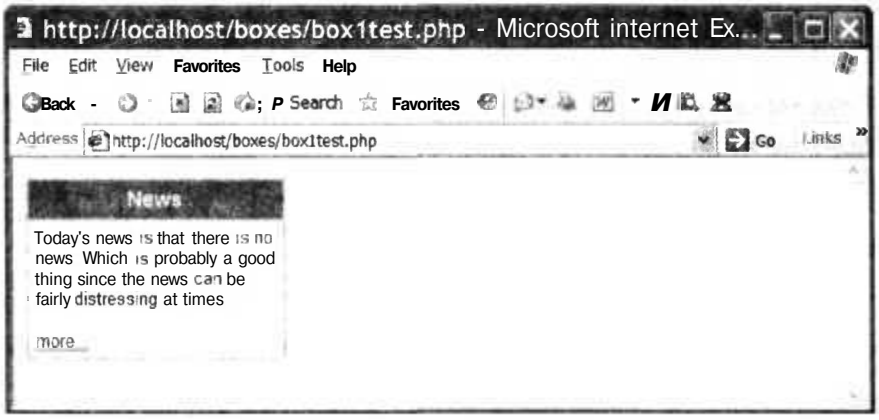


Рис. 2.6. Окончательный вид HTML-рамки

Реклама новостей размещена в черно-белой рамке, потому что этот код находится в скобках между вызовами start_box и endbox. Название рамки является входным параметром для функции start_box.

ПРИМЕЧАНИЕ

Если вам необходимо использовать рамки нескольких различных цветов, то можете создать для этого несколько CSS-классов. Вы также можете разместить атрибуты стиля в тегах <table> и <td>, чтобы переопределить цвет, используемый для рамки по умолчанию.

Улучшаем трюк

Если вам необходимо что-то более привлекательное, то можете создать круглые рамки, используя для этого файлы формата GIF или PNG. Для начала сохраните код из примера 2.7 в файле box2test.php.

Пример 2.7. Пример рамки с закругленными прямоугольниками

```
<html>
<head>
<? include( "box2.php" );
add_box_styles( );
?>
</head>
<body>
<div style="width:200px;">
<? start_box( "Mews" ); ?>
Today's news is that there is no news. Which is probably a good thing since
the news can be fairly distressing at times. <br/><br/>
<a href="morenews.html">more...</a>
<? end_box( ); ?>
</div>
</body>
</html>
```

Затем сохраните код из примера 2.8 в файл box2.php.

Пример 2.8. Использование изображения вместе с **CSS** для создания рамок различного вида

```
<?
function add_box_styles( ) { ?>
<style type="text/css">
.box
{
  font-family: arial, verdana, sans-serif;
}
.box-title
{
  font-size: small;
  font-weight: bold;
  color: white;
  background: #000063;
  text-align: center;
}
.box-content-container
{
  background: #000063;
}
.box-content
{
```

```
  background: white;
  font-size: x-small;
  padding: 5px:
}
</style>
<? }

function start_box( $name ) { ?>
<table cellspacing="0" cellpadding="0" class="box">
  <tr><td>

  <table cellspacing="0" cellpadding="0" width="100%" class="box-title">
  <tr><td width="20" height="20"><img src="blue_ul.png" /></td>
  <td><? print( $name ) ?></td>
  <td width="20" height="20"><img src="blue_ur.png"></td></tr></table>

  </td></tr>
  <tr><td>

<table width="100%" cellspacing="2" cellpadding="0"
<tr><td class="box-content">
<? }

function end_box( ) { ?>
</td></tr></table>
<tr><td>

<table cellspacing="0" cellpadding="0" width="100%" class="box-title">
<tr><td width="20" height="20"><img src="blue_ll.png" /></td>
<td> </td>
<td width="20" height="20"><img src="blue_lr.png"></td></tr></table>

</td></tr></table>
<? } ?>
```

После того как вы разместите новую версию исходного текста на сервере, при запуске сценария box2test.php браузер будет выглядеть так, как показано на рис. 2.7.

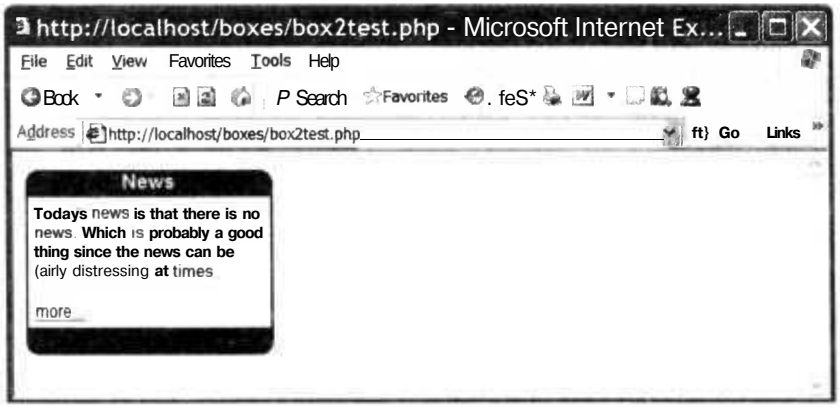


Рис. 2.7. Та же самая рамка, но с закругленными границами и с использованием PNG-рисунков

Теперь изображения из PNG-файлов расположены по углам рамки. Сама рамка, как и раньше, создается при помощи функций start_box и endbox. Однако HTML-код для создания рамки стал более сложным: внутри главного тега <table> находятся еще три. В первом создается заголовок, во втором хранится содержимое, а в третьем — нижняя граница.

Фоновые цвета границ должны точно соответствовать цветам рисунков. Если вы хотите использовать другие цвета, то необходимо создать другой CSS-стиль и другой набор файлов с картинками.

ПРИМЕЧАНИЕ

Файлы картинок, как и исходные тексты, можно загрузить с сайта книг O'Reilly (http://www.oreilly.com/catalog/phphks). Картинки были созданы с помощью Macromedia Fireworks. Но для этих целей также вполне подойдет Adobe Photoshop.

Смотрите также

- «Добавление вкладок к веб-интерфейсу» (см. трюк 6).

ТРЮК №6

Добавление вкладок к веб-интерфейсу

Используйте HTML и CSS для создания в веб-приложении интерфейсов с вкладками.

Иногда на странице необходимо разместить большое количество информации. Самый простой способ для разбиения на составные части сайта (а может, и просто большой по размерам страницы) — использование вкладок, при помощи которых информация разделяется на несколько блоков, каждый из которых соответствует определенной вкладке. К счастью для нас, организация такого разбиения при использовании PHP становится очень легкой задачей.

Код

Сохраните текст программы из примера 2.9 как index.php.

Пример 2.9. Использование библиотеки создания вкладок для демонстрации работы использующего их интерфейса

```
<?php
require_once("tabs.php");
?>
<html>
<head>
<?php tabs_header( ); ?>
</head>
<body>
<div style="width:600px;">
```

```
<?php tabs_start( ); ?>
<?php tab( "Tab one" ): ?>
This is the first tab.
<?php tab( "Tab two" ); ?>
This is the second tab.
<?php tabs_end( ); ?>
</div>
</body>
</html>
```

Приведенный далее исходный код — неплохая библиотека с использованием PHP и CSS. Сохраните текст примера 2.10 в файл tabs.php.

Пример 2.10. Использование PHP и CSS для создания дружественного для пользователя интерфейса с применением вкладок

```
<?php
Stabs = array( );

function tabs_header( )
{
?>
<style type="text/css">
.tab
{
  border-bottom: lpx solid black;
  text-align: center:
  font-family: arial, verdana:
}
.tab-active
{
  border-left: lpx solid black:
  border-top: lpx solid black:
  border-right: lpx solid black:
  text-align: center:
  font-family: arial. verdana:
  font-weight: bold:
}
.tab-content
{
  padding: 5px;
  border-left: lpx solid black:
  border-right: lpx solid black:
  border-bottom: lpx solid black:
}
</style>
<?php
}

function tabs_start( )
{
  ob_start( ):
}

function endtab( )
{
  global Stabs;

  Stext .= ob_get_clean( );
```

```
  $tabs[ count( Stabs ) - 1 ][ 'text' ] = $text;

  ob_start( );
}

function tab( Stitle )
{
  global Stabs;

  if ( countC Stabs ) > 0 )
    endtab( );

  Stabs [] - array(
  title -> Stitle.
  text => ""
  );
}

function tabs_end( )
{
  global Stabs;

  endtab( );
  ob_end_clean( );

  Sindex = 0;
  if ( $_GET['tabindex'] )
    Sindex = S_GET['tabindex'];

  ?>
  <table width="100%" cellspacing="0" cellpadding="0">
  <tr>
  <?php
  Sbaseuri = $_SERVER['REQUEST_URI'];
  Sbaseuri = preg_replace( "/\?.*$/". "". Sbaseuri );
  Scurindex = 0;
  foreach( Stabs as Stab )
  {
    Sclass - "tab";
    if ( Sindex — Scurindex )
      Sclass ="tab-active";
    ?>
    <td class="<?php echo(Sclass); ?>">
    <a href="<?php echo( $baseuri."?tabindex=".$curindex ); ?>">
    <?php echo( $tab['title'] ); ?>
    </a>
    </td>
    <?php
    Scurindex += 1;
  }
  ?>
  </tr>
  <tr><td class="tab-content" colspan="<?php echo( count( Stabs ) + 1 ); ?>">
  <?php echo( Stabs[Sindex ]['text'] ); ?>
  </td></tr>
  </table>
  <?php
}
?>
```

Я разработал API для более удобного использования вкладок. Для начала необходимо добавить tabs_header в секции заголовков вашего документа. Это позволит использовать CSS при работе с вкладками. Далее, в самом документе, чтобы подключить вкладки, необходимо вызвать функцию `tabs_start`. Создание каждой новой вкладки начинается с вызова функции tab, в которой нужно указать ее название. Создание вкладки завершается вызовом функции tabsend.

Система по работе с вкладками выглядит так: внутри хранится выходной буфер с содержимым каждой из вкладок, а отображается та из них, которая на данный момент выбрана.

Запуск трюка

Загрузите файлы на ваш PHP-сервер и перейдите в браузере к странице `index.php`.

Вы увидите окно, похожее на изображенное на рис. 2.8.

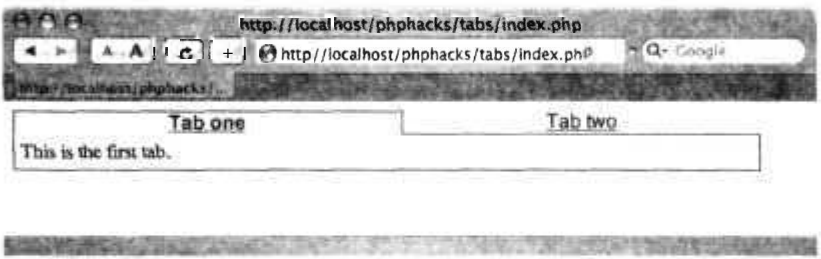


Рис. 2.8. Первая вкладка

Перейдите на вторую вкладку (помеченную как **Tab** two), и вы увидите, что отобразится ее содержимое (рис. 2.9).

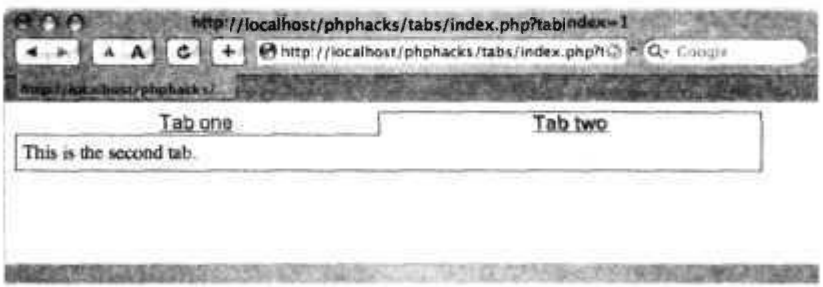


Рис. 2.9. Вторая вкладка

ПРИМЕЧАНИЕ

Каждый раз, когда пользователь переключается на другую вкладку, страница перезагружается. Используя CSS и Dynamic HTML (DHTML) для применения тега к содержимому каждой из вкладок и контролируя вручную видимость каждой из них, вы можете добиться того, чтобы при переключении вкладок страница не перезагружалась.

Смотрите также

- «Создание элементов внутренней ссылочной структуры сайта» (см. трюк 4).
- «Создание динамических меню навигации» (см. трюк 17).

ТРЮК №7

Возможность использования вашими клиентами контроля над форматированием при помощи XSL

Используйте поддерживаемый PHP XSL, чтобы позволить вашим пользователям самим создавать страницы.

Amazon предоставляет интересную услугу для своих корпоративных клиентов. Пользователи могут применять на страницах Amazon обложки, используя таблицы стилей XSL, которые изменяют внешний вид информации о товарах, ценах и сопутствующих сведений. Это означает, что если вы корпоративный клиент, то можете добавлять свои собственные ссылки и графику и даже изменять внешний вид страницы Amazon.com, чтобы обеспечить наилучшее восприятие страниц для торговли. Следующий трюк позволяет делать то же самое с применением XSL для PHP (и не требует статуса корпоративного пользователя!). На рис. 2.10 показан принцип работы XSL в этом трюке (и XSL в целом). На вход подаются два файла. В данном случае это `input.xml`, содержащий информацию, размещенную на странице, и `format.xsl`, содержащий форматирование для страницы, а также месторасположение информации на странице. Обработчик XSL, используя эти два файла, создает XML, HTML либо текст.

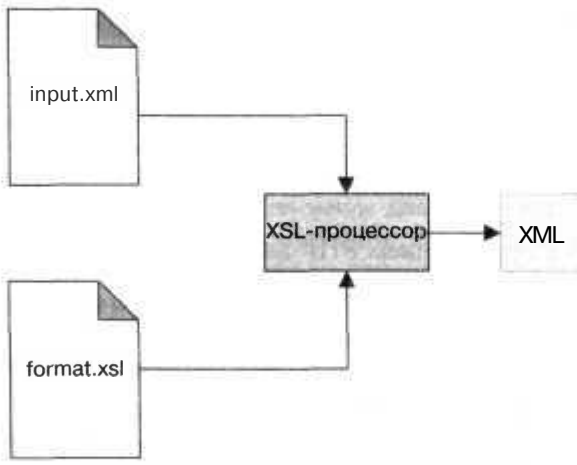


Рис. 2.10. Принцип работы обработчика XSL

Код

Сохраните исходный код из примера 2.11 в файл conv.php.

Пример 2.11. Создание выходного файла из XML и XSL

```
<?php
$xml = new DOMDocument( );
$xml->Load( "input.xml" );

$xsl » new DOMDocument( );
$xsl->Load( "format.xsl" );

$xslproc = new XSLTProcessor( );
$xslproc->importStylesheet( $xsl );
print( $xslproc->transformToXML( $xml ) );
?>
```

В качестве простого XML-файла сохраните текст из примера 2.12 в файл input.xml.

Пример 2.12. Простой XML-файл

```
<books>
  <book name="Code Generation in Action" />
  <book name="MDA Explained" />
  <book name="PHP in a Nutshell" />
</books>
```

Сохраните текст из примера 2.13 в файл с именем format.xsl.

Пример 2.13. Пример XSL-форматирования

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" />
  <xsl:template match="/">
  <html>
    <body>
      <xsl:for-each select="/books/book">
      <xsl:value-of select="@name" /><br/>
      </xsl:for-each>
    </body>
  </html>
  </xsl:template>
</xsl:stylesheet>
```

Запуск трюка

Трюк запускается с применением PHP-интерпретатора для командной строки. Запустите файл conv.php:

```
% php conv.php
```

Результат вы можете видеть на рис. 2.11.

Здесь выводятся данные, которые находятся в файле books.xml. Как видите, они переформатированы в HTML-формат. Вы можете запросто передать эти выходные данные в HTML-файл или создать сценарий, который сформирует HTML-файл после XML-запроса. Как бы вы ни поступили, будет очень просто получить

HTML из XML и XLS в качестве результата на запрос (или какие-либо другие структурированные данные, например, WML). Вы можете организовать доступ к вашим собственным данным динамически, используя в качестве источника SQL и создав объект DOMDocument XML, в котором будут храниться ваши данные, на ходу, что обеспечивает даже большую гибкость. Возможно, вам также захочется, чтобы XML и XSL получали информацию из командной строки, а не через PHP. Таким образом, conv.php — крайне полезная утилита для динамических сайтов.

Рис. 2.11. Запуск интерпретатора из командной строки

Вы можете воспользоваться этим кодом на ваших собственных страницах и разрешить пользователям загружать свои собственные таблицы стилей XSL. Трюк «Создание медиа-центра загрузок и выгрузок» (см. трюк 97) содержит исходный код, позволяющий загружать файлы и сохранять их в локальной директории. XSLT — очень мощный и немного загадочный язык для управления XML.

Если вы решили взяться за него всерьез, то рекомендую отличную книгу Майкла Кея «XSLT 2.0 Programmer's Reference» (издательство Wiley).

Смотрите также

- «Создание интерфейсов с применением обложек» (см. трюк 3).

ТРЮК №8

Создание графиков на HTML

Используйте HTML при создании простых графиков для ваших данных.

Последнее время складывается впечатление, что каждый сайт, который вы посещаете, для отображения ярких картинок и графиков требует от вас установленного плагина QuickTime или Flash. Тем не менее для простейших гистограмм вам совершенно не нужен необычный отрисовщик картинок или флэш-роликов. Вы можете воспользоваться данным трюком для создания гистограмм, используя несколько HTML-таблиц и немного кода на PHP. Результат выглядит так же круто, как и на других перегруженных флэшем сайтах, но не требует каких-либо дополнительных плагинов для установки.

Код

Сохраните текст из примера 2.14 в файл с именем htmlgraph.php.

Пример 2.14. Создание простых гистограмм

```
<html>
<?
$data - array(
        array( "movies". 20 ).
        array( "food". 30 ).
        array( "workout". 10 ).
        array( "work". 40 )
);
$max = 0;
foreach ( $data as $d ) { $max +- $d[l]; }
?>
<body>
<table width="400" cellspacing="0" cellpadding="2">
<? foreach( $data as $d ) {
$percent - ( $d[l] / $max ) * 100;
?>
<tr>
<td width="20%"><? echo( $d[0] ) ?></td>
<td width="10%"><? echo( $d[l] ) ?>%</td>
<td>
<table width="<? echo($percent) ?>%" bgcolor="#aaa">
<tr><td> </td></tr>
</table>
</td>
</tr>
<? ) ?>
</table>
</body>
</html>
```

Вы можете выбирать несколько вариантов для создания фафиков при помощи HTML. Я предпочитаю использовать две таблицы: в первой содержится текстовая информация, а во второй — несколько идущих друг за другом таблиц, ширина каждой из которых соответствует ширине значения в этом ряду.

Я рассчитываю ширину, находя сперва максимальное значение из предоставленных данных, и сохраняю ее в переменную `$max`. Затем я рассчитываю процент, разделив на `$max` текущее значение и умножив результат на 100 (чтобы получить шкалу от 0 до 100). Это число я сохраняю в переменную $percent, которая затем используется как ширина для таблицы.

Запуск трюка

Перейдите в браузере к странице `htmlgraph.php` (рис. 2.12).

ПРИМЕЧАНИЕ

Я знаю, что у изображения на рис. 2.12 нет никаких шансов получить какие-либо дизайнерские награды, но, тем не менее, оно выглядит вполне достойно. К тому же уже не так сложно использовать некоторые интересные возможности CSS и превратить обычный график в нечто необычное.

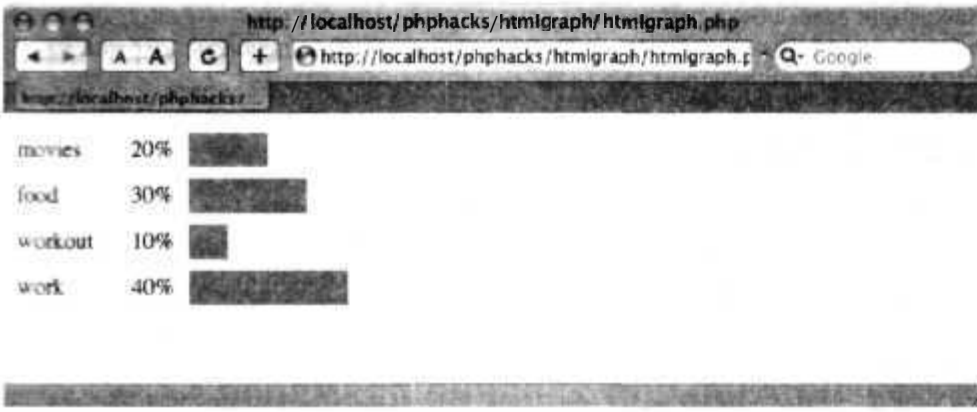


Рис. 2.12. Простая гистограмма в браузере Safari

Улучшаем трюк

Я признаю, что серый — не самый подходящий для графика цвет, поэтому в примере 2.15 мы слегка изменим сценарий, добавив немного цвета.

Пример 2.15. Добавляем цвет

```
<html>
<?
$data - array(
        array( "movies". 20. "red" ).
        array( "food". 30. "green" ).
        array( "workout". 10. "blue" ).
        array( "work". 40. "black" )
);
$max = 0;
foreach ( $data as $d ) { $max += $d[1]: }
?>
<body>
<table width="400" cellspacing="0" cellpadding="2">
<? foreach( $data as $d ) {
$percent - ( $d[1] / $max ) * 100:
?>
<tr>
<td width="20r><? echo( $d[0] ) ?></td>
<td width-"10r><? echo( $d[l] ) ?>%</td>
<td>
<table width="<? echo($percent) ?>%" bgcolor="<? echo($d[2]) ?>">
  <tr><td> </td></tr>
</table>
</td>
</tr>
<? ) ?>
</table>
</body>
</html>
```

Введенная нами новая переменная используется для изменения фоновых цветов таблицы с гистограммами. Это позволяет гистограмме выглядеть похожей на радугу, что намного приятнее для глаза, чем просто серый цвет.

Смотрите также

- «Создание динамических графиков на HTML» (см. трюк 14).
- «Добавление векторной графики при помощи PHP» (см. трюк 22).
- «Создание красивых рисунков при помощи SVG» (см. трюк 28).

Правильное задание размеров изображений

Используйте поддержку изображений в PHP, чтобы правильно задавать ширину и высоту ваших рисунков.

Все современные браузеры отображают страницу настолько быстро, настолько это возможно, и поэтому пользователю кажется, что задержка при загрузке страниц уменьшается. Это значит, что браузер начнет отображать страницу задолго до того, как загрузит все соответствующие изображения и ресурсы. Поскольку браузер еще не успел загрузить изображение, он не знает, какого размера оно будет, кроме того случая, когда вы явно указали ширину и высоту в теге <img>. Если вы не укажете эти атрибуты, то страница резко отобразит рисунок, как только он будет загружен. Браузер попытается угадать размер изображения (обычно 10 × 10 пикселов), но затем, когда оно будет загружено, определит, что размеры изображения другие. Таким образом, браузеру придется заново оформить страницу, чтобы изменить размеры изображений. Данный трюк позволяет задавать тег <img> с уже правильно указанными размерами, используя функцию getimagesize для их получения.

Исходный код

Сохраните текст из примера 2.16 в файл imagesize.php.

Пример 2.16. Некоторые хитрости при работе с изображениями

```
<html>
<?php
function placegraphic( $file )
{
  list( $width, $height ) - getimagesize("rss.png");
  echo( "<img src=\"$file\" width=\"$width\" height=\"$height\" />" );
}
?>
<body>
<?php placegraphic( "rss.png" ); ?>
</body>
</html>
```

Запуск трюка

Создайте файл rss.png с изображением в той же директории, в которой расположен PHP-файл, и отобразите в браузере страницу imagesize.php. Вы увидите, что картинка отобразится правильно, без изменений в размерах. Используйте в браузере команду **View Source** для просмотра исходного HTML-кода, чтобы убедиться, что высота и ширина изображения установлены правильно. Вы можете пользоваться функцией placegraphic везде, где вы раньше использовали тег <img>, но не рекомендуется использовать эту функцию в статических заголовках или колонтитулах, где размер изображения никогда не будет изменяться. Выигрыш в производительности будет за счет более быстрой отрисовки изображения с заданными размерами. Вам следует пользоваться этой функцией, если размер картинки не известен до тех пор, пока не было запроса на ее отображение. В дополнение ко всему, если вы используете базу данных для хранения библиотеки с изображениями, я рекомендую хранить в ней также ширину и высоту картинки наряду с ее месторасположением. Получение размеров изображения и пути к нему из базы данных требует намного меньше времени, чем использование функции getimagesize в тот самый момент, когда вы собираетесь отобразить рисунок.

Улучшаем трюк

Помимо задания размеров изображения в соответствующем теге, вам приходится устанавливать атрибут alt. Он служит для текстового описания изображения в случаях, когда у пользователя нет возможности отображать картинки или он пользуется текстовым браузером.

Для поддержки атрибута alt просто добавьте еще один аргумент в функцию placegraphic, а затем выводите значение аргумента при помощи echo.

Смотрите также

- «Создание предпросмотра изображений» (см. трюк 27).
- «Использование объектов для упрощения работы с графикой» (см. трюк 29).
- «Разбиение изображения на части» (см. трюк 30).

ТРЮК №10

Отправка электронной почты с помощью HTML

Используйте электронные письма из нескольких составных частей для отправки содержимого в текстовом и HTML-формате.

Электронная почта — это еще одна из составляющих частей интерфейса вашего веб-приложения. В идеале вам бы хотелось, чтобы эта часть интерфейса (как и другие, будь это телефон или другое переносное устройство) была полностью

доступна на вашем веб-сервере. Но, поскольку это пока не всегда возможно, было бы неплохо запомнить такие возможности, чтобы подтолкнуть себя к созданию более продвинутых пользовательских интерфейсов в дальнейшем. В трюке рассказано о том, как отправлять электронную почту, состоящую из нескольких частей, когда одна из них — текстовая версия письма, а другая — версия с использованием HTML. Если у пользователя нет возможности просматривать почту в HTML-формате, они, тем не менее, получат письмо без HTML-разметки.

На рис. 2.13 показано несколько различных форм сообщения по электронной почте. Слева вы можете увидеть простейшую форму почтового сообщения — текстовую. Сперва в письме идет заголовок, в котором определена тема, адресат, отправитель и т. д. После этого следует перевод каретки и само сообщение.

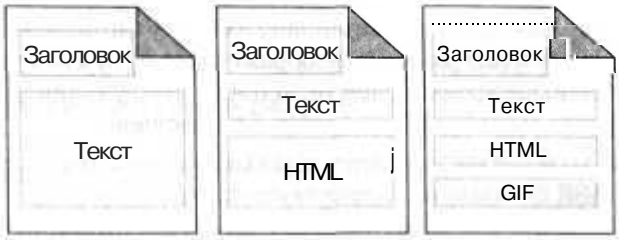


Рис. 2.13. Различные формы сообщения

Сообщения в центре и справа на рис. 2.13 являются составными. Заголовок в целом остается таким же, как и у простого письма, за исключением того, что добавлена некоторая информация о других частях сообщения. Далее текст помещается в одну часть письма, а HTML — в другую.

Благодаря этому получатель может решать, какую часть письма он хочет просмотреть. В электронном письме с HTML, содержащем графику, могут также находиться ссылки в виде изображений, как показано справа на рис. 2.13.

Код

Сохраните текст из примера 2.17 в файл html email .php.

Пример 2.17. Отправление составных сообщений с HTML и простых текстовых сообщений по электронной почте

```
<?php
$to = "to@email.com";
$to_full = "Sally Cool";
$from = "from@email.com";
$from_full = "Joe Schmoe";
$subject - "HTML Mail Test";

$random_hash = "zzz582x";

ob_start( );
```

```
?>
To: <?php echo($to_full); ?> «?php echo($to): ?»
From: <?php echo($from_full); ?> «?php echo($from); ?»
MIME-Version: 1.0
Content-Type: multipart/alternative;
boundary="==Multipart_Boundary_<?php echo($random_hash);
?>"
<?php
$headers - ob_get_clean( ):
ob_start( );
?>

This is a multi-part message in MIME format.

--==Multipart_Boundary_<?php echo( $random_hash ): ?>
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

This is the text of the message in a simple text format.

--==Multipart_Boundary_<?php echo( $random_hash ): ?>
Content-Type: text/html: charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

<html>
<body>
<p>Here is something with <b>HTML</b> formatting. That can include all of the usual:</p>
<ul>
<li>Bulleted lists</li>
<li>Tables</li>
<li>Images (if you include them as attachments or external links)</li>
<li>Character formatting</li>
<li>...and more!</li>
</ul>
</body>
</html>

--==Multipart_Boundary_<?php echo( $random_hash ); ?>--

<?php

$message = ob_get_clean( ):

$ok = @mail( $to, $subject. $message, $headers ):

echo( $ok ? "Mail sent\n" : "Mail failed\n" );
```

Запуск трюка

В сценарии измените адрес электронной почты и имена. Затем при помощи PHP-интерпретатора для командной строки запустите сценарий:

```
% php htmlmail.php
Mail sent
%
```

На моем компьютере под управлением OS X сообщение приходит довольно быстро в почтовый клиент Mail, как это показано на рис. 2.14.

Рис. 2.14. Приложение Mail.app, отображающее пришедшее сообщение

При двойном щелчке кнопкой мыши на сообщении можно прочитать его содержимое. Вы можете видеть форматирование при помощи HTML с применением маркированного списка на рис. 2.15.

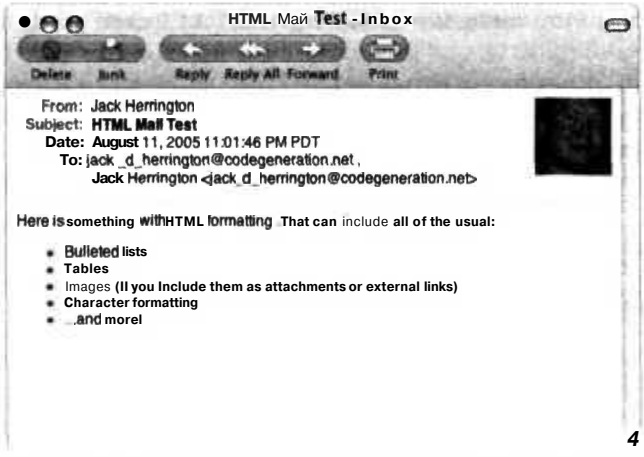


Рис. 2.15. Отформатированное при помощи HTML сообщение

Все стандартные варианты разметки работают, так как они поддерживаются встроенным в почтовый клиент браузером.

Единственная проблема — рисунки, расположенные не локально, а на других ресурсах. У вас есть два варианта размещения изображений. Первый — прописать ссылку на объект, находящийся на удаленном веб-сервере. И тут появляются некоторые проблемы. Во-первых, такой вариант не подходит, если вы читаете почту, не находясь в Сети. Во-вторых, во многих почтовых клиентах выключена возможность загрузки удаленных файлов с изображениями как средство борьбы

со спамерами, которые пользуются этим для подтверждения того, что сообщение дошло до адресата (в надежде отправить как можно больше спама). Второй вариант — добавить изображение как прикрепленный к другой части составного сообщения файл. Это сработает, но сами по себе сообщения по электронной почте будут большего размера из-за использования 64-битного кодирования изображения. Я пришел к выводу, что проще обойтись стандартным форматированием текстового сообщения плюс использовать CSS, если вы хотите отправить сообщение, содержащее HTML. Но становится очевидно, что это не выход, если пользователю необходимо именно изображение в вашем сообщении. Как вы уже поняли из данной главы, вы можете сделать очень многое, используя только HTML (см. трюк 8).

Смотрите также

J «Создание очереди сообщений» (см. трюк 50).

DHTML

Трюки 11-26

Эта глава посвящена использованию динамического HTML (DHTML) в ваших интернет-приложениях. DHTML — термин, используемый для обозначения мощной комбинации HTML, CSS и JavaScript. В трюках, описанных в данной главе, используется DHTML для создания всех возможных типов интерактивных интерфейсов ваших интернет-приложений, начиная с таблиц и заканчивая динамическими графиками, всплывающими окнами, возможностью просматривать слайды, векторной графикой и многим другим.

ТРЮК №11

Размещение интерактивных таблиц на вашей странице

Используйте библиотеку таблиц ActiveWidgets для размещения интерактивных компонентов управления данными на вашей странице.

Согласитесь, некоторая информация, в частности финансовая или статистическая, намного нагляднее отображается в виде таблицы. К сожалению, HTML предоставляет довольно бедные возможности в представлении информации в форме интерактивных таблиц, особенно когда возникает необходимость их просматривать, сортировать или обрабатывать каким-либо действительно интерактивным способом, чего и ожидает пользователь от элементов таблицы.

В этом трюке используется компонент в форме сетки ActiveWidgets (http://activewidgets.com/) для создания интерфейса с размещенными на веб-странице таблицами.

Код

Сохраните исходный код из примера 3.1 в файл index.php.

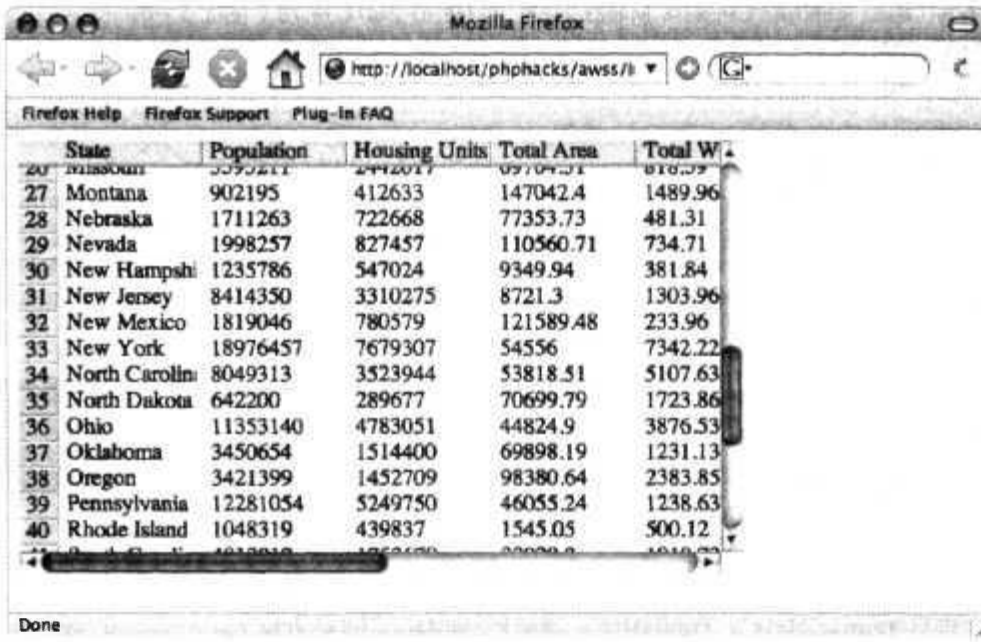
Пример 3.1. Сценарий, который выводит специфическую информацию в виде таблицы

```
<?php $states = array(
array( "Alabama",4447100,1963711,52419.02,1675.01,50744,87.6,38.7 ),
array( "Alaska",626932,260978,663267.26,91316,571951.26,1.1,0.5 ),
array( "Arizona",5130632,2189189,113998.3,363.73,113634.57,45.2,19.3 ),
array( "Arkansas",2673400,1173043,53178.62,1110.45,52068.17,51.3,22.5 ),
```

```
array( "California",33871648,12214549,163695.57,7736.23,155959.34,217.2,78.3 ),
array( "Colorado",4301261,1808037,104093.57,376.04,103717.53,41.5,17.4 ),
array( "South Dakota",754844,323208,77116.49,1231.85,75884.64,9.9,4.3 ).
...
array( "Tennessee",5689283,2439443,42143.27,926.15,41217.12,138,59.2 ).
array( "Texas",20851820,8157575,268580.82,6783.7,261797.12,79.6,31.2 ).
array( "Utah",2233169,768594,84898.83,2755.18,82143.65,27.2,9.4 ),
array( "Vermont",608827,294382,9614.26,364.7,9249.56,65.8,31.8 ).
array( "Virginia",7078515,2904192,42774.2,3180.13,39594.07,178.8,73.3 ).
array( "Washington",5894121,2451075,71299.64,4755.58,66544.06,88.6,36.8 ).
array( "West Virginia",1808344,844623,24229.76,152.03,24077.73,75.1,35.1 ),
array( "Wisconsin",5363675,2321144,65497.82,11187.72,54310.1,98.8,42.7 ).
array( "Wyoming",493782,223854,97813.56,713.16,97100.4,5.1,2.3 ),
arrayt "Puerto Rico",3808610,1418476,5324.5,1899.94,3424.56,1112.1,414.2 )
);
?>
<html>
<head>
<link href-"runtime/styles/xp/grid.css" rel="stylesheet" type="text/css" ></link>
<script src="runtime/lib/grid.js"></script>
</head>
<body>
<div style="width:500px;height:300px;">
<script>
var data - [
<?php $first = true; foreach( $states as $state ) { if ( !$first ) echo( "," );
?>
[ "<?php echo($state[0]); ?>". <?php echo($state[1]); ?>,
<?php echo($state[2]); ?>. <?php echo($state[3]); ?>.
<?php echo($state[4]); ?>. <?php echo($state[5]); ?>,
<?php echo($state[6]); ?>. <?php echo($state[7]); ?> ]
<?php $first - false; } ?>
];

var columns = [ "State". "Population". "Housing Units", "Total Area".
"Total Water". "Total Land", "Population Density". "Housing Density" ];

function dataLookupt row. col )
{
  return data[row][col];
}

function headerLookup( col )
{
  return columns[ col ];
}

var grid = new Active.Controls.Grid;
grid.setRowCount( data.length );
grid.setColumnCount( columns.length );
grid.setDataText( dataLookup );
grid.setColumnText( headerLookup );
document.write( grid );
</script>
</div>
</body>
</html>
```

Запуск трюка

Загрузите библиотеку ActiveWidgets для использования компонента `grid` и распакуйте ее. Далее переместите папку с полученными файлами на ваш сервер и поместите файл `index.php` в той же директории, в которой расположены файлы библиотеки Active Widgets. Затем откройте в вашем браузере i `ndex.php`; вы должны увидеть окно, похожее на изображенное на рис. 3.1.



	State	Population	Housing Units	Total Area	Total W
27	Montana	902195	412633	147042.4	1489.96
28	Nebraska	1711263	722668	77353.73	481.31
29	Nevada	1998257	827457	110560.71	734.71
30	New Hampshi	1235786	547024	9349.94	381.84
31	New Jersey	8414350	3310275	8721.3	1303.96
32	New Mexico	1819046	780579	121589.48	233.96
33	New York	18976457	7679307	54556	7342.22
34	North Carolin	8049313	3523944	53818.51	5107.63
35	North Dakota	642200	289677	70699.79	1723.86
36	Ohio	11353140	4783051	44824.9	3876.53
37	Oklahoma	3450654	1514400	69898.19	1231.13
38	Oregon	3421399	1452709	98380.64	2383.85
39	Pennsylvania	12281054	5249750	46055.24	1238.63
40	Rhode Island	1048319	439837	1545.05	500.12

Рис.3.1. Компонент grid библиотеки ActiveWidgets с информацией о городах США

На этой простой странице первым делом выполняется подключенный код на JavaScript, предназначенный для управления компонентом grid из библиотеки ActiveWidgets. Затем в массив, созданный при помощи JavaScript, заносится информация и создается элемент управления компонентом. После этого сценарий ищет информацию при помощи функции `dataLookup()`, которая возвращает информацию из заданной колонки и столбца.

ВНИМАНИЕ

На момент написания данной книги библиотека для работы с компонентом grid не была совместима с браузером Safari от Apple.

Лицензия на данную библиотеку принадлежит GNU Public License (GPL). Если вы хотите использовать ее в коммерческих продуктах, то необходимо будет приобрести ее у разработчиков. Но если вы ищете подходящий компонент для HTML в форме таблицы, пожалуй, это будет стоить затраченных денег.

Смотрите также

J «Создание списков с использованием Drag & Drop» (см. трюк 13).

- «Создание диаграмм ссылок» (см. трюк 24).

❑ «Создание интерактивного календаря» (см. трюк 25).

ТРЮК №12

Создание всплывающих подсказок

Используйте библиотеку overLIB для создания всплывающих подсказок на вашей странице при помощи PHP и JavaScript.

Используя библиотеку overLIB для JavaScript (http://www.bosrup.com/web/overlib/), вы можете создавать удобные всплывающие метки, которые появляются поверх текста на вашей странице. В описанном ниже трюке показано, как при помощи оберточной функции на PHP воспользоваться библиотекой для более простого способа создания такого рода ссылок.

Код

Сохраните исходный код из примера 3.2 в файл index.php.

Пример 3.2. Оберточная функция для упрощения использования библиотеки overLIB при помощи PHP

```
<?php
function popup( $text, $popup )
{
?>
<a href="javascript:void(0);" onmouseover="return overlib('<?php echo($popup); ?>
' ) ; " onmouseout="return nd( );"><?php echo($text); ?></a>
<?php
}
?>
<html>
<head>
<script type="text/javascript" src="overlib.js"><!-- overLIB (c) Erik Bosrup -->
</script>
</head>
<body>
<div id="overDiv" style="position:absolute; visibility:hidden; z-index:1000;">
</div>
So this is just a test of popups. Not something interesting about <?php popup(
'rabbits', 'Small furry woodland creatures.<br/>Rabbits also make good pets.'
); ?>. Because that would just be silly.
</body>
</html>
```

Вы также можете разместить оберточную функцию в отдельной PHP-библиотеке, подключать ее при создании ваших страниц и таким образом создать действительно полезную и готовую к многократному использованию функцию.

Запуск трюка

Загрузите и распакуйте библиотеку overLIB на ваш сервер в директорию для документов. Затем добавьте файл index.php и перейдите к нему в окне вашего браузера (рис. 3.2).

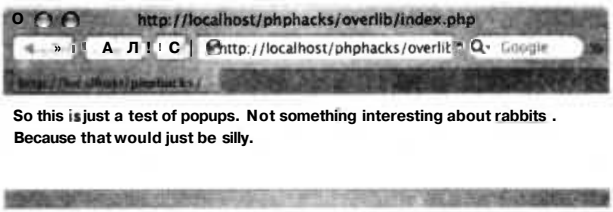


Рис. 3.2. Пример страницы с всплывающей ссылкой

Далее переместите указатель мыши к слову rabbits, и вы увидите всплывающую надпись, которая дает вам некоторую дополнительную информацию по выбранному слову (как видно на рис. 3.3).

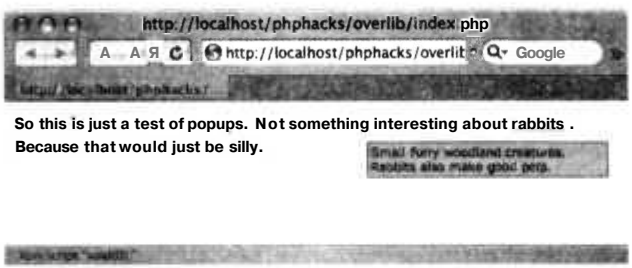


Рис. 3.3. Наведение указателя мыши на всплывающую ссылку

Эта всплывающая надпись может быть доработана как вам будет угодно: можно будет добавить изображения, таблицы, различные шрифты, стили и многое другое.

Смотрите также

- «Создание раскрывающихся вкладок» (см. трюк 16).

ТРЮК №13

Создание списков с использованием Drag & Drop

Используйте JavaScript, DHTML и PHP для создания и использования списков с применением Drag & Drop.

Создание интерфейса, который позволил бы пользователям эффективно работать со списками, всегда было большой проблемой при работе с HTML. Но теперь,

при помощи PHP, это больше не вызывает никаких затруднений. В этом трюке используется библиотека с открытым кодом Tool-Man (http://tool-man.org/) для создания Drag & Drop-списков.

Код

Сохраните исходный код из примера 3.3 в файл index.html.

Пример 3.3. Создание списка с применением Drag & Drop при использовании HTML и CSS

```
<html>
<head>

<style>

#states li { margin: Opx; }

ul.boxy li { margin: 3px: }

ul.sortable li { position: relative: }

ul.boxy
{
  list-style-type: none:
  padding: Opx;
  margin: 2px;
  width: 20em;
  font-size: 13px:
  font-family: Arial, sans-serif;
}
ul.boxy li
{
  cursor:move:
  padding: 2px 2px;
  border: 1px solid #ccc:
  background-color: #eee;
}
.clickable a
{
  display: block;
  text-decoration: none:
  cursor: pointer;
  cursor: hand;
}
 clickable li:hover
{
} background-color: #f6f6f6:

</style>
<script language="JavaScript" type="text/javascript"
        src="source/org/tool-man/core.js"></script>
<script language="JavaScript" type="text/javascript"
        src="source/org/tool-man/events.js"></script>
<script language="JavaScript" type="text/javascript"
        src="source/org/tool-man/css.js"></script>
```

```
<script language="JavaScript" type="text/javascript"
        src="source/org/tool-man/coordinates.js"></script>
<script language="JavaScript" type="text/javascript"
        src="source/org/tool-man/drag.js"x/script>
<script language-"JavaScript" type="text/javascript"
        src="source/org/tool-man/dragsort.js"x/script>
<script language="JavaScript" type="text/javascript"
        src="source/org/tool-man/cookies.js"></script>

<script language="JavaScript" type="text/javascript">

var dragsort = ToolMan.dragsort( )
var junkdrawer = ToolMan.junkdrawer( )

window.onload = function( )
{
  dragsort.makeListSortable(document.getElementById("states"),
  verticalOnly, saveOrder)
}

function verticalOnly(item) { item.toolManDragGroup.verticalOnly( ) }

function saveOrder(item) { }

function prepFields( )
{
  document.getElementById( "states_text" ).value = junkdrawer.
  serializeList( document.getElementById( "states" ) ):
  return true;
}
//-->
</script>
</head>
<body>

<ul id="states" class="boxy">
<li>California</li>
<li>Texas</li>
<li>Alaska</li>
</ul>

<form method="post" action="tellme.php">
<input type="hidden" name="states" value-"" id="states_text" />
<input type="submit" onclick="return prepFields( );">
</form>

</body>
</html>
```

Сохранив пример 3.4 в файл tellme.php, можно вывести на экран все значения массива.

Пример 3.4. Выводим значения массива

```
<body>
<html>
You chose: <?php echo( $_POST['states'] ): ?>
</html>
</body>
```

Запуск трюка

Загрузите и распакуйте библиотеки для использования Drag & Drop на ваш веб-сервер. Затем загрузите туда файлы с исходным кодом данного трюка и перейдите в браузере к файлу `index.html` (рис. 3.4).

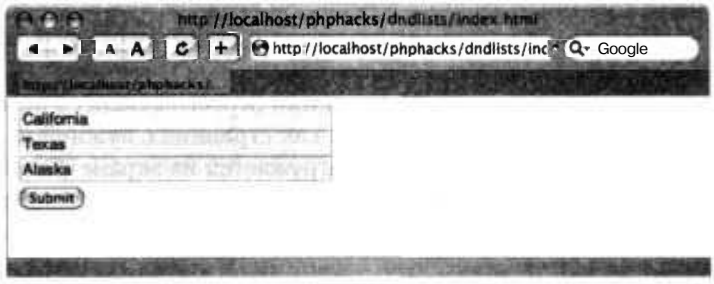


Рис. 3.4. Список с применением Drag & Drop

Теперь разместите строки с помощью мыши в том порядке, в каком вам нравится, после чего нажмите кнопку Submit (Отправить). В этом случае содержимое списка будет передано в скрытую переменную под названием states и загружено на сервер. Сценарий `tellme.php` выводит значения переменных в установленном вами порядке (рис. 3.5).

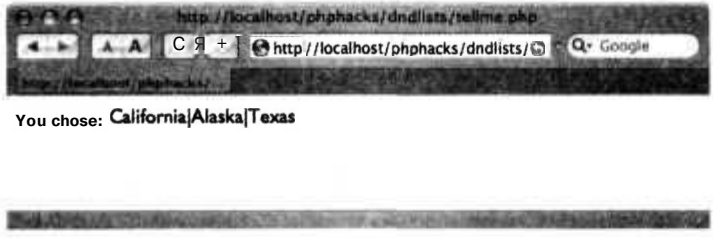


Рис. 3.5. Вид окна после нажатия кнопки Submit

Небольшие динамические элементы интерфейса, как этот, могут различаться в простоте использования для ваших интернет-приложений и приложений другого типа. Иногда очень полезно иметь возможность придать им немного лучший вид во время демонстрации работы! Используя сохраненное после перетаскивания значение переменной, вы можете легко получить доступ к данным при помощи PHP и сделать с ними все, что угодно.

Смотрите также

- «Создание всплывающих подсказок» (см. трюк 12).
- «Создание раскрывающихся вкладок» (см. трюк 16).
- «Создание палитры для выбора цвета» (см. трюк 23).
- «Создание интерактивного календаря» (см. трюк 25).

ТРЮК №14

Создание динамических графиков на HTML

Используя DHTML, вы можете создавать графики, которые будут изменяться даже без перезагрузки страницы. Что в результате? Ваши пользователи могут работать с данными в реальном времени.

Иногда Интернет работает из рук вон плохо. Вы щелкаете кнопкой мыши на ссылке, и этот симпатичный крутящийся шарик или пересыпающиеся песочные часики усердно продолжают работу, в то время как страница с нужной вам информацией (как хочется надеяться) по частям загружается на экране. Естественно, это не та интерактивность, которую мы привыкли наблюдать со стороны нашего клиентского программного обеспечения. Но, слава богу, вы можете создать приложение, которое работает без перезагрузки страницы. В этом трюке мы покажем, как создать интерактивную диаграмму рассеивания, используя несколько графиков, PHP и обширные возможности JavaScript.

Код

Содержимое файла index.php показано в примере 3.5.

Пример 3.5. JavaScript – настоящая рабочая лошадка в данном трюке

```
<?php $states - array(
array( "Alabama",4447100,1963711,52419.02,1675.01,50744,87.6,38.7 ),
array( "Alaska",626932,260978,663267.26,91316,571951.26,1.1,0.5 ),
array( "Arizona",5130632,2189189,113998.3,363.73,113634.57,45.2,19.3 ),
array( "Arkansas",2673400,1173043,53178.62,1110.45,52068.17,51.3,22.5 ),
array( "California",33871648,12214549,163695.57,7736.23,155959.34,217.2,78.3 ),
array( "Colorado",4301261,1808037,104093.57,376.04,103717.53,41.5,17.4 ),
...
array( "Washington",5894121,2451075,71299.64,4755.58,66544.06,88.6,36.8 ),
array( "West Virginia",1808344,844623,24229.76,152.03,24077.73,75.1,35.1 ),
array( "Wisconsin",5363675,2321144,65497.82,11187.72,54310.1,98.8,42.7 ),
array( "Wyoming",493782,223854,97813.56,713.16,97100.4,5.1,2.3 ),
array( "Puerto Rico",3808610,1418476,5324.5,1899.94,3424.56,1112.1,414.2 )
);
?>
<html>
<head>
<script language="Javascript">
var width = 300;
var height = 300;

var axes = [ "population", "housing_units", "total_area", "total_water", "total_
land", "people_density", "housing_density" ];

var data = [
<?php $first = true; foreach( $states as $state ) { if ( !$first ) echo( "," );
?>
{ state: "<?php echo($state[0]); ?>", population: <?php echo($state[1]); ?>,
```

```
housing_units: <?php echo($state[2]); ?>. total_area: <?php echo($state[3]); ?>.
total_water: <?php echo($state[4]); ?>, total_land: <?php echo($state[5]); ?>.
people_density: <?php echo($state[6]); ?>. housing_density: <?php
echo($state[7]); ?> }
<?php $first = false; } ?>
];

var axmin = {};
var axmax = {};

for( axind in axes )
{
  axmin[ axes[axind] ] = 100000000;
  axmax[ axes[axind] ] = -100000000;
}
fort ind in data )
{
  row = data[ind];
  for( axind in axes )
  {
    axis = axes[axind];
    if ( row[axis] < axmin[axis] )
      axmin[axis] = row[axis];
    if ( row[axis] > axmax[axis] )
      axmax[axis] = row[axis];
  }
}

function cleargraph( )
{
  graph = document.getElementById( "graph" );
  graph.innerHTML = "";
}

function adddot( value, size, x, y, text )
{
  var left = x - ( size / 2 );
  var top = width - ( y + ( size / 2 ) );
  var cleft = "auto";
  var ctop = "auto";
  var cright = "auto";
  var cbottom = "auto";

  if ( left < 0 ) { cright = ( left * -1 ) + "px"; }
  if ( left + size > width ) { cleft = ( width - left ) + "px"; }
  if ( top < 0 ) { ctop = ( top * -1 ) + "px"; }
  if ( top + size > height ) { cbottom = ( height - top ) + "px"; }

  if ( value <= 0.25 )
    img = "ltgray.gif";
  else if ( value <= 0.50 )
    img = "gray.gif";
  else if ( value <= 0.75 )
    img = "dkgray.gif";
  else
```

```
    img = "black.gif";

  html = "<img src=\""+img+"\" width=\""+size+"\" height=\""+size+"\" ";
  html += "style=\"position:absolute;left:"+left+"px;top:"+top+"px;";
  html += "clip:rect( "+ctop+" "+cleft+" "+cbottom+" "+cright+" );";
  html += "\" onclick=\"alert(\'"+text+"\')\"/>";

  graph = document.getElementById( "graph" );
  graph.innerHTML += html;
}

function calculate_value( row, field, min, max )
{
  var val = row[ field ] - axmin[ field ];
  var scale = ( max - min ) / ( axmax[ field ] - axmin[ field ] );
  return min + ( scale * val );
}

function drawgraph( )
{
  cleargraph( );

  var xvar = document.getElementById( "bottom" ).value;
  var yvar = document.getElementById( "side" ).value;
  var sizevar = document.getElementById( "size" ).value;
  var valuevar = document.getElementById( "color" ).value;

  for( rowind in data )
  {
    var row = data[rowind];
    var x = calculate_value( row, xvar, 5, width - 5 );
    var y = calculate_value( row, yvar, 5, height - 5 );
    var size = calculate_value( row, sizevar, 5, 30 );
    var value = calculate_value( row, valuevar, 0, 1 );
    adddot( value, size, x, y, row.state );
  }
}

function buildselect( axis, current )
{
  var html = "<select id=\""+axis+"\" onchange=\"drawgraph( )\">";
  for( axind in axes )
  {
    var selected = "";
    if ( axes[axind] == current )
      selected = " selected=\"true\"";
    html += "<option value=\""+axes[axind]+"\""+selected+">"+axes[axind]+"
    </option>";
  }
  html += "</select>";
  document.write( html );
}
</script>
</head>
<body onload="drawgraph( );">
```

```
Side: <script language="Javascript">buildselect( "side", "population" );
</script>
Bottom: <script language="Javascript">buildselect( "bottom", "housing_units" );
</script>
Size: <script language="Javascript">buildselect( "size", "total_area" );
</script>
Color: <script language="Javascript">buildselect( "color", "total_water" );
</script>
<div style="position:relative;border:1px solid #eee; clip:rect(0px 0px 300px
300px); width:300px; height:300px;" id="graph">
</div>
</body>
</html>
```

Для начала сценарий создает массив данных — сперва на PHP, а затем на JavaScript. Необходимо, чтобы данные были доступны для JavaScript, таким образом, график может быть создан динамически при использовании DHTML. Как только информация была загружена на страницу, дальнейшая работа ложится на браузер. Первое, что он делает, — создает раскрывающееся поле ввода, используя функцию `buildselect()`. Затем вызывается обработчик события onloadO, который переходит к тегу `<body>`, в котором вызывается функция drawgraph `()`. Эта функция, в свою очередь, создает новую строку для HTML, которая состоит из большого количества тегов `<img>`, один для каждого из возможных состояний. Размер изображения, как и его месторасположение на графике, зависит от того, какие значения атрибутов для графика были выбраны из раскрывающегося поля ввода. Когда график создан, функция drawgraph () задает внутри тега <di v> новый HTML для графика, перерисовывая все изображения. Если пользователь снова выберет другой атрибут в раскрывающемся поле ввода, то функция `drawgraph()` будет вызвана заново и график будет перерисован.

Запуск трюка

Загрузите PHP-страницу и изображения на сервер, а затем откройте страницу в браузере (рис. 3.6).

Теперь воспользуйтесь полями ввода, чтобы задать другие значения для различных частей графика. В поле ввода Side изменяется значение по оси Y. В поле ввода Bottom изменяется значение по оси X. С помощью поля Size можно задать размер шарика, а в поле Color можно изменить его цвет. Ось Y отвечает за количество жилых единиц, а увеличение площади происходит по оси X. В целом весь график выглядит достаточно компактно, однако все же есть несколько всплесков. Щелкните кнопкой мыши на одном из них, чтобы узнать, информацию о чем эти всплески отражают. Например, щелкнув кнопкой мыши на черном шарике в нижнем правом углу, можно увидеть всплывающее сообщение (рис. 3.7). В нем указано, что Аляска обладает наибольшей территорией, но на ней живет очень мало людей. На графике присутствуют еще два всплеска: серого цвета шарик в центре графика (Техас) и шарик в самом верху (Калифорния).

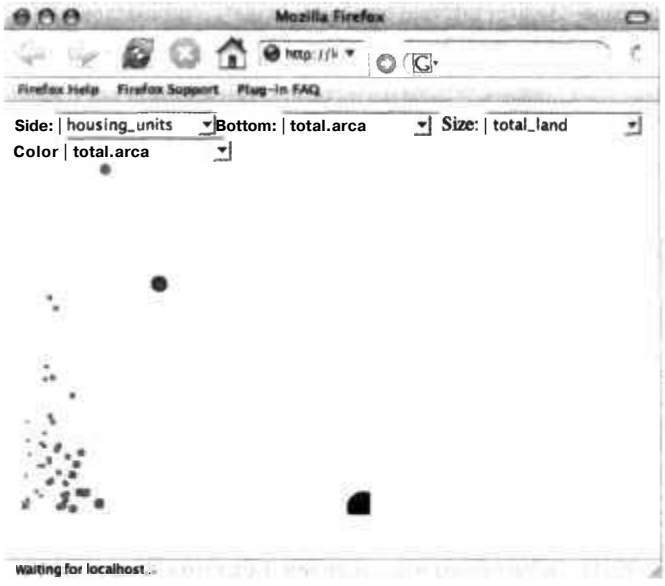


Рис. 3.6. Диаграмма разброса данных после нескольких изменений

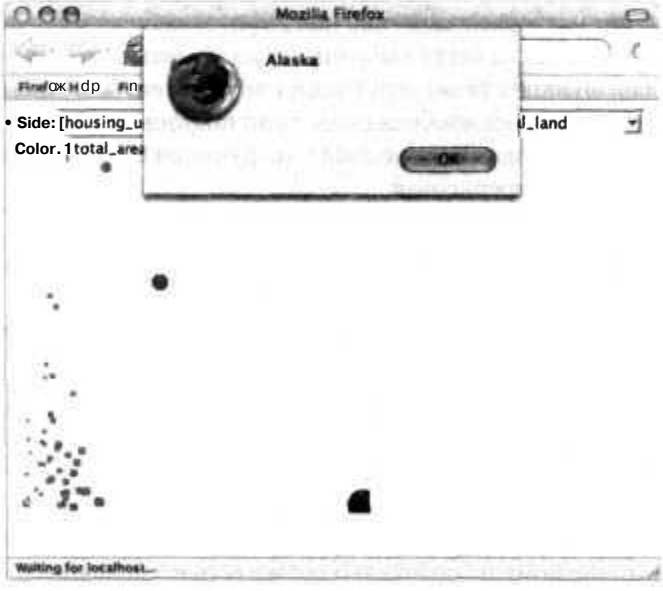


Рис. 3.7. Сообщение, открываемое после щелчка кнопкой мыши на шарике в правом нижнем углу

ПРИМЕЧАНИЕ

Информация для данного графика предоставлена Бюро переписи населения США (http://www.census.gov/). Когда вам необходимы какие-либо данные, этот сайт всегда к вашим услугам. Информация на нем предоставлена в текстовом виде, в форме блоков данных, разделенных запятой (CSV— comma-separated value) и в XLS-формате (Excel).

Смотрите также

J «Создание легковесных графиков на HTML» (см. трюк 8).

ТРЮК № 15

Разбиение содержимого страницы на части при помощи разделителей

Используйте разделители, чтобы разбивать содержимое вашей страницы на отдельные составные части, каждую из которых вы можете отобразить или скрыть независимо от других.

Иногда объем информации чересчур велик, чтобы пытаться отобразить ее всю на одной странице. Одним из выходов из данной ситуации является использование вкладок (см. трюк 6), а второй способ — разбиение информации при помощи разделителей, что позволяет пользователям открывать для просмотра какую-то определенную часть содержимого. В этом трюке показано, как создавать на странице секции при помощи разделителей, которые позволяют интерактивно отображать ту часть, которая нам необходима.

Код

Исходный текст файла index.php показан в примере 3.6.

Пример 3.6. Код на PMP, позволяющий пользователю выбирать конкретный разделитель

```
<?php

function start_section( $id, $title )
{
?>
<table cellspacing="0" cellpadding="0">
<tr>
<td width="30" valign="top">
<a href="javascript: void twist('<?php echo($id); ?>'):">
<img src="up.gif" border="0" id="img_<?php echo($id); ?>"/>
</a>
</td>
<td width="90%">
<h1><?php echo( $title ); ?></h1>
<div style="visibility:hidden;position:absolute;"
id="<?php echo($id); ?>" class="spin-content">
<?php
}

function end_section( )
{
?>
</div>
</td>
</tr>
</table>
```

```
<?php
}

function spinner_header( )
{
?>
<style type="text/css">
body { font-family: arial, verdana; }
h1 { font-size: medium; border-bottom: 1px solid black; }
.spin-content { font-size: small: margin-left: 10px; padding: 10px: }
</style>
<script language="Javascript">

function twist( sid )
{
  imgobj = document.getElementById( "img_"+sid ):
  divobj = document.getElementById( sid ):
  if ( imgobj.src.match( "up.gif" ) )
  {
    imgobj.src = "down.gif":
    divobj.style.position - "relative";
    divobj.style.visibility - "visible":
  }
  else
  {
    imgobj.src = "up.gif":
    divobj.style.visibility = "hidden";
    divobj.style.position - "absolute";
} }
</script>
<?php
}
?>
<html>
<head>
<?php spinner_header( ) ?>
</head>
<body>
<?php start_section( "one". "Report part one" ) ?>
This report will tell you a lot of stuff you didn't know before.
And that's good. Because that's what a report should do.<br/><br/>
But it will tell you so much that it needs to be rolled up into sections
so that you don't have to gasp as you see it all at once.
<?php end_section( ) ?>
<?php start_section( "two". "Report part two" ) ?>
This is a table of numbers and such:<br/>
<table>
<tr><th>State</th><th>Total</th></tr>
<tr><td>CA</td><td>$35M</td></tr>
<tr><td>PA</td><td>$22M</td></tr>
<tr><td>NC</td><td>$5M</td></tr>
<tr><td>FL</td><td>$15M</td></tr>
</table>
<?php end_section( ) ?>
</body>
</html>
```

Сперва в сценарии определяются функции start_section() и end_section(), заключающие в скобки определенный блок содержимого страницы, который будет интерактивно отображаться (либо скрываться). В начальной секции также определяется функция spinner_header(), содержащая участки кода на CSS и JavaScript, используемые DHTML.

ПРИМЕЧАНИЕ

Вы можете разместить весь этот код в отдельном PHP-файле, если хотите разбить исходный текст программы на модули.

Функция start_section() создает таблицу, в первом столбце которой хранятся изображения, используемые разделителем, а во втором — название секции и элемент DIV, в котором хранится само содержание. Элемент DIV по умолчанию невидимый, и значение его атрибута position равняется "absolute" (если у элемента DIV значение атрибута равно "relative", будет считаться, что он все равно присутствует на разметке страницы). Чтобы элемент DIV исчез, значение атрибута месторасположения устанавливается в "absolute". Позже оно может быть изменено на "relative", чтобы снова сделать элемент видимым.

Запуск трюка

Загрузите исходный код и изображения на сервер и перейдите в браузере к файлу index.php (рис. 3.8).

Рис. 3.8. Скрытые секции

Щелкните кнопкой мыши на стрелке перед первой частью отчета, и вы увидите, как раскроется секция с отчетом (рис. 3.9).

ПРИМЕЧАНИЕ

Up.gif и down.gif— это два файла с изображениями, которые были созданы при помощи Macromedia Fireworks. Их также можно легко создать при помощи Adobe Photoshop или графического редактора с открытым кодом GIMP. Если вы не хотите создавать свои собственные изображения, я рекомендую воспользоваться системой поиска картинок Google, чтобы подобрать подходящие в форме стрелки.

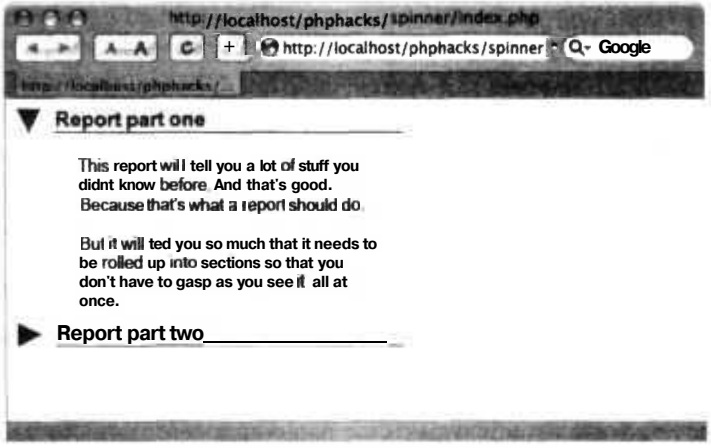


Рис.3.9. Развернутая первая секция

Кода вы будете писать код, отвечающий за динамическое размещение, как в этом трюке, очень важно проверить его на всех браузерах, которые вам доступны. Размещение видимых и невидимых элементов — одна из тонкостей DHTML, которую необходимо хорошо изучить.

Смотрите также

- «Добавление вкладок к вашему веб-интерфейсу» (см. трюк 6).

ТРЮК №16

Создание раскрывающихся вкладок

Используйте DHTML для размещения раскрывающихся окошек к ключевым словам в HTML.

Прикрепление раскрывающихся вкладок к словам или фразам в вашем документе — это простой способ добавить ценную информацию, не мешая отображению основной. Пользователь может щелкнуть кнопкой мыши на слове и получить дополнительную информацию, и все это произойдет без лишней прокрутки текста или лишних движений мышью.

Код

Сохраните исходный код из примера 3.7 в файле index.php.

Пример 3.7. Совместная работа PHP и JavaScript для создания раскрывающихся вкладок

```
<?php
$nextid = 1;
function start_link( $text )
{
  global $nextid;
```

```
  $idtext - "a"+$nextid:
?><a href="javascript: void drop( '<?php echo($idtext); ?>' );">
<span id="a_<?php echo($idtext); ?>"><?php echo($text); ?></span></a>
<div id="<?php echo($idtext); ?>" class="drop" style="visibility:hidden;">
<table cellspacing="0" cellpadding="0" width="170"><tr>
<td valign="top" width="20">
<a href="javascript: void close(<?php echo($idtext); ?>)"><img src="close.gif"
border="0"></a>
</td>
<td valign="top" width="150">
<?php
}

function end_link( )
{
?>
</td>
</tr></table>
</div><?php
}

function link_header( )
{
?>
<style type="text/css">
body { font-family: arial, verdana; }
.drop {
  padding: 5px;
  font-size: small;
  background: #eee;
  border: 1px solid black;
  position: absolute;
}
</style>
<script language="Javascript">
function drop( sid )
{
  aobj = document.getElementById( "a_"+sid );
  divobj = document.getElementByIdC sid );
  divobj.style.top = aobj.offsetBottom+10;
  divobj.style.left - aobj.offsetLeft+10;
  divobj.style.visibility - "visible";
}

function close( sid )
{
  divobj = document.getElementById( sid );
  divobj.style.visibility = "hidden";
}
</script>
<?php
}
?>
<html>
<head>
<?php link_header( ); ?>
```

```
</head>
<body>
Hey <?php start_link( "this is interesting" ): ?>
That really<br/>
Is interesting <?php end_link( ): ?>. How about that.
<br/>
The popup will go over text and all that.<br/>
And it will stay up until it's dismissed with the close
button.
</body>
</html>
```

В этом сценарии определены три функции в начале файла: `start_link()`, `end_link()` и `link_header()`. При вызове функции `start_link()` в нее в качестве аргумента передается ссылка. Далее заполняется содержимое раскрывающейся вкладки и вызывается функция `end_link()`.

ПРИМЕЧАНИЕ

Чтобы разместить классы CSS и JavaScript в заголовке, вам необходимо вызвать функцию link_header() внутри тега <head>.

Запуск трюка

Скопируйте исходный код и изображения на сервер. Запустите в вашем браузере сценарий `index.php`. Результат вы можете видеть на рис. 3.10.

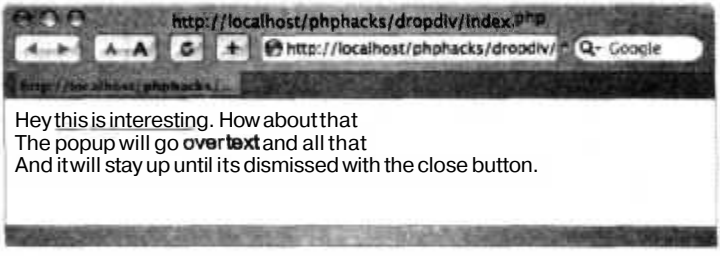


Рис. 3.10. Ключевое слово в документе, реагирующее на щелчок кнопкой мыши

Теперь щелкните кнопкой мыши на ссылке, и вы увидите раскрывающуюся вкладку со значком, предназначенным для закрытия окошка (рис. 3.11).

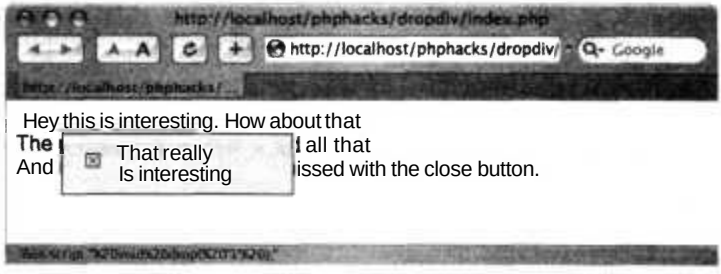


Рис. 3.11. Раскрывающаяся вкладка, размещенная под ссылкой

Смотрите также

- «Создание всплывающих подсказок» (см. трюк 12).

ТРЮК №17

Создание динамических меню навигации

Используйте PHP для создания виджета меню навигации, которое бы работало на всем вашем сайте.

Создание меню навигации для вашего сайта может стать настоящей головной болью. Вам ведь не хочется писать один и тот же код раз за разом для каждой страницы? В идеале хотелось бы создать меню в виде PHP-функции, которая бы формировала меню с помеченной текущей страницей. В этом трюке вам предоставляется простейшая функция для создания меню (и не более того из-за низкой стоимости данной книги!).

Код

Сохраните исходный код из примера 3.8, который демонстрирует использование библиотеки menu.php, в файл index.php.

Пример 3.8. Использование библиотеки для создания меню

```
<?php
require_once( "menu.php" );

$page = "home";
if ( $_GET['page'] )
  $page = $_GET['page'];
?>
<html>
<head>
<title>Page - <?php echo($page); ?></title>
<?php echo menu_css( ); ?>
</head>
<body>
<table cellspaceing="0" cellpadding="5">
<tr>
<td width="200" valign="top">
<?php page_menu( $page ); ?>
</td>
<td width="600" valign="top">
Page: <?php echo( $page ); ?>
</td>
</tr>
</table>
</body>
</html>
```

В примере 3.9 показана библиотека, размещенная в файле menu.php.

Пример 3.9. Все работает с использованием библиотеки на PHP

```
<?php
function menu_css( ) {
?>
<style type="text/css">
 menu-inactive, .menu-active {
   padding: 2px;
   padding-left: 20px;
   font-family: arial, verdana;
}
 .menu-inactive { background: #ddd: }
 .menu-active { background: #000; font-weight: bold; }
 .menu-inactive a { text-decoration: none; }
 .menu-active a { color: white: text-decoration: none: }
</style>
<?php
}

function menu_item( $id, $title, $current ) {
$class = "menu-inactive";
if ( $current — Sid )
  $class = "menu-active";
?>
<tr><td class="<?php echo($class); ?>">
<a href="index.php?page=<?php echo( Sid ); ?>">
<?php echo( $title ): ?>
</a>
</td></tr>
<?php
}

function page_menu( $page ) {
?>
<table width="100%">
<?php menu_item( 'home', 'Home', $page ); ?>
<?php menu_item( 'faq', 'FAQ', Spage ); ?>
<?php menu_item( 'download', 'Download', Spage ): ?>
<?php menu_item( 'links', 'Links', Spage ): ?>
<?php menu_item( 'credits', 'Credits', Spage ); ?>
</table>
<?php
}
?>
```

Файл index.php создает меню, вызывая функцию page_menu() и передавая в нее ID страницы. ID используется для определения выбранного пункта меню. Сценарий index.php также вызывает функцию menu_css() для задания CSS-стиля для меню.

Вы можете подкорректировать состав меню, внеся изменения в конце файла menu. php и таким образом добавив либо удалив элементы меню. Вы также можете изменить внешний вид меню, изменив определения CSS-класса в функции menu_css().

ПРИМЕЧАНИЕ

Очевидно, что вам придется также изменить названия пунктов меню и страниц в этом сценарии, чтобы он соответствовал структуре вашего сайта.

Запуск трюка

Загрузите исходный код на ваш сервер и откройте в браузере файл `index.php` (рис. 3.12).

Рис. 3.12. Домашняя страница

Теперь щелкните кнопкой мыши на ссылке FAQ. Рядом с меню будет указано, какая открыта страница (рис. 3.13).

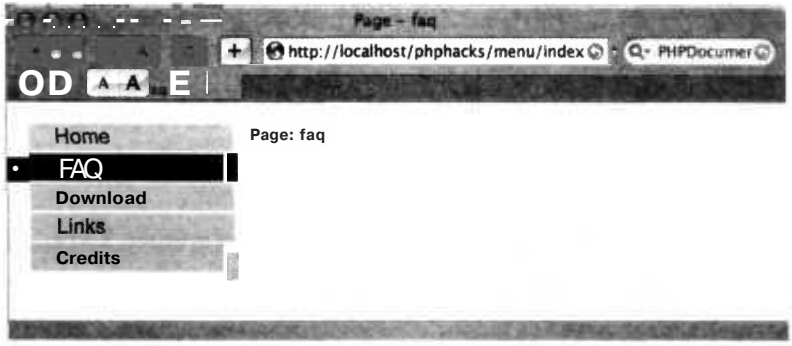


Рис. 3.13. Страница FAQ

Смотрите также

- «Создание элементов внутренней ссылочной структуры сайта» (см. трюк 4).

ТРЮК №18

Динамическое скрытие кода JavaScript

Используйте PHP для скрытия названий основных рабочих функций, написанных на JavaScript.

Иногда бывает очень полезно скрывать код, написанный на JavaScript, от постороннего глаза для защиты интеллектуальной собственности. Невозможно настолько

сильно завуалировать исходный код, чтобы пользователь не мог получить к нему доступ, но можно скрыть некоторые его основные части, чтобы избежать просмотра его посторонними наблюдателями. В этом трюке рассказано о базовых принципах скрытия JavaScript при помощи автоматического переименования вызываемых JavaScript функций. При помощи написанного ниже кода вы можете разместить на сервере открытый код на JavaScript, а затем скрыть его при доступе к нему браузера.

Код

Сохраните исходный код из примера 3.10 в файл index.php.

Пример 3.10. Сценарий, использующий библиотеку obscure.php

```
<?php
require_once( "obscure.php" );
obscurejs_start( ) ?>
<html>
<head>
<script language="JavaScript">

function dowrite( )
{
  document.write( "This is a test" ):
}
</script>
</head>
<body>
<script language="JavaScript">dowrite( );</script>
</body>
</html>
<?php obscurejs_end( ) ?>
```

Основной работой занимается библиотека obscure.php (пример 3.11).

Пример 3.11. Внесение изменений в названия методов

```
<?pnp
function obscurejs_start( )
{
ob_start( );
} "
$funcs = array( ):
function decreplace( $matches )
{
  global $funcs:
  $newname = "af".count($funcs);
  $funcs[ $matches[1] ] = $newname;
  return "function ".$newname."(";
}

function objscurejs( $matches )
{
  global $funcs;
  $js = $matches[2];
```

```
  $js - preg_replace_callback( "/function\s+(.*?)\s*\(/", "decreplace", $js );
  foreach( $funcs as $oldfunc -> $newfunc )
  {
    $js = preg_replace( "/".$oldfunc."/", $newfunc, $js );
  }
  return "<script".$matches[1].">".$js."</script>";
}

function obscurejs_end( )
{
  $doc = ob_get_clean( );
  $doc = preg_replace_callback( "/\<script(.*?)\>(.*?)\<\/script\>/s",
  "objscurejs", $doc );
  print( $doc );
}
?>
```

Запуск трюка

Скопируйте код на ваш PHP-сервер и откройте в браузере файл index.php. Вместо оригинального кода JavaScript в документе вы увидите следующий код (обратите внимание на выделенные имена методов, отличающиеся от изначальных).

```
<html>
<head>
<script language="JavaScript">

function afOO
{
  document.write( "This is a test" );
}
</script>
</head>
<body>
<script language="JavaScript">afO( );</script>
</body>
</html>
```

Функция JavaScript dowrite()была переименована в afOO, как и все ссылки на нее в HTML. Код, который отвечает за переименование, расположен в сценарии obscure.php. Он выполняется между вызовами функций obscurejs_start() и obscurejs_end(). Выходная информация от PHP перехватывается, затем находятся соответствующие блоки JavaScript и перезаписываются. Конечно, есть некоторые ограничения: код JavaScript, который будет взят из другого файла, не будет изменен, и функции, которые используются до того, как они были определены, не будут правильно переименованы.

ПРИМЕЧАНИЕ

Следует понимать, что такой вариант едва ли подходит для серьезных приложений, но он является отличной отправной точкой для реализации более мощных задумок. Лучшим решением проблемы будет размещение всего вашего JavaScript-кода во внешних библиотеках, а затем защита их от посторонних глаз. Но для внесения небольшой путаницы достаточно уже этого примера.

ТРЮК №19

Создание бинарных часов с помощью DHTML

Используйте комбинацию PHP и DHTML для создания бинарных часов наподобие тех, которые вы можете приобрести на сайте ThinkGeek.com.

Мой сослуживец недавно приобрел битовые часы фирмы ThinkGeek (http://www.thinkgeek.com/). У часов есть три секции с восемью источниками света в каждой. Каждая секция отображает секунды, минуты и часы, а каждому биту соответствует свой собственный источник света. Несмотря на то что ни я, ни он не смогли толком понять, который сейчас час (!!), мы оба пришли к выводу, что часы очень забавные. Итак, вашему вниманию предоставляются такие же суперкрутые, созданные по аналогии «непростые-в-понимании» часы на PHP.

Код

Как показано в примере 3.12, основная работа идет в файле clock.php.

Пример 3.12. Создание массива битовых масок и пр.

```
<?php
$bit_names = arrayC 1. 2. 4. 8. 10, 20 );
$bit_masks = array(
  array( '0'. '0x1'. '1' ) ,
  array( '1'. '0x2'. '2') .
  array( '2'. '0x4'. '4') .
  array( '3'. '0x8'. '8') .
  array( '4'. '0x10'. '16' ).
  array( '5'. '0x20'. '32' )
);

$size - 40:

function bit_table( $name )
{
  global $size;
?>
<table width="<?php echo($size*2); ?>" cellspacing="2" cellpadding="0">
<tr>
<td id="<?php echo( $name ); ?>_1" width="<?php echo($size); ?>" height="<?php
echo($size); ?>" /></td>
<td id="<?php echo( $name ); ?>_2" width="<?php echo($size); ?>" height="<?php
echo($size); ?>" /></td>
</tr>
<tr>
<td id="<?php echo( $name ): ?>_4" width="<?php echo($size); ?>" height="<?php
echo($size); ?>" /></td>
<td id="<?php echo( $name ); ?>_8" width="<?php echo($size); ?>" height="<?php
echo($size); ?>" /></td>
</tr>
<tr>
```

```
<td id="<?php echo( $name ); ?>_10" width="<?php echo($size): ?>" height="<?php
echo($size): ?>" /></td>
<td id="<?php echo( $name ): ?>_20" width="<?php echo($size): ?>" height="<?php
echo($size): ?>" /></td>
</tr>
</table>
<?php
}
?>
<html>
<head>
<script>
var second_bits - [];
var minute_bits = [];
var hour_bits = [];

function startup( )
{
<?php foreach( $bit_names as $name ) { ?>
second_bits.push( document.getElementById( "second_<?php echo( Sname ) ?>" ) );
minute_bits.push( document.getElementById( "minute_<?php echo( Sname ) ?>" ) ):
hour_bits.push( document.getElementById( "hour_<?php echo( Sname ) ?>" ) );
<?php } ?>

set_clock( );

window.setInterval( "set_clock( )". 200 );
}

function set_state( obj. val, on_color )
{
  obj.style.background = val ? on_color : "white":
}

function set_clock( )
{
  var now = new Date( );
  var seconds = now.getSeconds( );
  var minutes = now.getMinutes( );
  var hours - now.getHours( );

<?php foreach( Sbit_masks as $mask ) { ?>
  set_state( second_bits[<?php echo(Smask[0] ): ?>]. ( ( seconds & <?php
  echo(Smask[1] ); ?> ) ==<?php echo($mask[2] ): ?> ). "red" ):
  set_state( minute_bits[<?php echo($mask[0] ); ?>], ( ( minutes & <?php
  echo(Smask[1] ) : ? > ) - - <?php echo($mask[2] ): ?> ). "green" );
  set_state( hour_bits[<?php echo($mask[0] ):?>],(( hours & <?php
  echo($mask[1] ); ? > ) == <?php echo($mask[2] ); ?> ), "blue" );
<?php } ?>
}
</script>
</head>
<body onload="startup( ):">
<table cellpadding="5" cellspacing="0">
<tr><td>
```

```
<?php bit_table( "second" ): ?>
</td><td>
<?php bit_table( "minute" ): ?>
</td><td>
<?php bit_table( "hour" ); ?>
</td></tr></table>
</body>
</html>
```

Для начала в сценарии создается таблица битов, которая будет использоваться для создания страницы и JavaScript, отвечающего за обновление часов. В массиве bit_names хранятся все имена битов в элементах часов, начиная с верхнего левого и заканчивая нижним правым. В массиве bit_masks в первом поле хранится номер бита, во втором — значение битовой маски на JavaScript, а в третьем — значение бита.

Функция bit_table() создает HTML-таблицу с месторасположениями для каждого бита. Она используется трижды: первый раз для секунд, второй раз для минут и третий раз для часов. В части сценария, написанной на PHP, также описывается функция set_clock(), которая переводит время в битовый формат и соответственно изменяет часы. Когда страница готова, она отображается в браузере. Первое, что делает браузер после прорисовки таблиц для секунд, минут и часов, — вызывает функцию startup(), которая инициализирует биты в соответствии с текущим временем. Затем функция startup() вызывает функцию set_clock(), которая получает текущие часы, минуты и секунды и вызывает для каждой из этих составляющих функцию setstateO. Функция же setstateO просто задает фоновый цвет таблицы элементов при помощи CSS. Если бит равен единице, то задается определенный цвет ячейки, иначе цвет остается белым. Функция startup() также создает таймер, который вызывает функцию set_clock() каждые 500 мс, позволяя таким образом часам постоянно обновляться.

Запуск трюка

Загрузите файл clock.php на ваш сервер и откройте его в браузере (рис. 3.14).

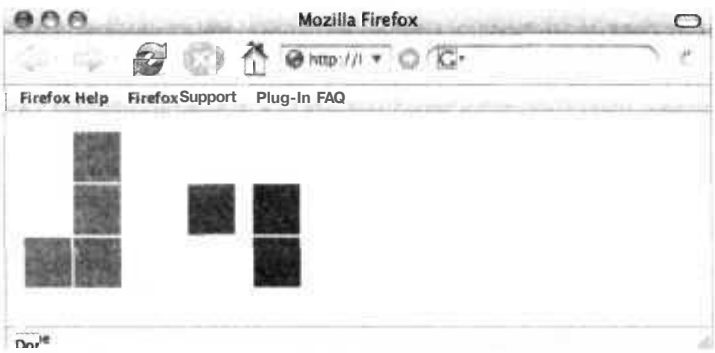


Рис. 3.14. Отображение текущего времени при помощи битовых часов

Битовые часы отображают текущее время в трех секциях по шесть бит на каждую, причем все секции разного цвета (здесь они выглядят черно-белыми). Красными блоками отображаются секунды, зелеными — минуты, а синими — часы. Часы, естественно, выдаются в военном формате (0-23). Этот скриншот был сделан около восьми часов вечера. Но, как я уже сказал, данный трюк предназначен скорее для того, чтобы сделать что-то интересное, чем для того, чтобы выдавать текущее время более эффективно (или даже более наглядно), чем обычные часы.

Смотрите также

- «Создание просмотрщика слайдов при помощи DHTML» (см. трюк 21).
- «Создание интерактивного календаря» (см. трюк 25).
- «Создание эффектов прокрутки при помощи Google Maps» (см. трюк 26).

ТРЮК №20

Приручаем Ajax при помощи JSON

Используйте JSON для упрощения работы с Ajax.

Комбинация запросов DHTML, CSS и XML SOAP или REST с браузера известна как Ajax. Ajax — это отличный способ создавать динамические веб-интерфейсы даже без запроса страницы. Будучи сильно привязанными к JavaScript при запросе HTTP, приложения, поддерживающие Ajax, часто используют DHTML для изменения HTML-страницы, создавая динамические приложения, которые больше похожи на клиентское программное обеспечение. В основном JavaScript связывается со сценарием, написанным на PHP или Perl и размещенным на сервере, а затем интерпретирует возвращаемый текст, который чаще всего является XML.

Анализ XML — это, конечно, не сложная наука, но и не прогулка в парке. Поэтому разработчики создали библиотеку JSON (http://json.org/), при использовании которой можно намного проще интерпретировать ответы от сервера при помощи JavaScript. В этом трюке используется модуль PEAR JSON для формирования информации, идущей от сервера, и показано, как код на JSON распоряжается ею в браузере.

Код

Сохраните код из примера 3.13 в файл getdata.php.

Пример 3.13. PHP отвечает на запрос от Ajax

```
<?php
require( 'JSON.php' );

$records = array( );
$records []= array( 'id' => 1, 'last' => 'Herrington', 'first' => 'Jack' );
```

```
$records [] = array( 'id' => 2, 'last' => 'Herrington', 'first' => 'Megan' );
$records [] = array( 'id' => 3, 'last' => 'Herrington', 'first' => 'Lori' );

$json = new JSON( );
echo( $json->encode( $records ) );
?>
```

Теперь создайте тестовую страницу, которую лучше всего назвать index.php (пример 3.14).

Пример 3.14. Простой пример того, как Ajax делает запрос, а затем обновляет страницу на основе возвращенных данных

```
<html>
<head>
<title>JSON Test</title>
<script>
var req;

function processReqChange( )
{
  if ( req.readyState == 4 && req.status == 200 )
  {
    var rows = eval( req.responseText );
    var html = "<table>";
    for( r in rows )
    {
      html += "<tr>";
      html += "<td>"+rows[r].id+"</td>";
      html += "<td>"+rows[r].first+"</td>";
      html += "<td>"+rows[r].last+"</td>";
      html += "</tr>";
    }
    html += "</table>";
    document.getElementById( "data" ).innerHTML = html;
  }
}

function getNames( )
{
  if (typeof window.ActiveXObject != 'undefined' )
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
    req.onreadystatechange = processReqChange;
  }
  else
  {
    req = new XMLHttpRequest( );
    req.onload = processReqChange;
  }
  try
  {
    req.open( 'GET', 'http://localhost:1222/json/getdata.php', true );
  }
  catch( e )
  {
    alert( e );
```

```
  }
  req.send("");
}
</script>
</head>
<body>
<div id="data">
</div>
<script>
getNames( ):
</script>
</body>
</html>
```

Страница index.php вызывает функцию getNames() — метод JavaScript, который создает объект типа HttpRequest и указывает ему на страницу getdata.php. Сценарий пакует данные и возвращает их, используя модуль JSON. Затем сценарий (снова в index.php) распаковывает эти данные, используя функцию eval О. После этого остается только объединить несколько участков HTML и задать innerHTML в теге .

На рис. 3.15 показана сессия Ajax обмена данными между index.php и getdata.php в описываемом трюке. Браузер, обозначенный здесь как компьютер, запрашивает страницу index.php. Страница index.php затем формирует запрос HTTP к getdata.php, который, в свою очередь, возвращает закодированную при помощи JSON информацию (причем index.php нет необходимости анализировать XML).

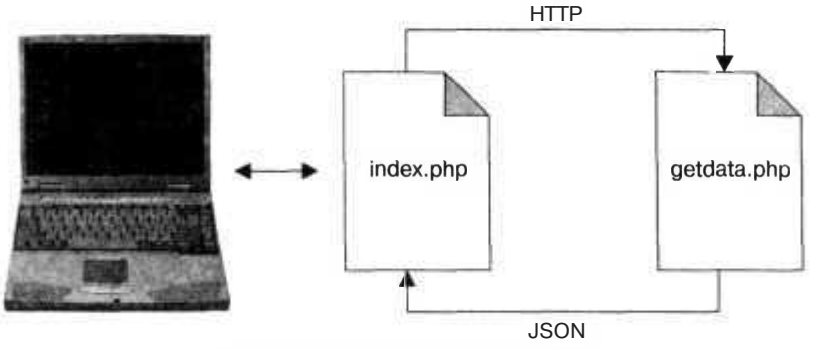


Рис. 3.15. Сессия Ajax на базе JSON

Запуск трюка

Загрузите файлы index.php и getpage.php на сервер. Затем перейдите в браузере к странице index.php. Результат вы можете видеть на рис. 3.16.

JSON действительно позволяет упростить программирование на Ajax с использованием JavaScript, но за это тоже приходится платить. Поскольку теперь веб-сервер вместо XML возвращает код на JSON, другим языкам программирования будет сложнее работать с вашими PHP-сценариями. В идеале страница getdata.php должна возвращать либо XML, либо JSON в зависимости от потребностей клиента.

Рис.3.16. Тестовая JSON-страница

ТРЮК №21

Создание просмотрщика слайдов при помощи DHTML

Используйте библиотеку PHP для работы с графикой и DHTML для создания просмотрщика слайдов в вашем браузере.

Просмотр семейных фотографий — классический камень преткновения в отношениях с родителями. Кому же не нравится рассматривать фотографии своих детей? (И кому не бывает скучно тупо рассматривать фотографии чужих детей?) Но неужели и в наши дни современных высоких технологий люди по-прежнему будут продолжать пользоваться фотоальбомами? А как насчет просмотра изображений в режиме онлайн? В этом трюке показано, как, используя комбинацию PHP 5, библиотеку изображений PHP, немного DHTML и JavaScript, создать в браузере просмотрщик ваших любимых фотографий.

Код

Сохраните исходный код из примера 3.15 в файл `index.php`.

Пример 3.15. Сценарий на PHP, отвечающий за просмотр изображений

```
<?php
$dh - new DirectoryIterator( "pics" );

$files = array( ):
foreach( $dh as $file )
{
  if C preg_match( "/[.]jpg$/", $file ) ) $files []= "$file";
}
?>
```

```
<html>
<head>
<title>Slideshow</title>
<style>
body { background: black: }
#thumbnails { height: 140px: width: 100%; overflow: auto: }
#pic { text-align: center: height: 400px; padding: 20px; }
</style>
<script>
var image_list = [
<?php $first = true; foreach( $files as $image ) { ?>
<?php echo( $first ? "" : ". " ): ?>"<?php echo( $image ): ?>"
<?php $first - false: } ?>
];

var curimage = 0:

function switchimg( ind )
{
  var image - image_list[ind]:
  var obj = document.getElementById( "selimg" ):
  obj.src = "scale.php?image="+image+"&y=400";
  curimage = ind;
}
function nextimage( )
{
  curimage++:
  if ( curimage >= image_list.length ) curimage = 0:
    switchimg( curimage ):
}
window.setInterval( "nextimage( )". 2000 );
</script>
</head>
<body>
<div id="thumbnails">
<table width="100%">
<tr>
<?php $ind - 0; foreach( $files as $image ) { ?>
<td width="160" nowrap align="center">
<a href="javascript:switchimg( <?php echo($ind): ?> )">
<img height="100" src="scale.php?image=<?php echo($image): ?>&y=100" border="0" /
>
</a>
</td>
<?php $ind++; } ?>
</tr>
</table>
</div>
<div id-''pic">
<img id="selimg" height="400" src="scale.php?image=<?php echo($files[0]);
?>&y=400" />
</div>
</body>
```

Теперь создайте файл scale.php, исходный код которого находится в примере 3.16. Этот сценарий будет отвечать за изменение размеров изображений.

Пример 3.16. Сценарий, использующий библиотеку PHP для обработки изображений и предназначенный для масштабирования

```
<?php
$image = $_GET["image"];
$maxy - $_GET["y"];
Sim = @imagecreatefromjpeg( "pics/".$image );
Scurx = imagesx( Sim ):
Scury - imagesy( Sim );
$ratio = $maxy / Scury;
Snewx = Scurx * Sratio:
$newy = Scury * Sratio:
$oim = imagecreatetruecolor( Snewx. Snewy ):
imageantialias( Soim, true );
imagecopyresized( Soim. Sim. 0. 0. 0. 0.
Snewx. Snewy. Scurx. Scury ):
header( "content-type: image/jpeg" ):
imagejpeg( Soim );
?>
```

Основная работа здесь ложится на JavaScript. Функция `switchimg()` изменяет изображение, устанавливая в атрибут src тега `<img>` значение ID переменной `selimg`. Когда срабатывает двухсекундный таймер (этот таймер задается вызовом метода `window. set Interval O)`, вызывается функция `nextimage()`. Страница scale.php — это удобная маленькая страничка, предназначенная для масштабирования изображений. Ее вы можете использовать и во многих других приложениях.

Если вы стремитесь к наибольшей эффективности, возможно, вам захочется, чтобы страница `scale.php` кэшировала изображения с уже измененными размерами, чтобы ей больше не приходилось их пересчитывать при каждом показе. Как вариант вы можете пользоваться версией сценария `scale.php` для командной строки для создания двух вариантов каждого изображения после масштабирования и загружать статические изображения на сервер. В этом случае вам никогда не придется масштабировать изображения на лету.

Запуск трюка

Загрузите оба PHP-сценария в директорию на сервер, создайте поддиректорию под названием **pics** и загрузите несколько ваших любимых фотографий в JPEG-формате. Затем откройте в браузере файл `index.php`. Изображения будут меняться самостоятельно каждые две секунды. Кроме того, вы можете щелкнуть кнопкой мыши на любом изображении вверху страницы, чтобы отобразилась какая-то определенная картинка (рис. 3.17).

ПРИМЕЧАНИЕ

Хорошо, я согласен, что одной из причин, по которой я разработал данный трюк, было желание разместить фотографию моего ребенка в этой книге. Вы ведь не будете осуждать меня за это? Она ведь такая хорошенькая!

Рис. 3.17. Фотографии моей дочки в бассейне

Смотрите также

- «Разбиение страницы на части при помощи разделителей» (см. трюк 15).
- «Создание интерактивного календаря» (см. трюк 25).
- «Создание эффекта прокрутки при помощи Google Maps» (см. трюк 26).

1 ТРЮК №22 Добавление векторной графики при помощи PHP

Работайте с векторной графикой при помощи JavaScript и без каких-либо плагинов.

Вы, наверное, подумаете, что раз HTML позиционируется как язык для создания научных документов, то он должен поддерживать и векторную графику для составления диаграмм. Но, увы, это не так, и для расширения возможностей браузеров по работе с графикой было разработано несколько плагинов. Наиболее известные из них — это Flash и Scalable Vector Graphics (SVG). Оба, как оказалось, получились чересчур тяжеловесными для работы с простыми графиками. Что же остается делать бедному DHTML-программисту?

Один из вариантов — воспользоваться векторно-графической библиотекой Вальтера Зорна JavaScript (http://www.walterzorn.com/). Это всего лишь один файл, содержащий код на JavaScript, но он позволяет вам разместить где угодно на вашей странице векторную графику и даже вносить в нее изменения в браузере в реальном времени. Сценарий создает тысячи небольших элементов DIV, по одному на каждый пиксел графа. Это не лучший выход, конечно, но это работает, и для небольших графиков такой вариант является вполне подходящим с точки зрения производительности.

Код

Сохраните исходный код из примера 3.17 в файл index.php.

Пример 3.17. Использование PHP-библиотеки Zorn JavaScript для создания графиков

```
<?php
$width = 400;
Sheight - 400;

$point_count = 10;

Spoints - array( );

for( Spoint - 0; $point < $point_count; $point++ )
{
  $d = ( 360 / $point_count ) * $point:
  $x = ( $width / 2 ) - ( ( $width / 2 ) * sin( deg2rad( $d ) ) );
  $y = ( Sheight / 2 ) - ( ( $height / 2 ) * cos( deg2rad( $d ) ) );
  Spoints []= array( 'x' -> $x, 'y' => $y );
}
?>
<html>
<head>
<script src="wz_jsgraphics.js"></script>
</head>
<body>
<div id="graph" style="width:<?php echo($width); ?>px;height:<?php echo($height);
?>px;position:relative;">
</div>
<script>
var jg = new jsGraphics( "graph" );
<?php
for( $start = 0; $start < count( Spoints ): $start++ ) {
  $sx = Spoints[Sstart]['x'];
  Ssy = Spoints[Sstart]['y'];
  fort Send = 0; Send < count( Spoints ); Send++ ) {
    Sex = Spoints[Send]['x'];
    $ey = $points[$end]['y']:
    ?>
    jg.drawLine( <?php echo(Ssx); ?>. <?php echo(Ssy); ?>.
    <?php echo(Sex); ?>, <?php echo($ey); ?> ):
    <?php
  }
}
```

```
?>
jg.paint( );
</script>
</body>
</html>
```

Код достаточно простой. Сперва PHP с помощью простых математических вычислений рассчитывает месторасположение десяти точек канвы с размерами 400 x 400 пикселов. Затем HTML создает страницу с канвой, размещенной соответствующим образом в элементе DIV. Для библиотеки необязательно, чтобы канва была размещена в элементе DIV, ведь проводить отрисовку изображений можно прямо в элементе BODY, но я считаю, что так будет более удобно. JavaScript создает объект для рисования. Затем PHP вызывает функцию drawLine() нужное количество раз. И наконец вызывается метод paint O, который выводит графику на канву.

Запуск трюка

Скопируйте файл index. php и сценарий wz_graphi cs.js на сервер и откройте в вашем браузере страницу index.php. Результат вы можете видеть на рис. 3.18. На моем Макинтоше G4 PowerBook на открытие страницы уходит около четырех секунд. Это время понадобилось не для того, чтобы интерпретировать код, а для того, чтобы вывести на экран графику. Очевидно, этот вариант неприемлем для игровых приложений, но для небольших графиков он вполне годится.

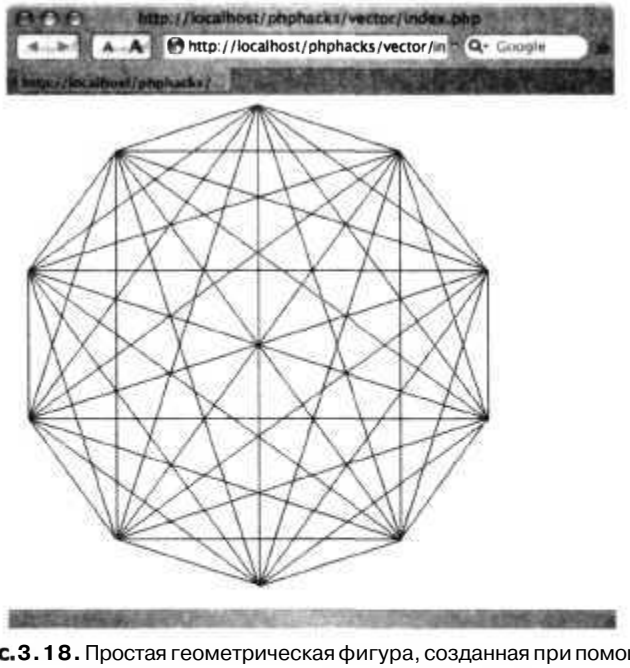


Рис.3.18. Простая геометрическая фигура, созданная при помощи библиотеки для работы с графикой

Смотрите также

- «Создание красивой графики при помощи SVG» (см. трюк 28).
- «Использование объектов для упрощения работы с графикой» (см. трюк 29).
- «Разбиение изображения на составные части» (см. трюк 30).
- «Создание графиков на PHP» (см. трюк 31).

ТРЮК №23

Создание палитры для выбора цвета

Используйте HSB и DHTML для создания палитры на PHP.

Пользователям нравится, когда у них есть возможность изменять цветовую гамму сайта или цвета, которыми отображаются данные в их интернет-приложениях. В этом трюке показан пример палитры на DHTML, которая позволяет пользователям выбирать цвет из HSB-сетки с цветами.

Код

В примере 3.18 находится исходный код файла index.php.

Пример 3.18. Преобразование значений из HSB в RGB

```
<?php
function hsb( $h, $s. $v )
{
  $r = $g = $b = 0;
  if ( $s == 0 )
  {
    $r = $g = $b = $v:
  }
  else
  {
    $h = $h / 60:
    $i = floor( $h );
    $f = $h - $i:
    $p = $v * ( 1 - $s );
    $q = $v * ( 1 - $s * $f ):
    $t = $v * ( 1 - $s * ( 1 - $f ) );
    switch( $i ) {
      case 0: $r = $v; $g = $t: $b = $p; break;
      case 1: $r = $q: $g = $v; $b = $p; break;
      case 2: $r = $p: $g = $v: $b - $t; break;
      case 3: $r = $p; $g = $q; $b = $v; break;
      case 4: $r = $t; $g = $p; $b - $v; break:
      default: $r - $v; $g = $p; $b = $q; break:
    }
  }
  return array( $r. $g, $b ):
```

```
}

function hsb2hex( $h. $s. Sb )
{
  list( $r, $g. $b ) = hsb( $h. $s. $b );
  return sprintf( "#%02x%02x%02x", $r. $g. $b );
}
?>
<html>
<head>
<script language="Javascript">

function mover( id )
{
  var obj - document.getElementById( id );
  obj.style.borderColor - "black":
}

function mout( id )
{
  var obj = document.getElementById( id );
  obj.style.borderColor = "white";
}

function selectColor( color )
{
  document.getElementById( "color" ).value = color;
}

function hover( color )
{
  document.getElementById( "hoverColor" ).innerHTML = color;
}
</script>
<style type="text/css">
body { font-family: arial. verdana. sans-serif; }
#color { font-family: courier: }
#hoverColor { font-family: courier; }
</style>
</head>
<body>
Color: <input id="color" type="text" size="8" />
<table cellspacing="10" cellpadding="0"><tr><td>
<table cellspacing="0" cellpadding="0">
<?php
Sid = 1:
fort Sh = 0: Sh < 360; $h += 18 ) { ?>
<tr>
<?php fort $b = 255; Sb >= 0; Sb -- 10 ) {
Scolor = hsb2hex( Sh, Sb / 255, Sb ):
?>
<td>
<div id="cp<?php echot Sid ); ?>" style="height:10px; width:10px; border:1px
solid white; background:<?php echot Scolor ); ?>;" onmouseover="mover('cp<?php
echot Sid ); ?>');hover('<?php echot Scolor ): ?>');" onmouseout="mout('cp<?php
```

```
echo( $id ): ?>')" onclick="selectColor('<?php echo( $color ); ?>'):"></div>
</td>
<?php
Sid +- 1:
} ?>
</tr>
<?php } ?>
</table>
</td><td valign="top">
<div id="hoverColor"></div>
</td></tr></table>
</body>
</html>
```

Основной здесь является функция HSB(), которая преобразует выбранный цветовой тон, насыщенность и яркость (HSB) в значение RGB. Я нашел данный пример в Интернете на языке C и просто переписал код на PHP. Функция `hsb2hex()` обрабатывает преобразованное HSB-значение и на выходе выдает RGB-значение в формате #RRGGBB.

Запуск трюка

Загрузите файл `index.php` на сервер и откройте его в браузере. Браузер должен будет выглядеть так, как на рис. 3.19. Просто поводите указателем мыши над сеткой, и вы увидите, как справа будет отображаться RGB-значение, над которым в данный момент находится указатель. Щелкнув кнопкой мыши на каком-либо из элементов сетки, вы увидите, что RGB-значение цвета отобразится в текстовом поле вверху страницы.

ПРИМЕЧАНИЕ

Вы можете встроить этот трюк в ваше **интернет-приложение,** добавив элемент для работы с цветом INPUT на вашу **веб-форму.**

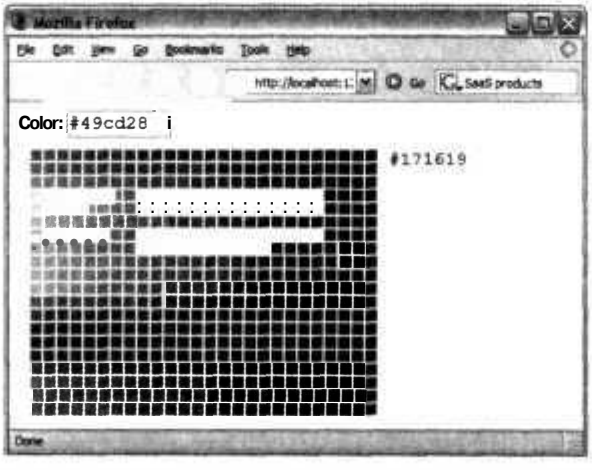


Рис. 3.19. Палитра с возможностью выбора цвета на DHTML

ТРЮК №24

Создание диаграмм ссылок

Изменяйте размер шрифта ваших ссылок, чтобы подчеркнуть важность определенных терминов.

Flikr (http://flikr.com/) — это сайт, который позволяет пользователю загружать изображения, а затем помечать их, используя термин, состоящий из одного слова. Позже вы сможете вновь посетить этот сайт, произвести поиск по базе данных Flikr, используя эти самые термины, и увидеть диаграмму ссылок (см. трюк 91), которая показывает наиболее часто используемые термины гораздо большим шрифтом, чем те, которые используются реже. В этом трюке будет показано, как создавать диаграмму ссылок, проанализировав ключевые слова статьи с новостного сайта Cable News Network (http://cnn.com/). Ведется подсчет слов, и размер шрифта для каждого из слов увеличивается пропорционально частоте их вхождения в статью.

Код

Сохраните код из примера 3.19 в файл `linkgraph.php`.

Пример 3.19. Исходный код для создания диаграммы ссылок

```
<?php
$wordcounts = array( );
$words = splitt " ". "CNN number Americans disapproving President Bush job
perance risen highest level presidency according CNN USA Today Gallup poll
released Monday According poll percent respondents disapproved Bush performance
compared percent approved margerror plus minus percentage points percent
figure highest disapproval rating recorded CNN USA Today Gallup poll Bush
president January approval percentage percent matches low point late March
point gap between those disapproved approved largest recorded during Bush tenure
As Bush prepares address nation Tuesday defend Iraq policy just percent those
responding poll approved handling war percent disapproved Full story approval
rating Iraq unchanged poll late May disapproval figure marked increase
percentage points But poll found issues other Iraq war dragging down Bush numbers
Respondents expressed stronger disapproval handling economy energy policy health
care Social Security lone bright spot president poll handling terrorism which
scored percent approval rating compared just percent disapproved presidents
worst numbers latest poll came issue Social Security respondents disapproving
performance margmore percent percent Bush made changing Social Security system
signature issue second term He proposed creating voluntary government sponsored
personal retirement accounts workers younger Under proposal workers could invest
portion their Social Security taxes range government selected funds exchange
guaranteed benefits retirement plan run instiff opposition Democrats accounts are
too risky undermine Social Security system Some Republicans are wary taking such
politically risky economy only percent poll respondents approved Bush
performance compared percent disapproved On energy policy percent approved
percent disapproved health care percent approved percent disapproved poll results
based interviews Friday Sunday American adults" );
foreach( Swords as Sword )
{
  $word = strtolower( $word );
```

```
  if ( strlen( $word ) > 0 )
  {
    if ( ! array_key_exists( $word, $wordcounts ) )
      $wordcounts[ $word ] = 0;
    $wordcounts[ Sword ] +- 1;
  }
}

$min = 1000000:
$max - -1000000:
foreach( array_keys( Swordcounts ) as Sword )
{
  if ( $wordcounts[ Sword ] > $max )
    $max = Swordcounts[ $word ];
  if ( $wordcounts[ Sword ] < Smin )
    Smin = Swordcounts[ Sword ]:
  }
  $ratio = 18.0 / ( Smax - Smin );
?>
<html>
<head>
<style type="text/css">
body { font-family: arial, verdana, sans-serif; }
.link { line-height: 20pt: }
</style>
</head>
<body>
<div style="width:600px;">
<?php
Swc = array_keys( Swordcounts );
sort( Swc );
foreach( Swc as Sword )
{
  Sfs = (int)( 9 + ( Swordcounts[ Sword ] * Sratio ) );
?>
<a class="link" href="http://en.wikipedia.org/wiki/<?php echo(Sword):
?>" style="font-size:<?php echo( Sfs ); ?>pt;">
<?php echo( Sword ); ?></a>  :
<?php } ?>
</div>
</body>
</html>
```

ПРИМЕЧАНИЕ

Я потратил много времени на создание списка ключевых слов из статьи, вы же можете выбрать их, написав небольшую программу.

Запуск трюка

Загрузите файл на ваш веб-сервер и откройте в браузере страницу `linkgraph.php` (рис. 3.20). Как вы можете видеть, такие термины, как **percent, bush, approved, disapproved, security** и **social,** выделяются из основной массы, так как они были использованы намного чаще. Интересно, что из этого результата вытекает, что эта

статья CNN была о результатах последних опросов, а также о повторной попытке Буша предпринять какие-то действия по обеспечению общественной безопасности. Слово disapproved отображается слегка большим шрифтом, чем остальные слова, что может означать, что либо идет речь о чем-то негативном, либо это просто стиль, в котором была написана эта статья. Несмотря на то что это крайне небольшой пример, видно, что даже по нему можно получить информацию, исходя из различий в размере ссылок диаграммы.

Рис. 3.20. Диаграмма ссылок, созданная в результате анализа статьи

ТРЮК №25

Создание интерактивного календаря

Воспользуйтесь PHP-функцией работы с датами для создания интерактивного календаря на HTML.

Календарь зачастую является лучшим способом для отображения каких-либо значимых событий. Календарь — это то, с чем знаком каждый пользователь, плюс ко всему он является одним из лучших способов отображения большого количества информации, не занимая при этом много места. В Интернете уже доступны элементы управления всплывающими календарями, а также можно найти исходные коды календарей с возможностью добавления в них каких-либо событий, но, насколько я заметил, нет элемента управления, который мог бы отображать какое-либо событие в обычной календарной форме. В этом трюке при помощи HTML создается обычный календарь с возможностью настройки текущей даты и месяца.

Код

Сохраните исходный код из примера 3.20 в файл cal.php.

Пример 3.20. Управление календарем при помощи PHP

```
<html>
<head>
<style type="text/css">
.calendar {
  font-family: arial, verdana, sans serif;
}
.calendar td {
  border: 1px solid #eee;
}
.calendar-title {
  text-align: center;
  font-style: italic;
}
.calendar-day-title {
  text-align: center;
  font-size: small;
  background: #ccc;
  font-weight: bold;
}
.calendar-day, .calendar-outmonth-day {
  height: 60px;
  vertical-align: top;
  text-align: center;
  font-size: small;
  padding: 0px;
}
.calendar-day-number {
  text-align: right;
  background: #ddd;
}
.calendar-content {
  padding: 2px;
  font-size: x-small;
}
.calendar-outmonth-day {
  color: #666;
  font-style: italic;
  background: #ddd;
}
</style>
</head>
<body>
<?php
class Day
{
  function Day( $inmonth, $month, $day, $year )
  {
    $this->{'month'} = $month;
    $this->{'day'} = $day;
```

```
    $this->{'year'} - $year:
    $this->{'inmonth'} = $inmonth:
    $this->{'number'} = $number:
    $this->{'text'} - "";
  }
  function get_day( ) { return $this->{'day'}; }
  function get_month( ) { return $this->{'month'}: }
  function get_year( ) { return $this->{'year'}; }
  function get_inmonth( ) { return $this->{'inmonth'}: }
  function get_number( ) { return $this->{'number'}: }
  function get_text( ) { return $this->{'text'}; }
  function set_text( $text ) { $this->{'text'} - $text; 1
}

function setCalendarText( $days, $m, Sd, $y, Stext )
{
  foreach( Sdays as Sday )
  {
    if ( Sday->get_day( ) — Sd &&
      Sday->get_month( ) == Sm &&
      $day->get_year( ) == Sy )
      $day->set_text( Stext ):
  }
}

function get_last_month( Smonth. Syear )
{
  Slastmonth = Smonth - 1;
  Slastyear - Syear;
  if ( Slastmonth < 0 ) { Slastmonth = 11; Slastyear -= 1; }
  return array( Slastmonth, Slastyear );
}

function get_next_month( Smonth. Syear )
{
  Snextmonth - Smonth + 1;
  Snextyear = Syear;
  if ( Snextmonth > 11 ) { Snextmonth = 0; Snextyear += 1; }
  return array( Snextmonth, Snextyear );
}

function makeCalendarDays( Smonth, Syear )
{
  list( Snextmonth. Snextyear ) = get_next_month( Smonth. Syear );
  list( $lastmonth, Slastyear ) - get_last_month( Smonth, Syear );

  $dimlm = cal_days_in_month( CAL_GREGORIAN, Slastmonth. $lastyear );

  $jd - cal_to_jd( CAL_GREGORIAN, Smonth + 1, 1, Syear );
  Sday = jddayofweek( Sjd );
  $dim = cal_days_in_month( CAL_GREGORIAN, Smonth + 1, Syear );

  $days = array( ):

  for( Sd = 0; Sd < Sday; Sd++ )
```

```
    $days []= new Day( 0. $lastmonth + 1. $dimlm - ( $day - $d ). Slastyear );

  for( $d = 1: Sd <= $dim; $d++ )
    $days []= new Day( 1. Smonth + 1, $d. $year );
  $left = ( ( floor( ( $day + $dim ) / 7 ) + 1 ) * 7 ) - ( $day + $dim );
  for( $d = 0: $d < $left: $d++ )
    $days []= new Day( 0, $nexmonth + 1, $d+1. $nextyear ):

  return $days;
}

$today = getdate( );
$year = $today['year'];
Smonth - Stoday['mon'] - 1:

if ( $_GET['year'] ) $year = $_GET['year'];
if ( $_GET['month'] ) Smonth = $_GET['month'];

Sdays = makeCalendarDays( $month. Syear ):

setCalendarText( &$days, $month + 1. 5. $year. "Meet<br/>Jim" ):
setCalendarText( &$days. Smonth + 1. 10. $year, "Meet<br/>Sue" );

Smonths = array(
  "January". "February", "March". "April".
  "May". "June". "July". "August".
  "September". "October". "November". "December" );
$day_names = array( "Sun". "Mon". "Tue", "Wed". "Thu". "Fri". "Sat" );
?>
<div style="width:600px;">
<table class="calendar" width="100%" cellspacing="0" cellpadding="1">
<tr><td colspan="7" class="calendar-title" width="13%">
<?php
list( $nextmonth, Snextyear ) - get_next_month( $month. Syear );
list( $lastmonth. $lastyear ) = get_last_month( Smonth. Syear );
?>
<a href="cal.php?year=<?php echo(Slastyear); ?>&month=<?php echo( $lastmonth ):
?>">&lt;&lt;</a>
<?php echo( $months[$month] ); ?> <?php echo( Syear ): ?>
<a href="cal.php?year=<?php echo($nextyear): ?>&month=<?php echo( $nextmonth );
?>">&gt;&gt;</a>
</td></tr>
<tr>
<?php foreach( $day_names as Sday ) { ?>
<td class="calendar-day-title"><?php echo( Sday ); ?></td>
<?php } ?>
</tr>
<?php
Sp = 0:
foreach( Sdays as $d ) {
if ( $p == 0 ) echo ( "<tr>" );
$day_style - $d->get_inmonth( ) ? "calendar-day" : "calendar-outmonth-day";
?>
<td class="<?php echo( Sday_style ): ?>" width="13%">
<div class="calendar-day-number">
```

```
<?php echo( $d->get_day( ) ); ?>
</div>
<div class="calendar-content">
<?php echo( $d->get_text( ) ); ?>
</div>
</td>
<?php
$p += 1;
if ( $p -- 7 ) $p = 0;
}
?>
</tr>
</table>
</div>
<body>
</html>
```

На рис. 3.21 показаны методы и члены основного в данном трюке класса Day. Каждый календарь — это объект типа Day с полями, в которых хранятся дата, день недели, значение переменной inmonth, определяющей, принадлежит ли этот объект текущему месяцу, а также дополнительный текст с информацией по данной дате.

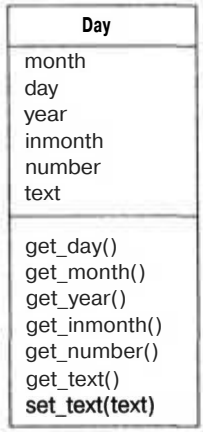


Рис. 3.21. Методы и члены класса Day

Сценарий занимается тем, что создает массив объектов типа Day, используя функцию makeCalendarDaysO, а затем отображает каждый из объектов, используя логику, описанную на PHP и HTML в конце страницы.

Запуск трюка

Загрузите файл cal.php на ваш веб-сервер и откройте его в окне браузера (рис. 3.22).

ПРИМЕЧАНИЕ

Вы можете изменить заданные в календаре события, воспользовавшись функцией setCalendarText(), которая изменяет текст, отображаемый для каждого определенного дня.

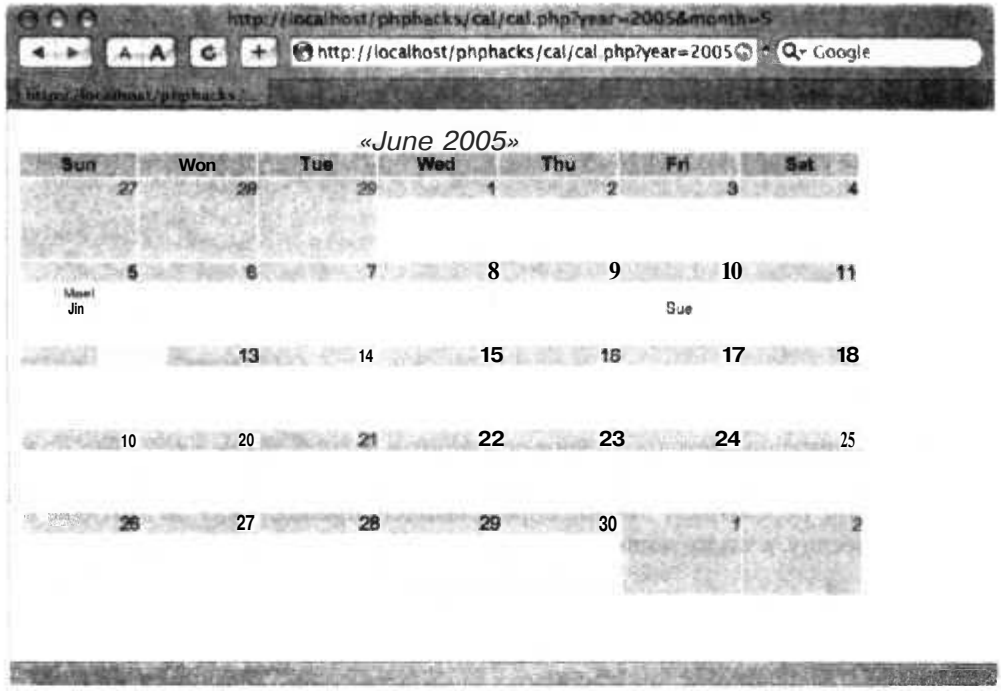


Рис. 3.22. Внешний вид календаря в браузере Safari

Смотрите также

- «Создание динамических графиков на HTML» (см. трюк 14).
- «Добавление векторной графики при помощи PHP» (см. трюк 22).

ТРЮК №26

Создание эффектов прокрутки в стиле Google Maps

Используйте комбинацию маленьких страниц, PHP и JavaScript, чтобы позволить пользователям работать с мышью для прокрутки изображения, которое намного больше, чем можно отобразить на экране.

Когда впервые была выпущена система Google Maps (http://maps.google.com/), ее интерактивные возможности просто поразили пользователей. Сайты для работы с картами, созданные раньше, отображали их с восемью стрелками по краям изображения. Когда вы щелкали кнопкой мыши на стрелке, страница перезагружалась и карта смещалась в направлении, которое вы выбрали. Из-за этого постоянного обновления страницы вам необходимо было снова ориентироваться по вновь загруженной карте.

С приходом Google Maps все изменилось. При помощи Google Maps вы просто щелкаете кнопкой мыши на карте, а затем перемещаетесь по ней в любом необходимом вам направлении (см. трюк 95). Страница никогда не обновляется (разве что только случайно), и вы никогда не собьетесь, перемещаясь по карте.

Я был так впечатлен этим подходом, что решил написать свою собственную версию Google Maps на PHP, JavaScript и воспользоваться очень большим по размеру изображением, созданным в трюке «Разбиение изображения на составные части» (см. трюк 30).

На рис. 3.23 показан внешний вид нашей системы. Слева мы видим саму страницу, а справа — набор картинок, которые составляют одно большое изображение. Карта смещается в пределах видимого прямоугольника, изменяя месторасположение элементов изображения в просматриваемой в данный момент области. Эти изображения берутся из хранилища справа (они доступны постоянно и не требуют обновления страницы).

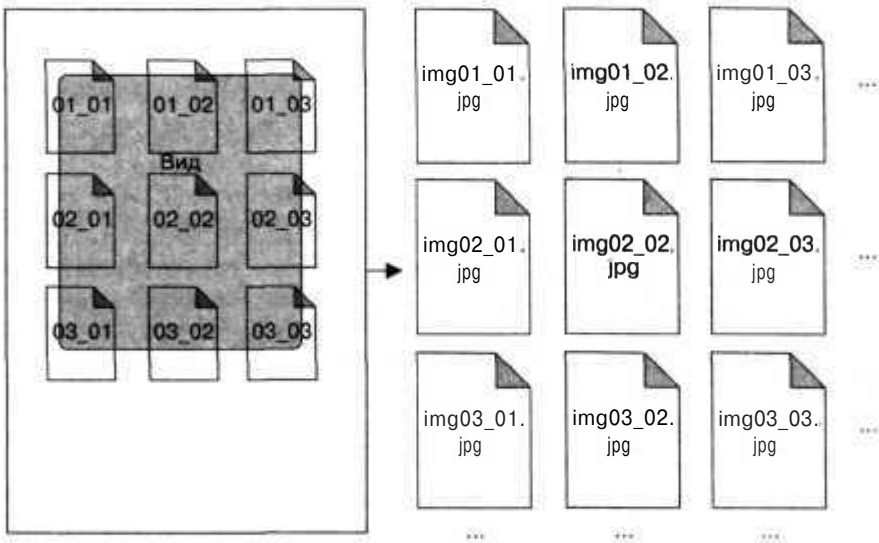


Рис. 3.23. Модель системы для прокрутки изображений

Код

Нужный вам исходный код находится в примере 3.21. Сохраните его в файле `index.php`.

Пример 3.21. Подражаем Google, используя волшебные возможности PHP

```
<?php
$rows - 5;
$cols = 5;
$maxrows = 40;
$maxcols = 40;
```

```
$width - 100;
$height - 100;
?>
<html>
<head>
<script language="Javascript">
var origimgs = [];
<?php
for( $col - 0; $col < $maxcols; $col++ ) {
?>
origimgs[ <?php echo($col); ?> ] = [];
<?php
for( $row = 0; $row < $maxrows; $row++ ) {
$id = sprintf( "img%02d_%02d", $row, $col );
?>
origimgs[ <?php echo($col); ?> ][ <?php echo($row); ?> ] = "<?php echo( $id )
?>.jpg";
<?php
} }
?>

var imgs = [];

function startup( )
{
<?php
for( $col = 0; $col < $cols + 2; $col++ ) {
?>
imgs[<?php echo($col) ?>] - [];
<?php
for( $row = 0; $row < $rows + 2; $row++ ) {
$id = sprintf( "img%02d_%02d", $row, $col ):
?>
imgs[<?php echo($col) ?>][<?php echo($row) ?>] - document.getElementById( "<?php
echo( lid ); ?>" );
<?php
} }
?>
position(0,0);
}

var scroll rows = <?php echo( $rows ); ?>:
var scrollcols = <?php echo( $cols ); ?>:
var width = <?php echo( $width ): ?>;
var height = <?php echo( $height ): ?>;
var maxrows - <?php echo( $maxrows ); ?>;
var maxcols = <?php echo( $maxcols ); ?>:
var xpos = 0;
var ypos = 0;

document.onmousemove - function(e)
{
  if ( dragging )
  {
    xpos += e.pageX - dragx;
```

```
    ypos += e.pageY - dragy;
    if ( xpos < 0 )
      xpos = 0;
    if ( ypos < 0 )
      ypos - 0;
    position( xpos. ypos ):
    dragx - e.pageX;
    dragy = e.pageY:
  }
}

document.onmousedown - function(e)

  dragging = true:
  dragx = e.pageX;
  dragy = e.pageY;
}

document.onmouseup - function(e)
;
  dragging = false;
}

function position( x, y )
(
  if ( x < 0 ) x = 0:
  if ( y < 0 ) y = 0;
  startcol = Math.floor( x / width );
  startrow = Math.floor( y / height );
  offsetx = Math.abs( x - ( startcol * width ) ) * -1;
  offsety = Math.abs( y - ( startrow * height ) ) * -1;
  viewheight - ( scrollrows + 1 ) * height:
  viewwidth = ( scrollcols + 1 ) * width:
  for( var row - 0: row < scrollrows + 2; row++)
  {
    for( var col = 0: col < scrollcols + 2: col++)
    {
      var left - offsetx + ( col * width );
      var top - offsety + ( row * height );
      imgs[row][col].style.left = left:
      imgs[row][col].style.top = top:
      imgs [row] [col].src =origimgs[startrow+row][startcol+col];
      remainderx = viewwidth - ( left + width );
      remaindery = viewheight - ( top + height );
      if ( remainderx > width )
        remainderx = width:
      if ( remainderx < 0 )
        remainderx = 0:
      if ( remaindery > height )
        remaindery = height;
      if ( remaindery < 0 )
        remaindery = 0;
      imgs[row][col].style.clip - "rect( 0px 0px "+remaindery+"px
      "+remainderx+"px )":
    }
  }
```

```
}

var dragging = false;
var dragx = 0;
var dragy - 0;

</script>
</head>
<body onload="startup( );">
<?php
for( $row = 0; $row < $rows + 2; $row++ ) {
  for( $col - 0; $col < $cols + 2; $col++ ) {
    Sid - sprintf( "img%02d_%02d", $row, $col );
?>
<img src="" style="position:absolute;left:0;top:0;" id="<?php echo($id) ?>" />
<?php
} }
?>
</body>
</htral>
```

В первой части сценария при помощи JavaScript создается двумерный массив с изображениями. Он будет использоваться при прокрутке. Все изображения должны быть одинакового размера, их ширина и высота задаются значениями переменных `width` и height в начале сценария. Значения `rows` и `cols` определяют, насколько большой будет область для просмотра, а значения переменных `maxrows` и `maxcols` — размер карты в целом.

Последнее, что делает PHP-сценарий, — размещает код для отображения изображений в области просмотра в теле страницы. Изначально размещение идет с использованием абсолютных значений с точкой отсчета в верхнем левом углу страницы. Когда страница загружена, остальной работой уже занимается браузер. Он вызывает функцию `startup()` при срабатывании события `onload` из элемента BODY. Функция startupO также инициализирует массив и вызывает JavaScript функцию positionO для обновления карты.

Самое интересное происходит именно в функции `position`O. В ней рассчитывается, какие изображения необходимо отрисовать, задаются значения атрибутов src в тегах изображений для соответствующих картинок, их месторасположение на странице, а также обрезается лишнее, чтобы изображение всегда имело форму квадрата.

Функции `onmousedown()`, `onmousemove()` и `onmouseup()` необходимы для отслеживания действий мышью при просмотре страницы. Эти функции, в свою очередь, вызывают функцию positionO для перерисовки изображений.

Запуск трюка

Загрузите исходный код и изображения на сервер с PHP и перейдите в браузере к странице `index.php` (рис. 3.24). Затем щелкните кнопкой мыши на изображении и перетяните его вправо вниз. Изображение должно плавно прокрутиться — без обновления страницы — примерно к той же позиции, что и на рис. 3.25.

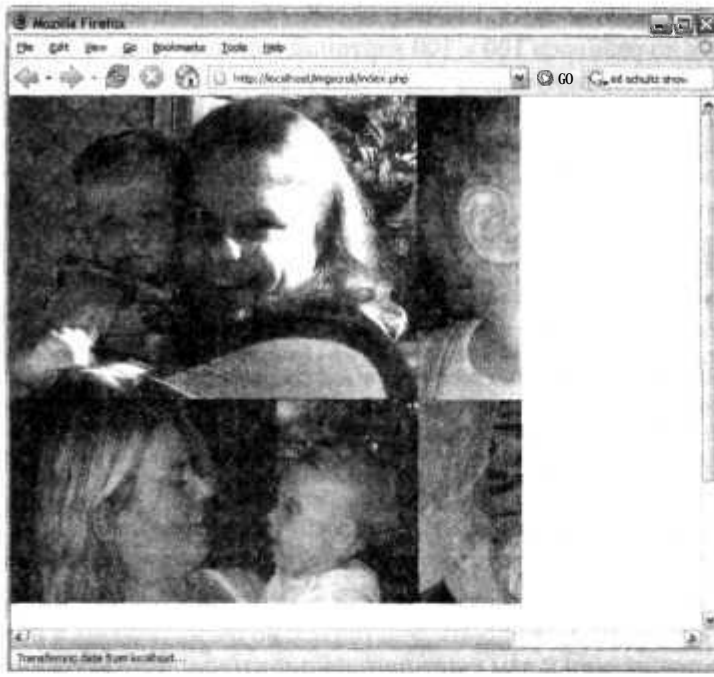


Рис. 3.24. Начальное положение области для прокрутки

Рис. 3.25. Изображение после прокрутки мышью

Помните, что это не одно изображение, которое вы проматываете, а набор из 1600 уменьшенных до размеров 100 x 100 картинок, которые плавно загружаются или удаляются из вашего документа. Такой подход был реализован при помощи сетки размером 7 x 7 из тегов с изображениями. Эти теги размещены на странице с использованием абсолютных координат и при помощи CSS. Когда пользователь проматывает изображение, картинка перемещается, а также отрезается лишнее содержимое. Когда же пользователь выходит за границы данного изображения, путь к картинке меняется на следующий.

ПРИМЕЧАНИЕ

Это сложное приложение с использованием DHTML, так что могут возникнуть проблемы при использовании различных браузеров или даже при работе с другими версиями одних и тех же браузеров. Если вы заглянете в код Google Maps, то увидите, что значительный объем исходного кода занимается исключительно поддержкой работы различных браузеров. Вы должны понимать, что код, предоставленный здесь, не будет корректно работать со всеми браузерами, но, тем не менее, его работоспособность была проверена на браузере Firefox. Если вы планируете использовать что-либо из данного примера в дальнейшем, то вам придется протестировать приложение на совместимость с большим количеством других популярных браузеров.

Разработка похожего приложения, работающего с браузером без необходимости обновления страницы, может быть действительно очень полезна и положительно сказаться на отношении к вам конечного пользователя. Когда страница обновляется, экран становится белым, и после этого медленно загружается новое содержимое. Это может вызывать большое неудобство и приводить к тому, что пользователь отвлечется. Если можно избежать обновления страницы, то пользователь сможет работать непрерывно (а счастливый пользователь — это счастливый программист, ведь так?).

Смотрите также

- «Создание динамических графиков на HTML» (см. трюк 14).
- «Создание бинарных часов на DHTML» (см. трюк 19).

Графика

Трюки 27-33

Как правило, считается, что возможности языка PHP ограничиваются написанием сценариев для HTML. Но это далеко не так: PHP позволяет работать с базами данных, графикой, обрабатывать изображения и многое другое. В данной главе подробно описаны трюки для создания красивых рисунков при помощи растрового отображения, векторной графики и даже Динамического HTML (DHTML). Вы даже узнаете, как можно получить доступ к фотографиям из вашей библиотеки iPhoto и экспортировать их в HTML — и все это при помощи «языка написания сценариев для HTML» под названием PHP.

ТРЮК №27

Реализация предпросмотра изображений

Используйте API GD для работы с графикой в PHP при создании миниатюр ваших изображений.

В этом простом трюке для нескольких изображений в JPEG-формате, расположенных в директории pics, создаются соответствующие миниатюры в директории thumbs. В той же директории создается файл со сценарием `index.html`, содержащий все миниатюры, а также ссылки на исходные файлы изображений.

Код

Сохраните исходный код из примера 4.1 в файл `mkthumbs.php`.

Пример 4.1. Сценарий для работы с миниатюрами изображений

```
<?php
$dir = opendir( "pics" ):
$pics = array( ):
while( $fname = readdir( $dir ) )
{
  if ( preg_match( "/[.]jpg$/", $fname ) )
    $pics [ ] - $fname;
}
closedir( $dir );
```

```
foreach( $pics as Sfname )
{
  Sim = imagecreatefromjpeg( "pics/$fname" );
  Sox - imagesx( Sim );
  Soy - imagesy( Sim ):

  Snx = 100;

  Sny - floor( Soy * ( 100 / Sox ) );

  Snm = imagecreatetruecolor( Snx. Sny );

  imagecopyresized( Snm. Sim, 0. 0. 0, 0. Snx. Sny. Sox. Soy ):

  print "Creating thumb for $fname\n";
  imagejpeg( Snm. "thumbs/$fname" ):
}

print "Creating index.html\n";

ob_start( );
?>
<html>
<head><title>Thumbnails</title></head>
<body>
<table cellspacing="0" cellpadding="2" width="500">
<tr>
<?php
Sindex - 0;
foreach( $pics as Sfname ) {
?>
<td valign="middle" align="center">
<a href="pics/<?php echo( Sfname ); ?>"><img src="thumbs/<?php echo( Sfname ):
?>" border="0" /></a>
</td>
<?php
Sindex += 1;
if ( Sindex % 5 == 0 ) { echo( "</tr><tr>" ); }
}
?>
</tr>
</table>
</body>
</html>
<?php
$html = ob_get_clean( );
Sfh = fopen( "index.html". "w" );
fwrite( Sfh. Shtml );
fclose( Sfh );
?>
```

Для начала сценарий перебирает все изображения в директории pics. Затем при помощи функции GD imagint он создает миниатюры для каждого рисунка и помещает их в директорию thumbs. Для создания миниатюры изображение сперва

должно быть считано и обработано при помощи функции imagecreatefromjpeg(). После этого рассчитывается новый размер миниатюры и при помощи функции imagecreatetruecolor() создается новое изображение. Затем исходное изображение копируется, а его размеры меняются при помощи функции imagecopyresized(). И наконец, функция imagejpeg() сохраняет миниатюры изображений. В оставшейся части сценария при помощи функций буферизованного вывода ob_start() и ob_get_clean() создается HTML-код для этих миниатюр, обе из которых, в свою очередь, предназначены для хранения HTML-кода в строке. После этого с использованием функций fopen(), fwrite() и fclose() такая строка записывается в файл.

Запуск трюка

Разместите сценарий mkthumbs.php в соответствующей директории, а затем создайте еще две поддиректории: pics и thumbs. В директорию pics скопируйте несколько изображений в JPEG-формате. Запустите сценарий при помощи PHP-интерпретатора для командной строки:

```
% php mkthumbs.php
```

Таким образом, вы создадите все необходимые миниатюры изображений и файл index.html, как это показано на рис. 4.1.

Рис. 4.1. HTML-файл для просмотра миниатюр

Сценарий такого рода может быть действительно полезным при создании семейных фотоальбомов. Мне бы хотелось, чтобы как можно больше людей пользовались сценариями для создания миниатюр. Но на самом деле я частенько получаю сообщения от моих друзей или сослуживцев, которые присылают мне ссылки на директории сервера Apache с фотографиями со своих последних поездок. Все без исключения фотографии полноразмерные, причем их владельцы даже

не потрудились убрать размытые или некачественные снимки! К счастью для нас всех, при помощи PHP можно преобразовать всю эту кучу изображений в удобные миниатюры и одновременно с этим иметь возможность получить доступ к оригиналам.

Смотрите также

- «Правильное задание размеров изображений» (см. трюк 9).
- «Разбиение изображения на части» (см. трюк 30).

ТРЮК №28 Создание красивых рисунков при помощи SVG

Используйте формат SVG XML для создания красивых масштабируемых изображений.

Стандарт Scalable Vector Graphics (SVG) XML компании Adobe позволяет PHP работать с графикой в ваших интернет-приложениях на качественно новом уровне. В этом трюке я при помощи простейшего сценария на PHP создам веб-страницу с векторной графикой.

ПРИМЕЧАНИЕ

Важно отметить, что, прежде чем получить возможность просматривать изображение, созданное при помощи SVG, вы должны установить плагин SVG для вашего браузера. Плагины размещены на серверах Adobe по адресу http://www.adobe.com/svg/main.html. SVG— открытый стандарт, поддерживаемый Adobe. На сайте **SVG.org (http://svg.org/)** собираются приверженцы открытого стандарта для поддержки различных браузеров и даже сотовых телефонов.

На рис. 4.2 демонстрируется, как плагин SVG взаимодействует со сценарием `circle_svg.php`, который и генерирует SVG. Объект SVG, внедренный в страницу, запрашивает у сценария XML, а тот возвращает XML со схемой плагина SVG.

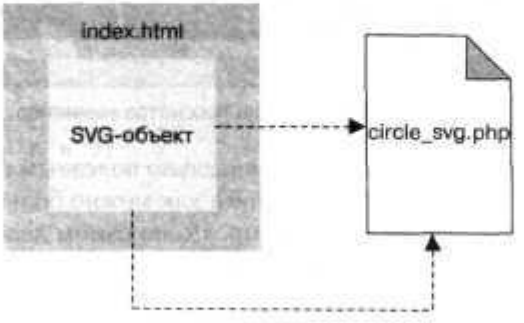


Рис. 4.2. SVG-плагин, запрашивающий SVG из сценария circle_svg.php

Код

Сохраните исходный код из примера 4.2 в файл index.html.

Пример 4.2. HTML-код, демонстрирующий работу SVG

```
<html>
<body>
<embed width="400" height="400" src="circle_svg.php" name="printable"
type="image/svg+xml" />
</body>
</html>
```

Основная работа ложится на сценарий circle_svg.php, исходный код которого находится в примере 4.3.

Пример 4.3. Здесь происходит основная работа SVG

```
<?php

header( "content-type: text/xml" ):

$points_count = 20:

$points - array( ):
for( $p=0: $p<$points_count; $p++ )
{
  $d = ( 360 / $points_count ) * $p:
  $x = 50 + ( cos( deg2rad( $d ) ) * 50 );
  $y - 50 + ( sin( deg2rad( $d ) ) * 50 ):
  $points []= array( 'x' -> $x. 'y' => $y ):
}
echo ("<?xml version=\"1.0\" standalone=\"no\"?>\n" );
?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/SVG/DTD/svg10.dtd">
<svg style="shape-rendering:geometricPrecision;" viewBox="0 0 100 100" xml
space="preserve" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://
www.w3.org/2000/svg" preserveAspectRatio="xMidYMid meet">
<?php
foreach( $points as $start ) {
  $sx = $start['x']:
  $sy - $start['y']:
  foreach( $points as $end ) {
    $ex = $end['x']:
    $ey = $end['y'];
?>
<path fill-rule="nonzero" style="fill:#000000;stroke:#FF0000;stroke-width:0.2"
d="M<?php echo( $sx." ".$sy ): ?> L<?php echo( $ex." ".$ey ): ?> Z"/>
<?php
} }
?>
</svg>
```

Запуск трюка

Перепишите оба файла на ваш сервер и откройте их в браузере, поддерживающем SVG. Я использовал Internet Explorer (рис. 4.3).

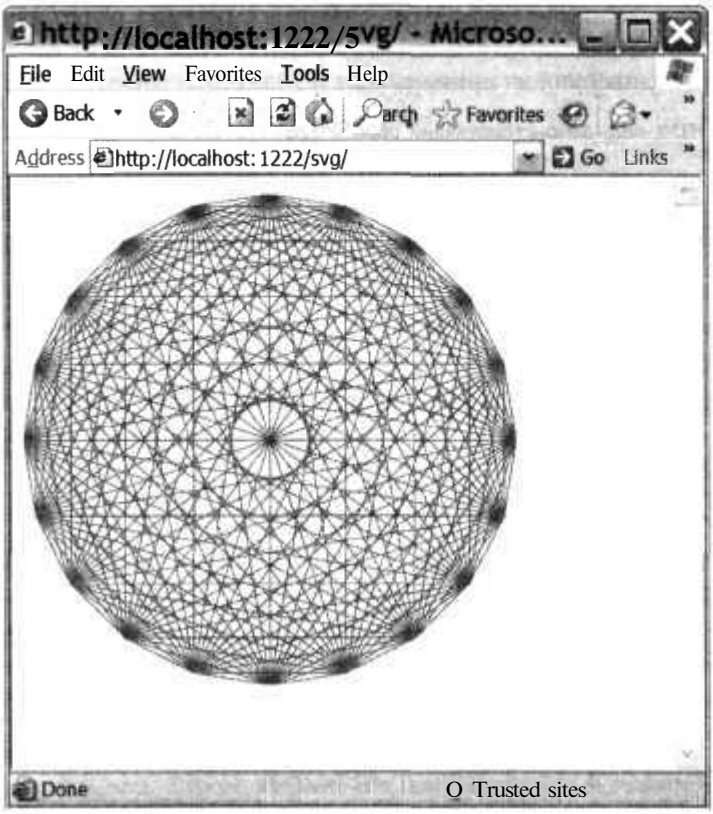


Рис. 4.3. Окружность, нарисованная при помощи SVG

В HTML-страницу встроен SVG-объект, который посылает запрос в сценарий circle_svg.php для получения кода SVG-файла на XML. Эта страница на PHP создает SVG-изображение, которое представляет собой всего лишь набор векторов, соединяющих различные точки окружности. Этот набор векторов очень похож на спирограф. При желании пользователь может щелкнуть кнопкой мыши на изображении и увеличить его, и изображение корректно изменит размеры, так как оно создано при помощи векторной графики.

В SVG есть поддержка большого количества возможностей по работе с графикой и создания спецэффектов. Кроме того, есть поддержка анимации и возможность написания сценариев на JavaScript (здесь нет таких примеров, так как они довольно сложные). На самом деле, если изучить функциональность SVG, то можно сделать вывод, что она находится на уровне технологии Flash 8, однако SVG использует для работы XML, чего не делают уже скомпилированные SWF-файлы. Большим недостатком SVG, конечно же, является небогатый набор компонентов при инсталляции, который ко времени написания данной книги был меньше, чем у Flash. Если вы заинтересовались разработкой Flash-анимации с использованием XML, ознакомьтесь с сайтом Laszlo (http://www.laszlosystems.com/). Здесь вы можете найти компилятор XML с открытым кодом для создания SWF-роликов.

Смотрите также

- «Использование объектов для упрощения работы с графикой» (см. трюк 29).

ТРЮК № 29

Использование объектов для упрощения работы с графикой

Используйте объектно-ориентированные возможности PHP для упрощения создания графики со слоями, вывода ее на экран с применением объектов, а также для удобного масштабирования окна просмотра.

Поддержка графики в PHP находится на очень высоком уровне. Но когда вы пытаетесь создать какое-либо сложное изображение, по некоторым причинам использование PHP становится крайне затруднительным. Во-первых, порядок вывода изображений на экран крайне важен. То, что вы выведете на экран заранее, будет перекрыто изображениями, которые вы отрисуете после этого (а нормального способа снять эти ограничения пока нет). Это значит, что вам придется формировать ваш исходный код исходя из порядка вывода изображений на экран, даже если это сложно реализовать с точки зрения программирования.

Другой проблемой является масштабирование. Чтобы вывести изображение на экран, вам необходимо знать, насколько оно велико и, следовательно, как изменить его размеры. В итоге мы приходим к тому, что нужно обработать множество различной информации при отрисовке содержимого. Добавьте к этому необходимость следить за слоями изображений, и работа с графикой в PHP может стать настоящей головной болью.

К счастью, вы можете решить эти проблемы, даже не будучи профессионалом по работе с фафикой. Для этого нужно лишь воспользоваться готовой графической библиотекой GD для PHP. В этом трюке я создаю простой графический API-объект, в котором порядок отрисовки управляется при помощи z-буфера, а масштабирование — при помощи окна для просмотра.

Код

Сохраните исходный код из примера 4.4 в файл layers.php.

Пример 4.4. Определение нескольких классов, используемых для создания графики со слоями

```
<?php
class GraphicSpace
{
  var $image;
  var $colors;
  var $xoffset;
  var $yoffset;

  var $xscale;
```

```
  var $yscale;

  function GraphicSpace( )
  {
   $this->colors = array( ):
  }

  function get_image( ) { return $this->image; }
  function set_image( $im )
  {
   $this->image = Sim:
  }

  function get_color( Sid ) { return Sthis->colors[ Sid ]; }
  function set_color( Sid. $color ) { $this->colors[ Sid ] = $color: }

  function set_viewport( $left. Stop. $right. Sbottom )
  {
    $this->xoffset = Sleft:
    $this->yoffset = Stop:

    $this->xscale = imagesx( $this->image ) / ( Sright - Sleft );
    Sthis->yscale = imagesy( Sthis->image ) / ( Sbottom - Stop ):
  }

  function transform_x( Sx ) { return ( $x-$this->xoffset ) * Sthis->xscale: }
  function transform_y( Sy ) { return ( $y-$this->yoffset ) * Sthis->yscale; }

  function scale_x( Sx ) { return Sx * $this->xscale: }
  function scale_y( Sy ) { return Sy * Sthis->yscale: }
}

class RenderItem
{
  var Sleft:
  var Sright;
  var Stop;
  var Sbottom;
  var Scolor:
  var $z;

  function RenderItem( Sleft. Stop. Sright, Sbottom. Scolor. Sz )
  {
    Sthis->left = Sleft:
    $this->right = Sright;
    Sthis->top - Stop;
    $this->bottom - Sbottom;
    Sthis->color = Scolor;
    $this->z = $z:
  }

  function get_left( ) { return $this->left; }
  function get_right( ) { return Sthis->right: }
  function get_top( ) { return Sthis->top: }
  function get_bottom( ) { return Sthis->bottom; }
```

```
  function get_z( ) { return $this->z; }

  function render( $gs ) { }
  function transform( $x. $y ) { }
}

class Line extends RenderItem
{
  var $sx;
  var $sy;
  var $ex;
  var $ey;
  var Sthickness;

  function Line( $sx. $sy. Sex. $ey. Scolor, Sz. $thickness )
  {
    $this->RenderItem( min( $sx, $ex ), min( $sy. $ey ),
                       max( $sx, $ex ), max( $sy. $ey ).
                       Scolor. $z 1;
    $this->sx - $sx;
    $this->sy = $sy;
    $this->ex = Sex;
    $this->ey = $ey;
    Sthis->thickness - Sthickness:
  }
  function render( $gs )
  {
    if ( Sthis->thickness > 1 )
      imagesetthickness( Sgs->get_image( ), Sthis->thickness ):
    $this->drawline( $gs->get_image( ).
    $gs->transform_x( $this->sx ).
    Sgs->transform_y( Sthis->sy ).
    $gs->transform_x( Sthis->ex ).
    Sgs->transform_y( Sthis->ey ).
    $gs->get_color( $this->color ) ):
    if ( $this->thickness > 1 )
      imagesetthickness( $gs->get_image( ). 1 );
  }
  function drawline( Sim, $sx. $sy. Sex. Sey. Scolor )
  {
    imageline( Sim. Ssx. Ssy, Sex. Sey, Scolor );
  }
}

class DashedLine extends Line
{
  function drawline( Sim, Ssx. Ssy. Sex. Sey, Scolor )
  {
    imagedashedline( Sim. Ssx, Ssy. Sex. Sey. Scolor );
  }
}

class Ball extends Renderltem
{
  var $text;
```

```
  function Ball( $x. $y. $size, $color. $text. $z )
  {
    $width = $size / 2;
    if ( $text )
      $width += 20;
    $this->RenderItem( $x, $y.
    $x + $width, $y + ( $size / 2 ).
    $color. $z );
    $this->text = $text;
    $this->size = $size:
  }
  function render( $gs )
  {
    imagefilledellipse( $gs->get_image( ),
    $gs->transform_x( $this->left ).
    $gs->transform_y( $this->top ),
    $gs->scale_x( $this->size ).
    $gs->scale_x( $this->size ).
    $gs->get_color( $this->color ) );
    if ( strlen( $this->text ) )
      imagestring($gs->get_image( ). 0.
    $gs->transform_x( $this->left ) + 7.
    $gs->transform_y( $this->top )-5. $this->text.
    $gs->get_color( $this->color ) ):
  }
}

function zsort( $a. $b )
{
  if ( $a->get_z( ) == $b->get_z( ) )
    return 0;
  return ( $a->get_z( ) > $b->get_z( ) ) ? 1 : -1:
}

class RenderQueue
{
  var $items:
  function RenderQueue( ) { $this->items = array( ); }
  function add( $item ) { $this->items [] - $item; }
  function render( $gs )
  {
    usort( &$this->items. "zsort" );
    foreach( $this->items as $item ) { $item->render( $gs ): }
  }
  function get_size( )
  {
    $minx = 1000; $maxx = -1000:
    $miny = 1000; $maxy = -1000;
    foreach( $this->items as $item )
    {
      if ( $item->get_left( ) < $minx )
        $minx » $item->get_left( ):
      if ( $item->get_right( ) > $maxx )
        $maxx = $item->get_right( );
      if ( $item->get_top( ) < $miny )
        $miny = $item->get_top( );
      if ( $item->get_bottom( ) > $maxy )
```

```
        $maxy = $item->get_bottom( );
    }
  return array( left => $minx, top => $miny, right -> $maxx, bottom => $maxy );
  }
}

$width = 400;
$height = 400;

function calcpoint( $d, $r )
{
  $x = cos( deg2rad( $d ) ) * $r;
  Sy = s1n( deg2rad( $d ) ) * $r;
  return array( $x, $y ):
}

$render_queue = new RenderQueue( );

Sox - null;
Soy = null;

for( $d = 0; $d < 380: $d += 10 )
{
  list( $x, Sy ) = calcpoint( $d, 10 ):

  $render_queue->add( new Ball( $x, $y, 1, "line", "", 10 ) ):
  Srender_queue->add( new Line( 0, 0, Sx, Sy, "red", 1, 1 ) );

  if ( Sox != null && Soy !- null )
  {
    $render_queue->add( new Line( Sox, Soy, Sx, $y, "red", 1, 1 ) );
  }
  Sox = Sx;
  Soy = $y;
}

$gsize = $render_queue->get_size( );

Sfudgex = ( $gsize['right'] - $gsize['left'] ) * 0.1:
$gsize['left'] -= Sfudgex;
$gsize['right'] += Sfudgex;
$fudgey - ( $gsize['bottom'] - $gsize['top'] ) * 0.1;
Sgsize['top'] -= Sfudgey:
$gsize['bottom'] += Sfudgey;

print_r( Sgsize );

Sim - imagecreatetruecolor( Swidth, Sheight );
imageantialias( Sim, true );
Sbg - imagecolorallocate($im, 255, 255, 255);
imagefilledrectangle( $im, 0, 0, Swidth, Sheight, Sbg );

Sgs = new graphicspace( ):
$gs->set_image( Sim ):
$gs->set_color( 'back', Sbg );
$gs->set_color( 'line', imagecolorallocate($im, 96, 96, 96) );
$gs->set_color( 'red', imagecolorallocate($im, 255, 0, 0) );
```

```
$gs->set_viewport( $gsize['left'], $gsize['top'], $gsize['right'],
$gsize['bottom'] );

$render_queue->render( $gs );

imagepng( $im, "test.png" );
imagedestroy( $im );
?>
```

На рис. 4.4 показана структура классов в данном сценарии. `RenderQueue` использует в своей работе два объекта: массив элементов RenderItems и объект `GraphicSpace`, в котором хранится изображение и информация об изменениях в нем. Производный от RenderItem класс создает различные типы фигур: линии, шары и прерывистые линии. Для добавления дополнительных фигур просто создайте еще несколько производных классов от класса RenderItem.

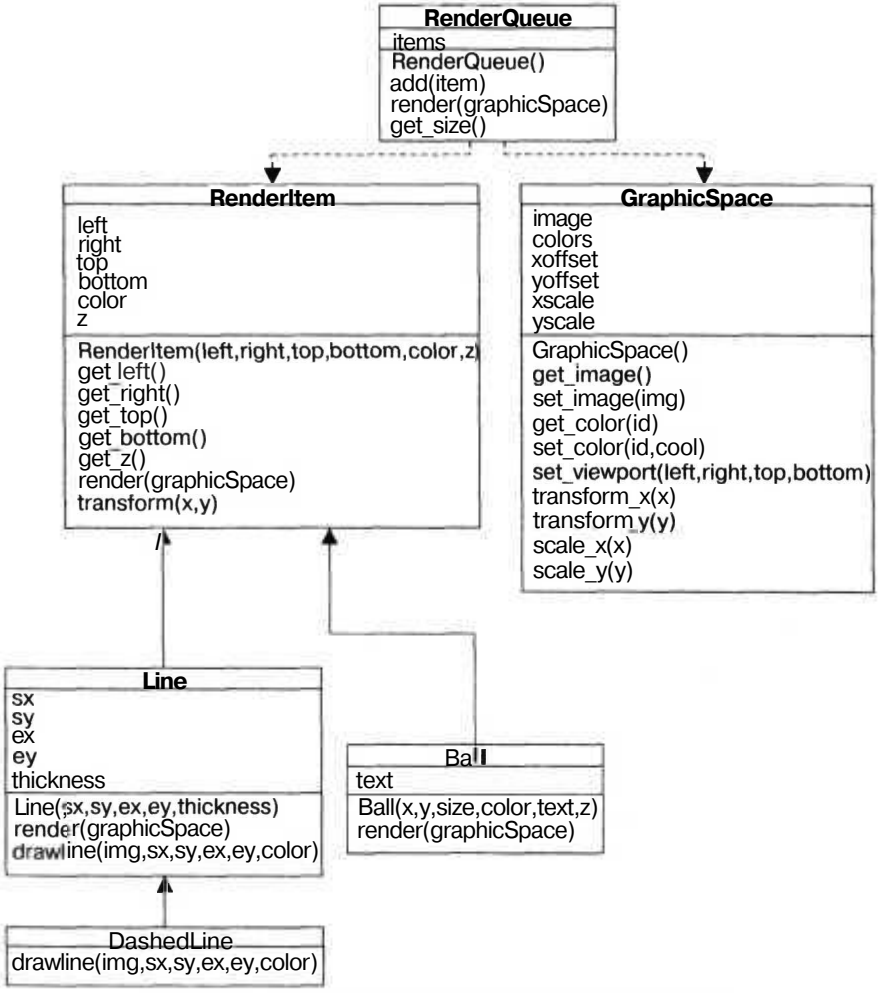


Рис. 4.4. Структура графического объекта на языке UML

У каждого элемента типа RenderItem есть соответствующий уровень z. Элемент с более низким уровнем z (или значением переменной z) будет расположен за другими элементами с более высоким уровнем z. Это позволяет создавать объекты в любой последовательности, которая вам нравится, а затем задавать им различные значения поля z и быть уверенным в том, что они будут отсортированы и выведены на экран в нужной вам последовательности.

Для использования системы координат я использовал viewport (окно для просмотра). Viewport — виртуальная область для вывода графики. Система координат может быть любой: от 0 до 1 или от 0 до 1 миллиарда. Система автоматически масштабирует изображение, подгоняя его под заданные размеры.

Взяв за основу простые объекты API, вы можете создать намного более сложные классы для работы с графикой, причем все это намного проще, чем использование стандартного API для работы с графикой в PHP. Исходный код становится намного более простым в понимании и доработке.

Запуск трюка

Воспользуйтесь PHP-интерпретатором для командной строки, чтобы запустить сценарий layers.php.

```
% php layers.php
Array
(
  [left] => -12.05
  [top] -> -12.05
  [right] => 12.55
  [bottom] => 12.55
)
```

Затем, используя браузер, посмотрите на полученный в результате файл test.png (рис. 4.5).

Объект состоит из элементов трех типов: линий, которые идут из центра к шарам, серых шаров и линий, соединяющих шары по периметру. Линии, идущие из центра, находятся на уровне z, равном 1. Шары — на уровне z, равном 10. И наконец, соединяющие линии — на уровне z, равном 20. Для изменения z-уровня внешних линий вы можете просто изменить значение переменной z с 20, например, на 1.

```
if ( $ox != null && $oy != null )
{
  $render_queue->add( new Line( Sox, $oy. $x, $y, "red". 1.1));
}
```

Теперь, если вы перезапустите сценарий и посмотрите на полученный результат в браузере, то увидите, что шары нарисованы поверх линий (рис. 4.6), в то время как ранее они располагались под ними.

Теперь линии, расположенные по периметру, прикрыты шарами, так как их z-уровень ниже.

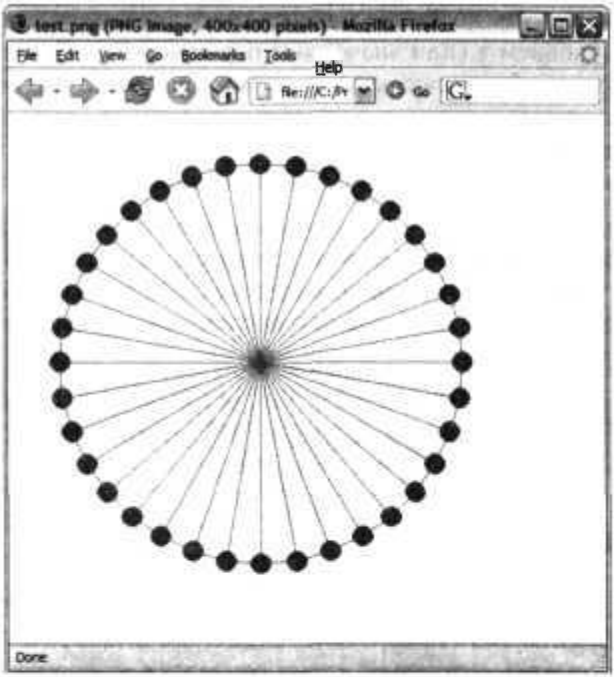


Рис. 4.5. Окружность, созданная с применением графических объектов

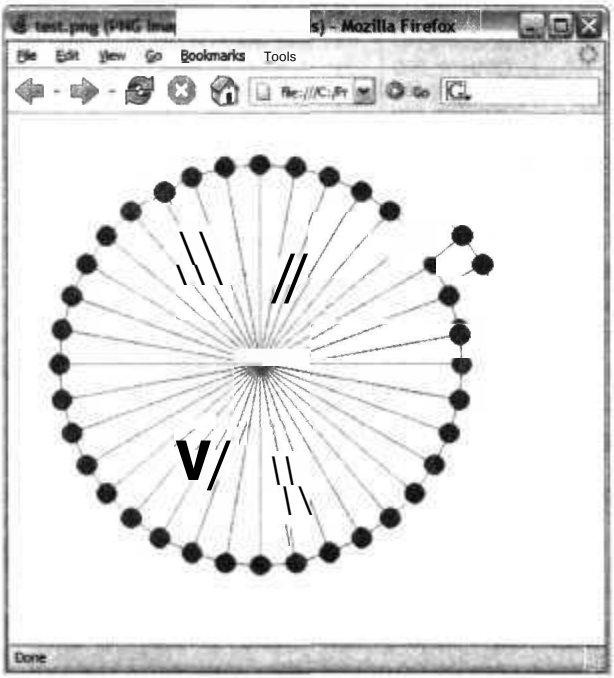


Рис. 4.6. Скрываем соединительные узлы линий за серыми шарами

Смотрите также

- «Создание красивых рисунков при помощи SVG» (см. трюк 28).
- «Создание блочных интерфейсов» (см. трюк 51).

ТРЮК № 30

Разбиение изображения на составные части

Используйте возможности PHP для работы с графикой, чтобы разбивать большие рисунки на множество более мелких.

Иногда намного удобнее иметь группу более мелких изображений, которые составляют одно большое, чем использовать цельный громоздкий рисунок. Примером может служить трюк «Создание эффектов прокрутки в стиле Google Maps» (см. трюк 26), в котором прокрутка идет с использованием большого количества более компактных изображений, очень плавно смещающихся. В результате складывается впечатление, что вы работаете с одним большим изображением. Чтобы иметь возможность воспользоваться вышеуказанным трюком, вам сперва необходимо разбить большое изображение на части, что и делается в описанном ниже трюке.

Код

Сохраните исходный код из примера 4.5 в файл `imgsplit.php`.

Пример 4.5. Разбиение изображения на части

```
<?php
$width - 100;
$height = 100;

$source = @imagecreatefromjpeg( "source.jpg" );
$source_width = imagesx( $source );
$source_height = imagesy( $source );

for( $col = 0; $col < $source_width / $width; $col++)
{
  for( $row - 0; $row < $source_height / Sheight; $row++)
  {
    $fn = sprintf( "img%02d_%02d.jpg", $col, $row );
    echo( "$fn\n" );
    Sim = @imagecreatetruecolor( $width, Sheight );
    imagecopyresized( Sim, $source, 0, 0,
  • $col * $width, $row * Sheight, Swidth, Sheight,
    $width, Sheight );
    imagejpeg( Sim, $fn );
    imagedestroy( Sim );
  }
}
?>
```

Это код, обратный исходному коду из примера 4.6, представленного далее. Он создает из одного большого изображения большое количество более мелких и размещает их по сетке. Данный трюк очень полезен для создания эффектов прокрутки в стиле Google Maps («Создание эффектов прокрутки в стиле Google Maps» (см. трюк 26)). Константы в начале файла определяют, насколько большими должны быть изображения, на которые будет разбит основной рисунок. В сценарии считывается исходное изображение и выясняются его размеры. Затем при помощи нескольких вложенных циклов for перебираются все элементы из сетки и копированием отдельных частей из исходного рисунка создаются изображения, а затем они сохраняются.

Запуск данного исходного кода при помощи PHP-интерпретатора для командной строки выглядит следующим образом:

```
% php imgsplit.php
img00_00.jpg
img01_00.jpg
...
```

В сценарии идет поиск файла под названием source.jpg. Файл разбивается на несколько файлов с названиями img<col>_<row>.jpg, где элементы col и row дополняются нулями. Таким образом, файл с изображением, расположенным в колонке с номером 0 и в строке с таким же номером, будет назван img00_00.jpg. Каждое созданное изображение имеет размеры 100 x 100 пикселов. Эти значения задаются в переменных $width и $height в самом начале сценария.

Улучшение трюка

Что же насчет слияния изображений? В написанном ниже сценарии при использовании нескольких мелких изображений создается одно большое. Вы можете воспользоваться данным исходным кодом как основой при создании средства для просмотра больших изображений с возможностью прокрутки, как это показано в трюке «Создание эффектов прокрутки в стиле Google Maps» (см. трюк 26). Кроме того, вы можете создать просто коллаж из нескольких изображений, который можно использовать в качестве фонового рисунка либо заставки для Рабоч его стола.

Сохраните исходный код из примера 4.6 в файл imgmerge.php.

Пример 4.6. Слияние изображений — еще одна задача, с которой с легкостью справляется PHP

```
<?php
$targetsize_x = 4000;
$targetsize_y = 4000;
$outfile = "merged.jpg";
$quality = 100;

$im = @imagecreatetruecolor( $targetsize_x, $targetsize_y );

$sources = array( );
$dh = opendir( "." );
```

```
while ((Sfile = readdir($dh)) !== false)
{
  if ( preg_match( "/[.]jpg$/", $file ) &&
    $file !- Soutfile )
  {
    Ssources [ ] - imagecreatefromjpeg( $file );
  ]
}

$x = 0:
$y = 0;
$index = 0;
while( true )
{
  $width = imagesx( $sources[ $index ] ):
  $height - imagesy( $sources[ $index ] ):
  imagecopy( $im, $sources[ $index ],
  $x. $y. 0. 0. $width. Sheight ):
  $x += $width:
  if ( $x >= $targetsize_x )
  {
    $x - 0:
    $y +- Sheight;
    if ( $y >- $targetsize_y )
      break;
  }
  $index += 1;
  if ( Sindex >= count( Ssources ) )
    Sindex - 0;
}
imagejpeg( Sim. Soutfile. $quality );
imagedestroy( Sim );
?>
```

В действительности это очень простой сценарий, который позволяет взять изображения из директории и сохранить их в массив ресурсов. Затем создается огромное изображение, в которое копируется исходные рисунки. В цикле `while` собирается большое изображение, при этом перебираются и вставляются в строки и столбцы все составляющие его более мелкие рисунки. В конце сценария созданное ранее большое изображение сохраняется в JPEG-формате. При этом используется функция `imagejpeg()`. Все это происходит при запуске сценария из директории с JPEG-файлами.

ПРИМЕЧАНИЕ

Входные файлы должны иметь расширение **JPG,** а также должны быть одинакового размера. Иначе в составном изображении будет много пустых мест (если приложение к тому времени просто не вылетит с ошибкой).

Пример готового изображения можно увидеть на рис. 4.7.

Сценарий запускается следующим образом:

```
% php imgmerge.php
```

Рис. 4.7. Одно из исходных изображений

Рис. 4.8. Полученное сложное изображение открыто и уменьшено в браузере Firefox

Выходной файл создается в той же папке, в которой расположены исходные файлы. Он называется merged.jpg. Я создал сложное изображение, состоящее из нескольких фотографий моей жены Лори и дочки Меган, увидеть которое можно на рис. 4.8.

Получилось даже лучше — по счастливой случайности Firefox здесь показал себя молодцом. Он изменил масштаб изображения, чтобы оно поместилось в окне браузера. Если вы подведете указатель мыши к изображению, то увидите, что он изменился на увеличительное стекло, и вы можете увеличить составное изображение при масштабе 100 %.

Вы можете подкорректировать качество изображения, полученного при слиянии, изменив значение переменной $quality. Ее значение колеблется в пределах от 0 до 100, где 100 означает наилучшее качество. Переменные $targetsizex и $targetsize_y определяют желаемый размер изображения, получаемого в результате слияния, а в переменной $outfile задается имя выходного файла.

Смотрите также

- «Создание эффектов прокрутки в стиле Google Maps» (см. трюк 26).

ТРЮК №31

Создание графиков на PHP

Используйте набор инструментов PHP для работы с изображениями при создании динамических графиков на основе ваших данных.

PHP обладает отличными возможностями для динамической работы с изображениями. Вы можете пользоваться ими для создания слоев (см. трюк 32) или для создания полноценных новых изображений на лету. В этом трюке используется набор возможностей для создания нескольких простейших научных графиков в форме синусоидальной волны (в качестве подтверждения того, что PHP подходит не только для решения задач, связанных с обработкой изображений, но и для решения некоторых математических задач).

Код

Сохраните исходный код из примера 4.7 в файл graph.php.

Пример 4.7. Графическое отображение математической функции

```
<?
$width - 400;
$height = 300;

$data = array( );
for( $i = 0; $i < 500; $i++ )
{
  $data [ ] - sin( deg2rad( ( $i / 500 ) * 360 ) );
```

```
}
Sxstart - $width/10;
Systart - $height - ($height/10);

Simage - imagecreate($width , $height);
$back = imagecolorallocate($image, 255, 255, 255);
Sborder - imagecolorallocate($image, 64, 64, 64);

imageline( Simage, Sxstart, 0, Sxstart, Systart, Sborder );
imageline( Simage, Sxstart, Systart, Swidth, Systart, Sborder );

imagestringC Simage, 2, Sxstart-20, $ystart-10, "1", Sborder );
imagestringC Simage, 2, Sxstart-20, 0, "-1", Sborder );
imagestringC Simage, 2, Sxstart, $ystart+5, "0", Sborder );
imagestringC Simage, 2, Swidth-20, Systart+5, "360", Sborder );

Sdatatop = 1;
Sdatabottom - -1;

Soldx = 0;
Soldy = 0;
Sdatacount = count( Sdata );
Sxscale = ( Swidth - Sxstart ) / Sdatacount;
Syscale - Systart / ( Sdatatop - Sdatabottom );
Smidline = Systart / 2;
fort Si - 0; Si < Sdatacount; Si++ )
{
  Sx - Sxstart + ( Si * Sxscale );
  Sy - Smidline - ( Sdata[Si] * Syscale );
  if ( Si > 0 )
  {
    imageline( Simage, Soldx, Soldy, Sx, Sy, Sborder );
  }
  Soldx = Sx;
  Soldy = Sy;
}
header("Content-type: image/png");
imagepng($image);
imagedestroy($image);
?>
```

Сперва в сценарии задаются некоторые константы, которые определяют размер выходного изображения. Затем создается новое изображение и назначаются цвета. После этого при помощи функции `imagestring()` рисуется граница графика, а также координатная ось с нанесенными на нее значениями. После этого в сценарии в цикле for, предназначенном для перебора каждой из точек, соответствующих определенным значениям, в графической форме отображается математическая информация, а также при помощи функции `imageline()` соединяются точки. Поскольку предполагается, что сценарий будет использоваться в Интернете, то в качестве типа содержимого (поле `content-type`) необходимо задать `image/png`, чтобы указать браузеру, что на экран выводится изображение в PNG-формате. Многие браузеры автоматически смогут определить тип содержимого, но все-таки будет лучше корректно задать его значение вручную. Когда все эти действия выполнены, изображение выводится на экран при помощи функции `imagepng()`.

Лучше всего задать значение поля content-type в самом конце сценария, так как в этом случае, если выполнение прервется из-за ошибки, вы сможете увидеть информацию о ней в окне браузера. Если же вы зададите значение заголовка слишком рано, то браузер сочтет, что сообщение об ошибке от PHP на самом деле является файлом в PNG-формате, и попытается отобразить его.

Запуск трюка

Перепишите файлы с исходными кодами на ваш PHP-сервер и откройте в браузере страницу graph.php (рис. 4.9).

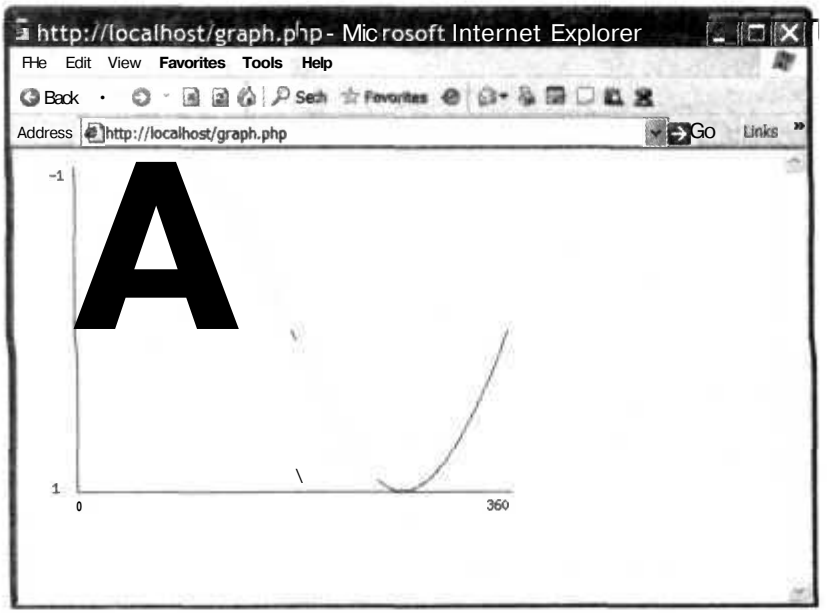


Рис. 4.9. Полученный в итоге график

Если вы не увидите фафика, изображенного на рис. 4.9, то, возможно, возникли проблемы с настройками сервера. Есть множество различных параметров при установке PHP, и для его корректной работы (как правило) не обязательно устанавливать библиотеку с изображениями. Однако, если не установлены библиотеки для работы с изображениями, вы не сможете создать файл с изображением в PNG-формате (будет выдано сообщение об ошибке).

Смотрите также

- «Создание фафиков на HTML» (см. трюк 8).
- «Создание динамических фафиков на HTML» (см. трюк 14).
- «Создание красивых рисунков при помощи SVG» (см. трюк 28).

ТРЮК №32

Наложение изображений друг на друга

Используйте возможности PHP для работы с графикой и создания из нескольких рисунков одного составного изображения.

Одним из способов обработки изображений является размещение рисунков поверх каких-либо базовых изображений, расположение которых высчитывается исходя из каких-то данных. В этом трюке в качестве базового изображения выступает карта, которую вы можете видеть на рис. 4.10.

Рис.4.10. Карта

Затем в трюке поверх карты над городом Сан-Франциско размещается изображение в виде звезды (рис. 4.11), как если бы вы искали какой-либо определенный город по названию или почтовому индексу.

Рис. 4.11. Звезда

Код

Сохраните довольно простой исходный код из примера 4.8 в файл `graphic.php`.

Пример 4.8. Создание наложенных друг на друга изображений в PHP — довольно простая задача

```
<?php
$map - imagecreatefrompng("map.png");
```

```
$star = imagecreatefromgif("star.gif");
imagecopy( $map, $star, 5, 180. 0. 0. imagesx( $star ), imagesy( $star ) );
header("Content-type: image/png");
imagepng($map);
```

Для начала профамма считывает карту и изображение звезды. Затем она создает новое изображение путем размещения звезды на карте, для чего используется функция `imagecopy()`. После этого новая версия карты, которая к данному моменту пока существует только в памяти, отображается в браузере при помощи функции `imagepng()`.

Запуск трюка

После того как вы загрузите PHP-сценарий и изображения на ваш сервер, откройте в браузере файл `graphic.php` (рис. 4.12).

Рис. 4.12. Звезда, изображенная поверх карты

Улучшаем трюк

Звезда на карте — это, конечно, круто, но она слегка великовата. Вместо того чтобы отобразиться лишь поверх Сан-Франциско, она занимает чуть ли не всю Калифорнию. Давайте ее слегка уменьшим. Сохраните исходный код из примера 4.9 в файл graphic2.php.

Пример 4.9. Слегка уменьшаем изображение

```
<?
$map = imagecreatefrompng("map.png");
$star = imagecreatefromgif("star.gif");
imagecopyresized( $map, $star, 25. 205. 0, 0.
imagesxt  $star )/5,  imagesy( $star )/5,
imagesxt  $star ).  imagesy( $star )  );
header("Content-type: image/png");
imagepng($map):
?>
```

Перейдите в браузере к вновь созданному сценарию, и вы увидите изображение, показанное на рис. 4.13.

Рис. 4.13. То же изображение с размещенной поверх карты слегка уменьшенной звездой

В этой новой версии сценария используется функция imagecopyresizecK), предназначенная для изменения размеров звезды перед копированием ее на карту. Сценарий уменьшает ширину и высоту изображения в пять раз, уменьшая таким образом звезду до 20 % от ее исходного размера.

ПРИМЕЧАНИЕ

Изображение звезды растровое, поэтому если вы захотите увеличить ее размер, то увидите, что у нее появятся рваные края.

Смотрите также

- ❑ «Создание предпросмотра изображений» (см. трюк 27).
- • «Получение доступа к фотографиям из iPhoto при помощи PHP» (см. трюк 33).
- • «Разбиение изображения на части» (см. трюк 30).

ТРЮК №33

Получение доступа к фотографиям из iPhoto при помощи PHP

Используйте возможности PHP для работы с XML, чтобы получить доступ к базе данных с фотографиями iPhoto.

Apple — компания, известная своими инновациями и созданием простой в использования продукции. Следуя этим же курсом, она недавно выпустила в свет проект iLife (http://www.apple.com/ilife/), с помощью которого можно оченъ просто создавать и упорядочивать большие объемы мультимедийных данных. Тем не менее я был слегка обескуражен возможностями, предоставляемыми iPhoto для размещения моих фотографий. В частности, когда я загружаю мои цифровые снимки с любого фотоаппарата и упорядочиваю их при помощи iPhoto, я просто хочу иметь возможность показать эти фото моей семье и моим друзьям. Я совсем не хочу получить хостинг и учетную запись на сервисе, предназначенном для печати фотографий, ждать пока куда-то загрузятся сотни файлов, конвертировать все изображения, уменьшив их размер, или вновь упорядочивать все мои фотографии в каком-либо другом приложения после того, как я сделал это в iPhoto. Я просто хочу, чтобы прямо сейчас мои фотографии были доступны всем для просмотра, и я не хочу даже лишний раз пальцем пошевелить, чтобы сделать это. Я уже проделал и так много работы, сделав фотографии, не считая того, что я их еще подписал и отсортировал!

Все это заставило меня разработать myPhoto (http://agent0068.dyndns.org/~mike/projects/myPhoto). Одной из особенностей Mac OS X, которую многие пользователи просто не замечают, является встроенный веб-сервер, включающий в себя как Apache, так и PHP, причем им обоим не терпится поработать. Добавьте к этому широкополосный доступ в Интернет, а также загрузите всю нужную информацию в iPhoto, и вы сможете с легкостью (как это и должно быть по идее) поделиться вашими фотографиями.

Если вашему проекту на PHP потребуется фотогалерея, то очень заманчиво будет возложить на плечи пользователя всю работу по загрузке, созданию подписей и упорядочиванию всех их фотографий в вашей системе. Но, тем не менее, если пользователи уже проделали всю эту работу в iPhoto, сделайте все остальное за них! Вооружившись простым анализатором XML, можно получить всю ценную информацию из iPhoto и конвертировать ее в более простую форму, которая является более подходящей для работы с PHP.

Закулисный обзор: данные в формате iPhoto

Первым логичным шагом будет изучение работы iPhoto, чтобы вы могли понять, доступ к каким данным можно с легкостью получить.

ПРИМЕЧАНИЕ

Я исхожу из того, что вы пользуетесь iPhoto 7.0. Это наиболее современная и доступная на момент написания данной книги версия. Тем не менее, добавив там-сям кое-какие изменения, можно с легкостью применять данную концепцию к iPhoto других версий — кое-что из написанного я начинал делать с версии iPhoto 2.0.

На рис. 4.14 показан небольшой фрагмент из альбома iPhoto.

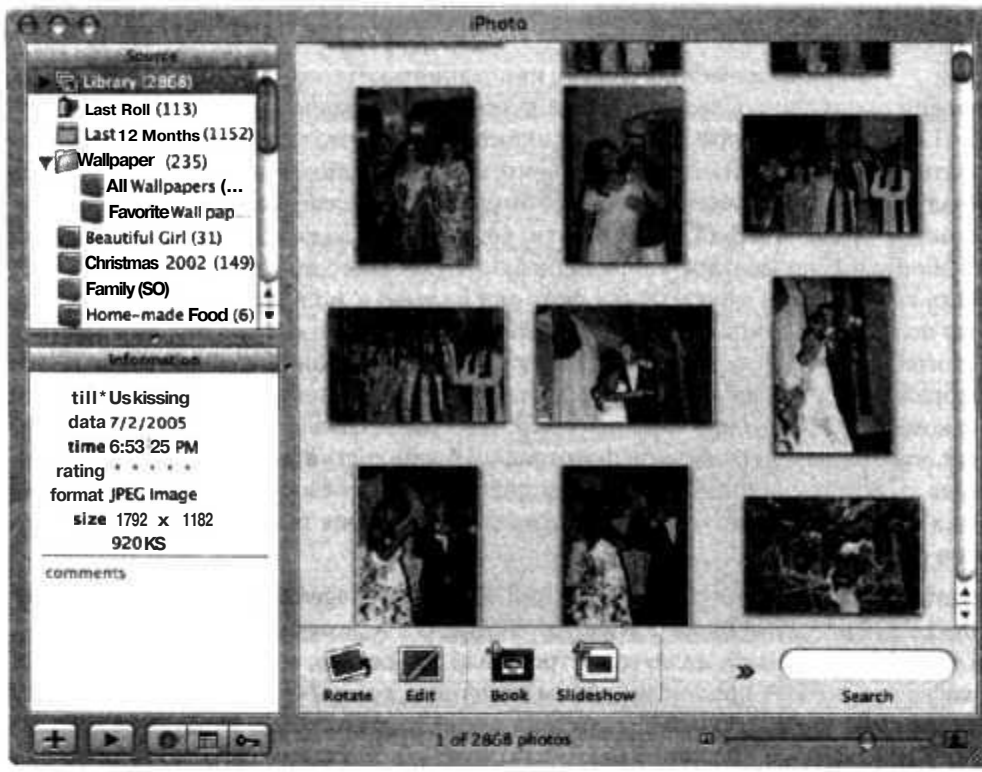


Рис. 4.14. Альбом фотографий с моей свадьбы в iPhoto

Быстрого взгляда в директорию `~/Pictures/iPhoto Library/` достаточно для получения информации обо всем, что нас интересует относительно iPhoto.

- **Директории разбиты по дате.** Например, в директории `~/Pictures/iPhotoLibrary/2005/07/02/` содержатся фотографии от 2 июля 2005 года. Изображения в этой директории полноразмерные, но в них также содержатся все изменения, сделанные пользователем при работе с iPhoto (например, разворот, смена цветов и т. д.).

В данной директории также содержатся две поддиректории: Thumbs с миниатюрами для каждой из фотографий размером 240 x 180 пикселов и Originals, в которой хранятся исходные, неизмененные изображения (только в том случае, если пользователь производил хотя бы какие-либо изменения с данными изображениями в iPhoto). К тому же в большинстве случаев эти изображения хранятся в JPEG-формате, являющемся наилучшим вариантом для Интернета.

ПРИМЕЧАНИЕ

Важное замечание: если пользователь сохраняет изображения в RAW-формате (доступен для фотоаппаратов класса hi-end), в директории Originals будут храниться RAW-файлы, а все другие изображения будут храниться в JPEG-формате.

- AlbumData.xml. В этом XML-документе содержится вся самая интересная (а также не очень интересная) информация относительно данных фотографий: путь к файлу для каждой фотографии, заголовки, рейтинги, даты последних изменений и т. д. Кроме того, в этом файле содержится информация о целых группах фотографий, называемых альбомами, а также об определенных пользователями ключевых словах. В нем также хранятся метаданные и дополнительная информация о версиях, но она не является критически важной.

Теперь нам нужно воспользоваться полученной из файла `AlbumData.xml` информацией. Для начала следует сказать, что это не просто XML-файл, а файл в формате Apple Property List. Это означает, что этот ограниченный набор XML-тегов используется для определения общих для программирования структур данных, таких как строки (strings), числовые данные (integers), массивы (arrays) и словари (dictionaries) (также известные как ассоциативные массивы в некоторых других языках программирования). Следовательно, для интересующих нас структур из данного файла необходимо найти соответствующие и стандартные типы, так как сами по себе XML-теги не очень хорошо описаны. Правильнее будет сказать, что в содержимом тегов хранится самая важная информация о структурах. Я вырезал для краткости только несколько фрагментов, но более важная информация из файла предоставлена здесь целиком.

Начало файла выглядит примерно так (что само по себе нас не сильно интересует):

```
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
<key>Application Version</key>
<string>5.0.4 (263)</string>
<key>Archive Path</key>
<string>/Users/mike/Sites/myPhoto/iPhoto Library</string>
```

Но дальше идет список всех фотографий в алфавитном порядке с уникальным идентификатором для каждого из снимков. В следующем примере вы можете видеть, что мы рассматриваем отдельную фотографию с уникальным ID, равным 5. Кроме того, это изображение (определенно изображение, а не видео) с заголовком

«No more pictures, please» и с таким же ключевым словом (уникальный ID данного ключевого слова равен 2).

```
<key>Master Image List</key>
<dict>
<key>5</key>
<dict>
<key>MediaType</key>
<string>Image</string>
<key>Caption</key>
<string>No more pictures. please</string>
<key>Aspect Ratio</key>
<real>0.750000</real>
<key>Rating</key>
<integer>0</integer>
<key>DateAsTimerInterval</key>
<real>62050875.000000</real>
<key>ImagePath</key>
<string>/Users/mike/Sites/myPhoto/iPhoto Library/2002/12/19/DSC00107.JPG</
string>
<key>OriginalPath</key>
<string>/Users/mike/Sites/myPhoto/iPhoto Library/2002/12/19/Originals/
DSC00107.JPG</string>
<key>ThumbPath</key>
<string>/Users/mike/Sites/myPhoto/iPhoto Library/2002/12/19/Thumbs/5.jpg</
string>
<key>Keywords</key>
<array>
<string>2</string>
</array>
</dict>
<key>6</key>
...and so on...
</dict>
```

В другом разделе данного файла (содержимое которого показано в следующем XML-фрагменте) перечислены все определенные пользователем группы фотографий, известные в iPhoto как альбомы (albums). Они хранятся в указанном пользователем порядке в массиве (в отличие от Master Image List, информация в котором расположена исходя из определенных ключей). Сюда включены все типы альбомов: нормальные альбомы, маленькие альбомы, папки, альбомы с возможностью просмотра слайдов, книжные альбомы и т. д. Описаны также различные атрибуты альбомов, например уникальный ID, имя, упорядоченный список ID-фотографий для снимков, содержащихся в определенных альбомах, признак того, что данный альбом является основным (master) (в каждой фототеке должен быть только один основной альбом), ID родительского (parent) альбома, если тип альбома определен как альбом-папка (folder album)) и т. д.

```
<key>List of Albums</key>
<array>
<dict>
<key>AlbumId</key>
<integer>2</integer>
<key>AlbumName</key>
```

```
<string>Vacation to somewhere</string>
<key>KeyList</key>
<array>
  <string>4425</string>
  <string>4423</string>
  <string>4421</string>
  <string>4419</string>
</array>
<key>Master</key>
<true/>
<key>PhotoCount</key>
<integer>2868</integer>
<key>Parent</key>
<integer>2196</integer>
</dict>
<dict>
...and so on...
</dict>
</array>
```

Кроме того, следует отметить, что здесь объявлена структура с ключом List of Rolls, идентичная структуре List of Albums. Этот автоматически генерируемый список группирует снимки каждый раз, когда они импортируются в iPhoto, работая с группой как с одной «катушкой» пленки. И, наконец, последним важным фрагментом в данном файле является список ключевых слов, упорядоченный в алфавитном порядке по ID. Есть ключевые слова, определенные пользователем, и которые вы можете использовать, занося в тег сразу несколько фотографий, вместо того чтобы вручную указывать один и тот же заголовок для каждой фотографии. Он состоит из пар ID/ключевое слово. В этом примере ID равен 1 и ключевым словом является _Favorite_:

```
<key>List of Keywords</key>
<dict>
<key>1</key>
<string>_Favorite_</string>
<key>2</key>
<string>...and so on...
</dict>
```

ПРИМЕЧАНИЕ

Имейте в виду, что в более старых версиях iPhoto формат файла немного другой. Удостоверьтесь, что вы знаете и понимаете, что работаете с подходящим для данной версии iPhoto файлом. Периодически туда заносятся некоторые изменения, что может привести к ошибкам в вашем исходном коде, отвечающем за анализ, если вы не внесете исправления или не обратите внимания на эти изменения.

Код

Сохраните исходный код из примера 4.10 в файл iphoto_parse.php.

Пример 4.10. Обработка XML из iPhoto

```
<?php
//$curTag обозначает текущий тег. который мы ищем в форме набора строк
```

```
//ScurKey обозначает текущий атрибут метки, который помнит
//последний атрибут.
//т.е. $curKey="AlbumName" для <key>AlbumName</key>
//$data обозначает элемент между тегами.
//т.е. $data="Library" для <string>Library</string>
//Читая код, обратите внимание, что ScurKey не обязательно равен $data.

ScurTag="";
$curKey="";
$readingAlbums=false;
$firstTimeAlbum=true;
$firstTimeAlbumEntry=true;

$readingImages=false;
$firstTimeImage=true;
$firstTimeImageEntry=true;
$curID=0;

$masterImageList=array( );

class Photo
{
  var $Caption;
  var $Date;
  var $ImagePath;
  var $ThumbPath;
}

function newPhoto(Scapt, $dat. SimgPath, $thumb)
{
  $aPhoto=new Photo( );
  $aPhoto->Caption=$capt;
  $aPhoto->Date=$dat;
  $aPhoto->ImagePath=$imgPath;
  $aPhoto->ThumbPath=$thumb;
  return SaPhoto:
}

//Эта функция вызывается при открытии тега
function startElement($parser, $name, $attrs)
{
  global $curTag;
  ScurTag .- "^$name";
}

//Эта функция вызывается при закрытии тега
function endElement($parser, Sname)
{
  global ScurTag;
  $caret_pos = strrpos($curTag,'^'1•,
  ScurTag = substr($curTag,0,$caret_pos);
}

//В этой функции присутствует код для просмотра того, что находится между тегами
function characterData($parser, $data)
{
  global ScurTag, ScurKey. $outputAlbums, $outputImages,
```

```
        $readingAlbums, $firstTimeAlbum, $firstTimeAlbumEntry,
        $readingImages, $masterImageList, $firstTimeImage,
        $firstTimeImageEntry, $curID;
        //Очищаем от ненужных данных
        $data - str_replace('!$-a-0*', '&', $data):

if(!ereg("(\t)+(\n)?$", $data) && !ereg("^\n$", $data))
//если $data=не пробел
{
  //Общие моменты. Просто список с открытыми тегами
  //атрибутов album, т.е. "AlbumName"
  $albumName = "^PLIST^DICT^ARRAY^DICT^KEY";
  $integerData = "^PLIST^DICT^ARRAY^DICT^INTEGER";//ID альбома
  $stringData = "^PLIST^DICT^ARRAY^DICT^STRING"; //Текущее название альбома
  $albumContents = "^PLIST^DICT^ARRAY^DICT^ARRAY^STRING"; //номер photo ID
  $majorList - "^PLIST^DICT^KEY"; //"List of Albums", "Master Image
  List"
  $photoID - "^PLIST^DICT^DICT^KEY"; //уникальный ID отдельной
  фотографии
  $photoAttr="^PLIST^DICT^DICT^DICT^KEY"; //"Caption", "Date", "ImagePath"
  //название, путь к файлу и т. д.
  $photoValStr="^PLIST^DICT^DICT^DICT^STRING";
  //дата, коэффициент сжатия и т. д.
  $photoValReal="^PLIST^DICT^DICT^DICT^REAL";
  if($curTag == SmajorList)
  {
    if($data=="List of Albums")
    {
      //Ставим флаг для  <key>List of Rolls</key>
      $readingAlbums=true;
      $readingImages=false;
    }
    else if($data=="Master Image List")
    <
      $readingAlbums=false;
      $readingImages=true;
    }
    else
    $readingAlbums=false;
  }
  if($readingAlbums)
  {
    if  ($curTag==$integerData)
    {
      if($data — "AlbumId")
      {
        $curKey = $data:
      }
    }
    //Ищем атрибут, то есть AlbumName
    else if ($curTag==$albumName)
    { //Следующая вещь, которую мы видим,- это название альбома
      //или список всех фотографий в альбоме
      if($data == "AlbumName" |j $data="KeyList")
      {
        $curKey = $data; //$curKey будет напоминать для нас время
```

```
    }
  }
  else if($curTag - $stringData || ScurTag — SintegerData)
  //Сейчас ищем интересные данные.
  {
    //Так. последний атрибут, который мы видели, был AlbumName.
    if($curKey -» "AlbumName")
    {
      ScurAlbum = Sdata; //say the album name was "Library"...
      //теперь $data="Library"
      $curAlbum = str_replace("&amp:". '&'. Sdata):
      SserializedObj - "";
      if(!$firstTimeAlbum)
        SserializedObj.="\n\t\t)\n\t.\n":
      $serializedObj .- "\t\"".addslashes($curAlbum).
      "\" =>\n\t\tarray(\n":
      $firstTimeAlbum=false;
      fileWrite($outputAlbums,$serializedObj,'a');
      $firstTimeAlbumEntry=true;
    }
  }
  else if($curTag == $albumContents) // ищем список фотографий
  {
    if($curKey — "KeyList")
    {
      //$data==ID-номер фотографии в ScurAlbum
      SserializedObj - "":
      if(!$firstTimeAlbumEntry)
        $serializedObj.=",\n";
      $serializedObj .= "\t\t\tSdata";
      fileWrite($outputAlbums,$serializedObj,'a');
      $firstTimeAlbumEntry=false;
    }
  }
  //Заполняем все остальные альбомы...
}
else if($readingImages)
{
  if($curTag==$photoID) //мы нашли новую фотографию, сохраняем ID.
  {
    ScurID="";
    if(!$firstTimeImage)
      ScurID=").\n";
    $curID.="\t\"$data\"=>array(";
    $firstTimeImageEntry=true;
    $firstTimeImage=false;
  }
  else if(ScurTag==SphotoAttr)
    if($data=="Caption" || $data=="DateAsTimerInterval" ||
      Sdata=="ImagePath" || $data=="ThumbPath")
      $curKey=$data;
    else
      $curKey="";
  }
  else if($curTag==$photoValStr || $curTag==$photoValReal)
```

```
    {
      if($curKey -- "Caption" || $curKey == "DateAsTimerInterval" ||
         $curKey=="ImagePath" || $curKey=="ThumbPath")
      {
        if(!$firstTimeImageEntry)
          $curID.=", ":
        if($curKey=="Caption")
          $curID .- "\"caption\"=>\"".addslashes($data)."\"";
        //интервал времени на основе типа Даты
        else if($curKey=="DateAsTimerInterval")
          // измеренный в секундах от 1/1/2001
          $curID .- "\"date\"=>\"".
          date("F j. Y. g:i a". mktime(0,0,$data,1,1,2001)).
          "\"":
        else
          $curID .= "\"$curKey\"=>\"$data\"";
        $firstTimeImageEntry=false:
      }
      if($curKey=="ThumbPath") //последний атрибут для фотографии...
        fileWrite($outputImages.$curID,'a');
        //...и извлечение других данных о фотографии...
    }
  }
}
}

//Эта функция вызывается для анализа XML
function parseAlbumXML($albumFile)
{
  global $outputAlbums, $outputImages;
  $xml_parser - xml_parser_create( );
  xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
  //разбираем с помощью функций
  xml_set_element_handler($xml_parser. "startElement", "endElement");
  xml_set_character_data_handler($xml_parser. "characterData");
  if (!($fp = fopen($albumFile. "r")))
    die("Can't open file: $albumFile");
  fileWrite($outputAlbums,"<?php\n$albumList - array (\n",V):
  fileWrite($outputImages,"<?php\n//key=photo ID. value={",'w');
  fileWrite($outputImages," [0]caption. [1]date. [2]image",'w');
  fileWrite($outputImages,"path. [3]thumb path}\n$masterList=array(\n",'w');
  while ($data = fread($fp, 4096))
  {
    $data - str_replace('&', '!$-a-0*', $data);
    if (!xml_parse($xml_parser, $data. feof($fp)))
    {
      die(sprintf("$albumFile : ".$lang["errXMLParse"].": %s at line Id"
      xml_error_string(xml_get_erTor_code($xml_parser)),
    ) xml_get_current_line_number($xml_parser)));
  }
  fileWrite($outputAlbums,"\n\t\t)\n\t\n\n);\n?>",'a');
  fileWrite($outputImages,")\n);\n?>",'a'):
  //все. выходим из парсера
  xml_parser_free($xml_parser);
  echo "Done parsing.";
```

```
}
function fileWrite($dest, $dataToWrite, $writeMode)
{
  global $err:
  if (is_writable($dest))
  {
    if C!$fp = fopen($dest, $writeMode))
      $err .= "Can't open file: ($dest) <br>";
    else
    {
      if (!fwrite($fp, $dataToWrite))
        $err .= "Can't write file: ($dest) <br>";
      fclose($fp):
    }
  }
  else
    ierr .= "Bad file permissions: ($dest) <br>":
}
set_time_limit(0); //если у нас огромный файл AlbumData.xml,
//По умолчанию в PHP 30-секундный тайм-аут выполнения
$outputImages="out_images.php";
$outputAlbums="out_albums.php";
parseAlbumXML("myPhoto/iPhoto Library/AlbumData.xml");
?>
```

Кроме того, чтобы воспользоваться выходным файлом из предыдущего кода, сохраните исходный код примера 4.11 в файл iphoto_display.php. Он будет отвечать за вывод изображений через Интернет.

Пример 4.11. Сценарий для отображения фотографий

```
<?pnp
include "out_images.php";
$photoIDs=array_keys($masterList);
$thumbsPerPage=6;
$thumbsPerRow=3:
if(!isset($_GET["tStart"]))
  $thumbStart=0;
else
  $thumbStart=$_GET["tStart"];
if($thumbStart+$thumbsPerPage>count($photoIDs))
  $thumbLimit=count($photoIDs);
else
  $thumbLimit=$thumbStart+$thumbsPerPage;
echo "<table border=\"0\" width=\"100%\">\n";
for($x=$thumbStart; $x<$thumbLimit; $x++)
{
  $aPhoto=$masterList[$photoIDs[$x]];
  $thumb="<table>";
  $thumb.="<tr><td align=\"center\"><img ";
  $thumb.="src=\"".$aPhoto["ThumbPath"]."\"></td></tr>";
  $thumb.="<tr><td align=\"center\"><small>";
  $thumb.=$aPhoto["date"]."<br>".$aPhoto["caption"]."</small></td></tr>";
  $thumb.="</table>";
  if($x % $thumbsPerRow == 0)
    echo "\n<!--New row-->\n<tr><td>\n".$thumb."\n</td>\n";
```

```
  else if($x 3! $thumbsPerRow == ($thumbsPerRow-1))
    echo "\n<td>\n".$thumb."\n</td></tr>\n<!--End row-->\n";
  else
    echo "\n<td>\n".$thumb."\n</td>\n";
}
echo "\n</table>\n";
?>
```

Запуск трюка

В последних нескольких строках файла iphoto_parse.php содержатся строго установленные пути к файлу AlbumData.xml, а также к итоговому файлу (полученному в результате работы сценария iphotodisplay.php), поэтому удостоверьтесь, что вы ввели правильные пути. Затем просто откройте файл iphoto_parse.php в вашем браузере. Кроме того, стоит отметить, что PHP необходимо разрешение на запись в выходной файл, так как иначе он просто не будет создан. Браузер скажет вам, когда сценарий закончит работу, выдав сообщение Done parsing. Откройте полученные файлы, и вы увидите в каждом из них массивы, похожие на описанные в примерах ниже.

Содержимое файла out_albums.php будет выглядеть приблизительно так:

```
<?php
$albumList = array (
  "Library" =>
  array(
    4425,
    4423,
    ...
    3796,
    3794,
   3792
  )
);
?>
```

Содержимое же файла out_images.php будет выглядеть так:

```
<?php
//key=photo ID, value={[0]caption, [1]date, [2]image path, [3]thumb path}
$masterList = array (
"13"=>array(
  "caption"=>"The wreath, out of focus again",
  "date"=>"December 23, 2002, 2:59 am",
  "ImagePath"=>"/~mike/myPhoto/iPhoto Library/2002/12/22/DSC00151.JPG",
  "ThumbPath"=>"/~mike/myPhoto/iPhoto Library/2002/12/22/Thumbs/13.jpg"),
  ...
);
?>
```

Вы можете просмотреть результат, открыв в браузере файл iphoto_display.php (рис. 4.15). Поскольку XML-формат очень многофункциональный, то файл AlbumData.xml может достигать больших размеров, храня в себе даже средних размеров библиотеку с фотографиями. У меня в итоге в библиотеке хранится всего 2868 фотографий, а размер файла AlbumData.xml уже достиг 2,4 Мбайт. Таким

образом, я рекомендую вам использовать обработчик XML, включенный в PHP 4 для анализа файла AlbumData.xml и разбиения его на полноценные фрагменты, которые я в дальнейшем буду приводить к более простому виду. В частности, полученные в результате обработки данные разделяются на два файла, в которых хранятся интересующие нас даты в виде PHP-массивов.

Рис. 4.15. Свадебные фотографии, сохраненные в iPhoto и открытые в моем браузере

Основной идеей обработчика является использование строк, хранящих структуру тегов, представляя таким образом файл в более удобочитаемом виде. Он напоминает стек, больше похожий на строку, чем на привычный нам массив или связанный список. Имейте в виду, что этот анализатор обрабатывает только некоторые элементы из фрагмента файла AlbumData.xml, в котором хранится информация об альбоме, а также из фрагмента с информацией о снимках. Я также продемонстрировал вам, как вы можете работать с полученными в результате анализа данными. Прежде чем садиться писать код, возможно, было бы неплохо решить, как вы будете работать с вашими фотографиями. Например, по умолчанию в Mac OS X у Apache (и, следовательно, у PHP) нет доступа к директории ~/Pictures/, где хранится вся информация из iPhoto, и, таким образом, вам придется напрямую задавать права доступа. Вы можете сделать это несколькими способами.

- Внесите соответствующие изменения в файл /etc/httpd/httpd.conf.
- Воспользуйтесь символическими ссылками.

J Выйдите из iPhoto, переместите директорию Library в директорию ~/Sites/, перезапустите iPhoto и, увидев сообщение о том, что все ваши фотографии были удалены, укажите программе новое месторасположение директории Library.

❑ Загрузите директорию Library на другой компьютер при помощи FTP, воспользовавшись утилитой rsync или какой-либо другой наиболее подходящей для вас программой, предназначенной для передачи файлов.

Улучшаем трюк

У вас есть богатые возможности по улучшению данного трюка.

O Улучшить обработчик XML, чтобы с его помощью получать дополнительную информацию, которая вам интересна, а не только информацию об альбомах и содержащихся в них файлах.

❑ Вместо того чтобы хранить полученную из AlbumData.xml информацию в текстовом файле, можно разместить ее в базе данных SQL или в каком-либо другом формате с более богатыми возможностями.

- Если вы действительно собираетесь создать настолько дружественную к пользователям программу, использующую информацию из iPhoto, почему бы не пойти дальше и полностью не автоматизировать процесс? Это сделать на самом деле очень просто. Для этих целей будут использоваться средства как для обработки XML-файла, так и для кэширования полученной в результате анализа информации. Последним шагом будет механизм, определяющий, когда следует использовать кэш, а когда его необходимо создавать заново. То, как будет решена эта задача, будет зависеть от вашего приложения, но вот некоторые моменты, которые следует взять на вооружение.
  - Пользуйтесь службой cron, которая будет вызывать вашу функцию для пересоздания кэша ежечасно/ежедневно или по истечении необходимого вам промежутка времени.
  - Отслеживайте дату изменения файла AlbumData.xml. Если вы анализировали файл до внесения в него изменений, то проведите его анализ заново.

Так, например, для реализации последнего добавьте в ваше приложение функцию с приблизительно следующим содержанием:

```
//Возвращаем булево значение, указывающее, необходимо ли
//снова анализировать кэш
function needToUpdateCache( )
{
  global $cacheTime, $albumFile, $err;
  $cacheTimeFile="lastCacheTime.txt"; //text file where
  //Строка показывает
  //время последнего анализа кэша.
  //то есть "January 28 2005 16:31:26."
  $compareFile="iPhoto Library/AlbumData.xml";
  if (file_exists($cacheTimeFile))
  {
```

```
    //Проверяем файл, в котором было сохранено время последнего анализа кэша
    if($fp - fopen($cacheTimeFile, "r"))
    {
      $lastTime = fread($fp, filesize($cacheTimeFile));
      fclose($fp);
    }
    else
    {
      $err.= "Can't read last cache time";
      return true;
    }
    // Сейчас определим последнее время изменения данных iPhoto.
    //Если нужен повторый анализ, запишем
    //текущее время в $cacheTimeFile
    //(поэтому мы повторно проанализируем)
    if($lastTime!=date ("F d Y H:i:s.", filemtime($compareFile)))
    {
      if (!$fp = fopen($cacheTimeFile, 'w'))
      {
        $err.= "Can't open file: $cacheTimeFile";
      }
      else
      {
        if (!fwrite($fp, date ("F d Y H:i:s.", filemtime($compareFile)) ))
        $err.= "Can't open file: $cacheTimeFile";
        fclose($fp);
      }
      return true;
    }
    else
      return false;
  }
  else
  {
    $err.- "Can't find file: $cacheTimeFile";
    return true;
  }

//В начале каждой загрузки страницы убеждаемся,
//что мы получили последние фотографии
if(needToUpdateCache( ))
  parseAlbumXML($pathToYourAlbumXMLFile);
```

Используя ее, вы можете быть уверены, что производите анализ файла только в том случае, если в iPhoto были внесены какие-либо изменения.

Смотрите также

- «Реализация предпросмотра изображений» (см. трюк 27).
- «Наложение изображений друг надруга» (см. трюк 32).

Базы данных и XML

Трюки 34-50

Большинство приложений на PHP используют в своей работе базы данных, и чаще всего это базы данных MySQL. В данной главе описано несколько трюков, изучив которые, вы сможете создавать приложения с возможностью доступа к базам данных, использующих в своей работе XML. В частности, вам следовало бы обратить внимание на трюк, использующий динамический объект базы данных, в котором разработан единственный класс для взаимодействия с любой базой данных. Добавьте к этому трюки для генерирования кода, которые позволяют вам автоматизировать доступ к базе данных, используя представление схемы в виде XML, и вы увидите, что в работе с базами данных в PHP нет ничего сложного!

ТРЮК №34

Разработка более качественных схем SQL

Большинство PHP-приложений используют базы данных SQL. Здесь я дам вам несколько советов, позволяющих избежать самых распространенных ошибок.

При разработке серверных СУБД на PHP обычно используются базы данных MySQL. Я сам разработал множество приложений, причем это были как приложения с открытым кодом, так и коммерческие и использующие в своей работе базы данных. В процессе их разработки я столкнулся с некоторыми периодически появляющимися проблемами, и здесь я постараюсь их перечислить, а также описать возможные пути их решений.

Плохо подобранный первичный ключ

Чтобы все записи в базе данных были уникальными, вам необходим первичный ключ. Обычно это уникальное, не повторяющееся целое число, начинающееся с единицы. Во всех базах данных есть возможность управлять первичными ключами без вашего участия, но, похоже, что не все разработчики знают об этом. Взгляните на простую схему в примере 5.1. Вы увидите таблицу со списком авторов с полями `id` и `name`.

Пример 5.1. SQL с первичным ключом

```
DROP TABLE IF EXISTS author;
CREATE TABLE author (
  id INT.
  name TEXT
);
```

Но кто же проверяет, является ли ID уникальным? Скорее всего, код на PHP, использующий таблицу, подобную этой, сперва выполнит команду SELECT для нахождения максимального значения поля id, а затем создаст новую запись, добавив к этому значению единицу. Но для этого требуется использование дополнительной команды SQL, а самому PHP-разработчику приходится помнить о необходимости выполнять вышеуказанные шаги. Было бы намного лучше, чтобы база данных сама выполняла эту (довольно, кстати, утомительную) работу.

В примере 5.2 показана намного более продуманная версия схемы из примера 5.1.

Пример 5.2. Добавление автоматически увеличивающегося поля id

```
DROP TABLE IF EXISTS author;
CREATE TABLE author (
  id INT NOT NULL AUTO_INCREMENT,
  name TEXT,
  PRIMARY KEY( id )
);
```

Теперь поле id определено как числовое и как автоматически увеличивающееся на единицу, причем его значение не может быть равно нулю. Оно также является первичным ключом. Чтобы добавить новую запись в таблицу такого типа, воспользуйтесь следующей командой:

```
INSERT INTO author VALUES ( 0. "Brad Phillips" );
```

MySQL заменяет нулевое значение поля ID автоматически увеличивающимся на единицу значением. Чтобы узнать номер ID последней добавленной записи, воспользуйтесь командой SELECT:

```
SELECT LAST_INSERT_ID( );
```

Эта версия таблицы работает намного быстрее, чем первая, так как на базе первичных ключей создается каталог, что ускоряет работу поиска.

Если вы столкнетесь с кодом, в котором при добавлении записи сперва при помощи команды SELECT находится максимальное значение поля id, а затем выполняется команда INSERT, в которую и передается это значение, увеличенное на единицу, знайте, что в этом коде не используются значения основных ключей, автоматически увеличивающиеся на единицу. И дело здесь не только в производительности: на сайтах, требующих передачи больших объемов информации, запросто могут добавиться несколько записей с одинаковым первичным ключом из-за большой задержки между выполнениями команды SELECT, ищущей самое большое значение первичного ключа, и командой INSERT, создающей новую запись.

Неправильное понимание принципов работы реляционных баз данных

Реляционные базы данных, такие как Oracle и MySQL, отличаются от размещаемых в памяти структур данных, которые вы создаете в различных языках программирования (таких как PHP, C и Java) или объектно-ориентированных языках. В частности, языки программирования могут создавать структуры данных, содержащие в себе массивы. Структуры данных такого рода не могут быть так же просто использованы в реляционных базах данных. В примере 5.3 вы можете увидеть схему, в которой текстовое поле применяется в качестве массива для ID.

Пример 5.3. Применение массивов, используемых в обычных языках программирования

```
DROP TABLE IF EXISTS author;
CREATE TABLE author (
  idINT.
  name TEXT
);

DROP TABLE IF EXISTS book;
CREATE TABLE book (
  id INT.
  name TEXT.
  authors TEXT
);
```

На рис. 5.1 показаны связи между этими двумя таблицами (то есть на самом деле отсутствие какой-либо связи между ними).

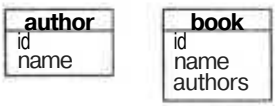


Рис. 5.1. Таблицы name и book

Рассмотрим пример того, как вы, возможно, будете пользоваться этими таблицами:

```
INSERT INTO author VALUES ( 1. "Brad Phillips" ):
INSERT INTO author VALUES ( 2. "Don Charles" ):
INSERT INTO author VALUES ( 3. "Brad silver" );
INSERT INTO book VALUES ( 1. "MySQL in a bucket". "1.2" );
INSERT INTO book VALUES ( 2. "Databases for Delinquents". "3" ):
```

Книги добавляются из списка с разделенными запятыми ID авторов. Едва ли это работает лучше, чем структуры баз данных, не так ли? Чтобы выяснить, какие авторы соответствуют тем или иным книгам, сценарий сперва посылает запрос на записи с книгами, затем разбивает текстовое поле с авторами, используя функцию `preg_split()`, и, наконец, выполняет еще целый набор запросов, используя полученные ID.

Теперь поговорим о том, как на самом деле это должно быть сделано. В примере 5.4 показана правильная схема с использованием трех таблиц. Одна для авторов, вторая для книг, а третья для связи первых двух.

Пример 5.4. Использование связующей таблицы

```
DROP TABLE IF EXISTS author;
CREATE TABLE author (
  id INT NOT NULL AUTO_INCREMENT,
  name TEXT.
  PRIMARY KEY( id )
);

DROP TABLE IF EXISTS book;
CREATE TABLE book (
  id INT NOT NULL AUTOJNCREMENT.
  name TEXT.
  PRIMARY KEY( id )
);

DROP TABLE IF EXISTS book author;
CREATE TABLE book_author 7
  book_id INT.
  author_id INT
):
```

Третья таблица в нашей схеме очень важна. Она, используя поля id из первых двух таблиц, хранящих автора и книгу, организует между ними связь. В таблице book_author также создается соответствующая запись. На рис. 5.2 показана связь между этими тремя таблицами.

Рис. 5.2. Теперь связь между таблицами авторов и книг организована правильно

Начальные данные заносятся следующим образом:

```
INSERT INTO author VALUES ( 0. "Brad Phillips" );
INSERT INTO author VALUES ( 0. "Don Charles" );
INSERT INTO author VALUES ( 0. "Brad silver" );
INSERT INTO book VALUES ( 0. "MySQL in a bucket" ):
INSERT INTO book VALUES ( 0. "Databases for Delinquents" );
INSERT INTO book_author VALUES ( 1. 1 ):
INSERT INTO book_author VALUES ( 1. 2 );
INSERT INTO book_author VALUES ( 2. 3 ):
```

Последние несколько команд INSERT связывают первых двух авторов с первой книгой, а третьего автора — с последней книгой. Вот запрос, при помощи которого можно получить таблицу с указанным в одной строке автором и книгой:

```
SELECT
  a.name AS author,
  b.name AS book,
  a.id AS author_id.
  b.name AS book_id
```

```
FROM
  author AS a.
  book AS b.
  book_author AS ba
WHERE
  a.id - ba.author_id AND
  b.id = ba.book_id;
```

Далее приведен похожий запрос (но только с одним аргументом, которым является какая-либо определенная книга):

```
SELECT
  a.name AS author,
  b.name AS book,
  a.id AS author_id,
  b.name AS book_id
FROM
  author AS a.
  book AS b,
  book_author AS ba
WHERE
  a.id = ba.author_id AND
  b.id = ba.book_id AND
  ba.book_id = 1;
```

Результатом этого запроса будут все авторы одной из книг, причем нет необходимости использовать подзапросы или обрабатывать текстовую строку. В результате мы имеем более быстрые запросы и схему базы данных, которая использует ее преимущества (реляционность), а не недостатки (такие как необходимость анализа строк).

Не используйте нулевые поля

Базы данных предоставляют богатые возможности для проверки хранимой в них информации. Но некоторые программисты не производят даже элементарной проверки, просто не зная, что существует такая возможность. Взгляните хотя бы на таблицу из примера 5.5.

Пример 5.5. Таблица, в которой пренебрегают проверкой хранимой информации, хотя она и подразумевается здесь

```
DROP TABLE IF EXISTS user;
CREATE TABLE user (
  id INT,
  first TEXT,
  last TEXT.
  username TEXT,
  password TEXT,
  description TEXT
);
```

Что мы знаем об этой таблице? Ну, для начала, мы знаем, что поля first, last, username и password никогда не должны быть пустыми. Согласитесь, было бы неплохо, чтобы база данных сама могла выполнить такого рода проверку. Конечно же, она может это сделать — посмотрите на пример 5.6.

Пример 5.6. Небольшие изменения, приводящие к очень печальному результату

```
DROP TABLE IF EXISTS user;
CREATE TABLE user (
  id INT NOT NULL AUTO_INCREMENT,
  first TEXT NOT NULL,
  last TEXT NOT NULL,
  user-name TEXT NOT NULL,
  password TEXT NOT NULL,
  description TEXT,
  PRIMARY KEY ( id )
);
```

Здесь я не только улучшил основной ключ, чтобы он генерировался автоматически, но также добавил условия NOT NULL к полям, которые не могут быть пустыми. Это позволит добиться того, что при попытке ввести некорректные данные будет выдаваться ошибка. Зачем полагаться исключительно на грамотное программирование, если база данных сама может следить за соблюдениями всех условий?

Смотрите также

- «Создание неприступных баз данных» (см. трюк 35).
- «Экспорт схем баз данных в XML» (см. трюк 39).
- «Формирование баз данных SQL» (см. трюк 41).

ТРЮК №35

Создание неприступных баз данных

Изучите способы использования модулей PEAR для работы с базами данных, чтобы научиться создавать неприступные базы данных для ваших интернет-приложений.

За последние годы я прочитал несколько книг по PHP, и почти во всех встречаются одни и те же ошибки, когда заходит речь о доступе к базам данных. Приложения, неправильно использующие SQL, потенциально подвержены SQL-атакам, основанным на вводе данных. Это может привести к тому, что хакеры повесят вашу базу данных (и, следовательно, сделают недоступным ее содержимое). Причем сделать базу данных с правильно организованным доступом к ней на самом деле проще, чем с организованным неправильно.

Чтобы продемонстрировать это, в примере 5.7 показана корректная конструкция SQL-команды.

Пример 5.7. Правильная конструкция SQL-команды

```
<?php
require_once("DB.php");
$dsn = 'mysql://root:password@localhost/books';
$db =& DB::Connect( $dsn, array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )); }
$sth = $db->prepare( "INSERT INTO author VALUES ( null, ? )" );
$db->execute( $sth, array( $_POST['name'] ) );
?>
```

Для подготовки выражения я пользуюсь модулем PEAR DB со знаком ? в том месте, где будут идти аргументы. Затем я отправляю на выполнение полученное выражение применительно к базе данных и получаю массив аргументов, которыми будут заполнены поля со знаком ?. Драйвер расставляет, где нужно, кавычки и символы управляющей последовательности, чтобы удостовериться, что команда будет выполнена правильно независимо от входных данных.

В соответствии с вышесказанным, в примере 5.8 показан вариант соответствующего запроса применительно к базе данных.

Пример 5.8. Команда SQL SELECT, которая не вызовет никаких проблем

```
<?php
require_once("DB.php");
$dsn = 'mysql://root:password@localhost/books';
$db =& DB::Connect( $dsn, array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )); }
$res = $db->query( "SELECT * FROM author WHERE id = ?", array( $id ) );
while( $res->fetchInto( $row ) )
{
...
}
```

В данном случае в метод query при вызове передается строка SQL, где на месте аргументов стоит символ ?. Значения аргументов задаются при помощи второго аргумента функции, который всегда является массивом (независимо от количества требуемых аргументов).

ПРИМЕЧАНИЕ

Вы, наверное, спрашиваете себя: «Если это правильная версия, то как же выглядит неправильная?». Я не собираюсь размещать неправильную версию в этой книге, так как читатели могут воспользоваться ею, заранее не уточнив, что этот исходный код на самом деле работает некорректно.

Некоторые из разработчиков PHP предполагают, что PEAR DB работает медленнее. Я этого не заметил, но даже если бы это было так, я бы по-прежнему использовал PEAR DB из-за его портативности и возможностей по обеспечению безопасности, чем не обладают обычные функции для доступа к базам данных.

Кстати, на горизонте уже маячит замена PEAR DB — библиотека PHP Data Objects (PDO). Пока она экспериментальная, но в будущем она позиционируется как альтернатива PEAR DB. Интересно отметить, что если вы будете пользоваться генераторами кода, описанными в этой главе, то сможете изменить PEAR DB на PDO, не внося никаких изменений в ваше приложение для доступа к базе данных более высокого уровня.

Смотрите также

- «Разработка более качественных схем SQL» (см. трюк 34).

ТРЮК №36 Создание динамических объектов для доступа к базам данных

Воспользуйтесь новыми объектно-ориентированными возможностями PHP 5 для создания классов, реализующих доступ к любым таблицам баз данных.

PHP 5 представляет из себя значительно улучшенную с точки зрения поддержки объектно-ориентированных возможностей версию языка PHP. Наряду с некоторыми улучшениями в производительности PHP 5 обладает более продвинутыми возможностями для создания *динамических классов*. Это классы, в которых их методы и свойства изменяются от объекта к объекту. Это может быть очень полезным при создании приложений, работающих с базами данных. Обычно для каждой таблицы в базе данных есть свой класс PHP. Например, если у вас есть таблицы с названиями books (книги), authors (авторы) и `publishers` (издатели), то у вас будут PHP-классы с названиями Book, Author и `Publisher`. У каждого PHP-класса есть методы, предназначенные для установки значений в записи в соответствующей таблице. С одной стороны, это очень простая и легкая в понимании модель. С другой стороны, для поддержки этих классов требуется много работы (и это только для трех таблиц!). Можно ли разработать только один класс, который будет работать с любой таблицей из базы данных? Да. Благодаря поддержке в PHP 5 функций `__call`, `__get` и `__set`.

Чтобы понять, почему методы `__call`, `__get` и `__set` настолько важны, вам необходимо понять механизм вызова методов объектами. Когда вы пытаетесь вызвать метод, интерпретатор сперва изучает класс, чтобы определить, есть ли у данного класса этот метод. Если он присутствует, то он вызывается, если нет, то изучается базовый класс данного класса и т. д. по всей цепочке классов. В PHP 5, когда поиск нужного метода ни к чему не приводит, вызывается метод `__call`, если он существует. В качестве аргументов в него передаются имя метода и массив с аргументами для него. Если вы объявите метод `__call`, который будет возвращать реальное значение, то PHP 5 будет считать, что он нашел то, что искал, и что нужный ему метод запущен и работает. Методы `__get` и `__set` предназначены для задания и получения значений определенных переменных объекта. У `__get` только один параметр: имя переменной. У метода `__set` два параметра: имя переменной и новое значение.

Это означает, что вы можете создавать новые методы и новые переменные на лету. Кроме того, у вас может быть класс, загружающий запись из таблицы базы данных и обладающий динамическими методами и переменными, причем он будет выглядеть как класс, созданный специально для этой записи. На рис. 5.3 показано, как работают эти динамические методы и поля класса. В коде мы обращаемся либо к методу, либо к полю. Затем объект определяет, что такого поля нет. PHP обращается к полю либо к методу в попытке получить значение, а затем, если получает правильный ответ, возвращает полученное значение, как если бы этот метод или это поле действительно были объявлены.

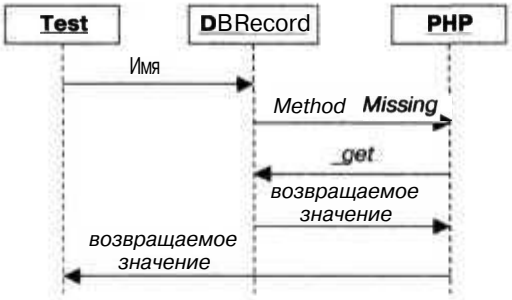


Рис. 5.3. Механизм работы с динамическими полями

Если все это кажется вам немного запутанным, не беспокойтесь. Это новая мощная возможность для объектно-ориентированного программирования. Чтобы освоить ее, требуется немного времени, и немного больше времени, чтобы удачно и безопасно пользоваться ею. Думайте об этом как о проверке новых возможностей.

Код

Сохраните исходный код из примера 5.9 в файл dbrecord.php.

Пример 5.9. Небольшой пример кода на PHP, являющегося на удивление мощным сценарием

```
<?pnp
require_once( "DB.php" );
Sdsn - 'mysql://root:password@localhost/books';
Sdb =& DB::Connect( Sdsn. array( ) ):
if (PEAR::isError($db)) { die(Sdb->getMessage( )): }

class DBRecord
{
  var $h;
  public function DBRecord( Stable. Sid )
  {
    global Sdb:
    Sres = Sdb->query( "SELECT * from Stable WHERE id=?", array( Sid ) );
    $res->fetchInto( Srow. DB_FETCHMODE_ASSOC );

  } $this->{'h'} - Srow;

  public function___call( $method. Sargs )

  } return $this->{'h'}[strtolower($method)];

  public function___get( Sid )
    return $this->{'h'}[strtolower($id)];
  }
}
?>
```

Для проверки воспользуйтесь исходным кодом из примера 5.10, предварительно сохранив его в файле test.php.

Пример 5.10. Простой сценарий для проверки работоспособности сценария, предназначенного для доступа к базе данных

```
<?php
require_once( "DBrecord.php" );

$rec - new DBRecord( "author". 2 );
print $rec->Name( )."\n";
?>
```

В файле books.sql, текст которого вы можете видеть в примере 5.11, для нашего примера создается база данных.

Пример 5.11. SQL-сценарий, создающий простую тестовую базу данных

```
DROP TABLE IF EXISTS author;
CREATE TABLE author (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  name TEXT,
  PRIMARY KEY( id )
);

INSERT INTO author VALUES ( 0. "jack" );
INSERT INTO author VALUES ( 0. "bob" );
```

Запуск трюка

Этот трюк запускается при помощи PHP-интерпретатора из командной строки:

```
% mysql --user=root --password=password books < books.sql
% php test.php
bob
```

Не похоже, что мы создали что-то действительно мощное, но самое интересное в том, что у нас есть объект, который выглядит полностью так же, как строка в таблице с авторами. Тем не менее тот же самый объект мог бы представлять собой запись в таблице книг или в таблице с издателями, то есть он не привязан к какой-либо определенной схеме базы данных или таблице. В коде просто создается новый объект типа DBRecord с заданными именем таблицы и ID записи в ней. Затем вызывается метод Name(), но у объекта типа DBRecord такого метода нет, поэтому выполняется метод __call. В этой функции имя метода переводится в нижний регистр (PHP всегда поступает так). После этого метод __cal l объекта DBRecord проверяет хешированную информацию, полученную из базы данных и записанную в переменную $h, и возвращает значение запрошенного поля.

Улучшаем трюк

Считать информацию из базы данных — это только полдела. Но можем ли мы улучшить наше приложение, чтобы оно могло не только читать, но и записывать в базу данных? Конечно же. Сохраните исходный код из примера 5.12 в файл dbrecord2. php.

Пример 5.12. Дополнительный исходный код, реализующий возможность не только чтения информации из базы данных, но и записи в нее

```
<?php
require_once( "DB.php" ):
$dsn = 'mysql://root:password@localhost/books';
$db =& DB::Connect( $dsn. array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )); }

class DBRecord
{
  var $h;
  var Stable:
  var $id;
  public function DBRecord( Stable. Sid )
  {
    global $db:
    $res - Sdb->query( "SELECT * from Stable WHERE id=?", array( Sid ) );
    $res->fetchInto( $row, DB_FETCHMODE_ASSOC ):
    $this->{'h'} - $row;
    Sthis->{'table'} - Stable:
    $this->{'id'} = Sid;
  }
  public function____call( $method, $args )
  {
    return $this->{'h'}[strtolower($method)];
  }
  public function____get( Sid )
  {
    print "Getting Sid\n";
    return $this->{'h'}[strtolower($id)];
  }
  public function____set( Sid. $value )
  {
    $this->{'h'}[strtolower($id)] - Svalue:
  }
  public function Update( )
  {
    global Sdb:
          $fields = array( ):
          $values - array( );
    foreach( array_keys( $this->{'h'} ) as Skey )
    {
      if C Skey !- "id" )
      {
        Sfields []= Skey." = ?";
        Svalues []= $this->{'h'}[$key];
      }
    }
    Sfields = joint ".". Sfields ):
    Svalues []= $this->{'id'};
    $sql = "UPDATE {$this->{'table'}} SET Sfields WHERE id = ?";
    $sth = Sdb->prepare( Ssql );
  } Sdb->execute( Ssth. Svalues ):
}
?>
```

Для проверки этого исходного кода сохраните текст программы из примера 5.13 в файл test2.php.

Пример 5.13. Сценарий для проверки динамического обновления базы данных

```
<?php
require_once( "DBrecord2.php" ):

$rec - new DBRecord( "author". 2 );
print $rec->Name( )."\n";
$rec->Name - "New Name";
$rec->Update( );
?>
```

Для проверки запустите файл test2. php:

```
% php test2.php
bob
% php test2.php
New Name
%
```

Сперва выдается текущее значение записи в базе данных. Затем оно изменяется на New Name, а сама запись в базе данных обновляется. Я перезапустил сценарий, чтобы удостовериться, что значение было изменено. Весь трюк заключается в том, что в метод__set передается в качестве аргумента New Name, и хеш полей из записи изменяет свое значение на новое. Затем вызывается метод Update(), который выполняет команду UPDATE для базы данных SQL.

ПРИМЕЧАНИЕ

Rails framework (http://www.rubyonrails.org/) для Ruby (http://ruby-lang.org/) использует похожий подход, позволяя интернет-приложениям быстро адаптироваться к любой схеме базы данных. Кажется, Cake (http://cakephp.org) разработал что-то похожее для PHP.

Смотрите также

- «Формирование кода, выполняющего команду базы данных CRUD» (смотрите трюк 37).
- «Формирование кода, выполняющего команду базы данных SELECT» (смотрите трюк 42).
- «Преобразование любого объекта в массив» (смотрите трюк 53).

ТРЮК №37 Формирование кода, выполняющего команду базы данных CRUD

Генерируйте код, предназначенный для создания, чтения, обновления и удаления записей (CRUD) из таблиц баз данных.

В этой книге вашему вниманию предоставлены несколько трюков, которые помогут вам ускорить процесс разработки базы данных при использовании автомати-

ческого формирования требуемого кода на PHP и SQL. В этом трюке я покажу вам, как разработать генератор, создающий классы для PHP 4 (или 5), предназначенные для работы с записями в базах данных. Используя эти классы, вы сможете создавать, считывать, обновлять и удалять конкретные записи в любых таблицах, не тратя огромное количество времени на создание исходного кода для работы с базами данных вручную. На рис. 5.4 показано взаимодействие файла со схемой и генератором, который, в свою очередь, создает исходный код на PHP. Я пометил сгенерированный код, закрасив его более темным цветом, так как он является временным и не должен изменяться вручную.

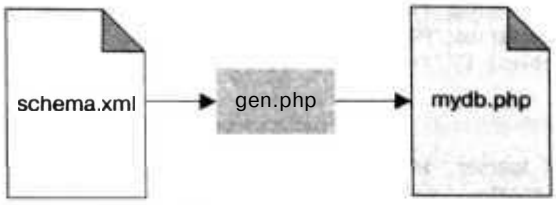


Рис. 5.4. Взаимодействие, налаженное через генератор

Код

Сохраните XML-документ со схемой базы данных (показанный в примере 5.14) в файл `schema.xml`.

Пример 5.14. XML-документ, отражающий схему базы данных

```
<schema>
<table name="book">
  <field name="id" type-"int" primary-key="true" />
  <field name="title" type="text" />
  <field name="publisher_id" type-"int" />
  <field name="author_id" type="int" />
</table>
<table name="publisher">
  <field name-"id" type-"int" primary-key="true" />
  <field name="name" type="text" />
</table>
<table name="author">
  <field name="id" type-"int" primary-key="true" />
  <field name="name" type="text" />
</table>
</schema>
```

В примере 5.15 показан исходный код, используемый для автоматического формирования кода. Я сохранил его в файле `gen.php`.

Пример 5.15. Приложение на PHP, реализующее автоматическое формирование исходного кода, предназначенного для работы с базами данных

```
<?php
Stables - array( ):
function start_element( Sparser, $name, $attribs )
{
  global Stables:
```

```
  if ( Sname == "TABLE" )
  {
    Stable = array( );
    Sfields = array( );
    $table['name'] - $attribs['NAME'];
    $table['fields'] - arrayt ):
    Stables [ ] - Stable;
  }
  if ( Sname == "FIELD" )
  {
    Sfield = arrayt );
    $field['name'] = $attribs['NAME'];
    $field['type'] - $attribs['TYPE'];
    $field['pk'] - ( $attribs['PRIMARY-KEY'] == "true" ) ? 1 : 0;
    $tables[count($tables)-1]['fields'] [ ] - Sfield:
  }
}

function end_element( Sparser. Sname ) { }
Sparser - xml_parser_create( );
xml_set_element_handler($parser, "start_element", "end_element" );
while( Tfeof( STDIN ) ) {
  $text - fgets( STDIN );
  xml_parse( Sparser. Stext );
}
xml_parser_free( Sparser );
ob_start( );
echo( "<?php\n" ):
?>
require_once( "dbwrap.php" );
<?php
foreach( Stables as Stable ) {
  Spk = null;
  Supdsets = array( );
  Supdfields = array( );
  Sinsfields - array( ):
  Sinsvalues - array( );
  Sinsvars - array( );
  foreach( $table['fields'] as Sfield ) {
    Sinsfields []» $field['name']:
    if ( $field['pk'] )
    {
      Spk = $field['name'];
      Sinsvalues []= 0;
    }
    else
    {
      Supdsets [ ] - $field['name']."=?";
      Supdfields []» '$this->'.$field['name'];
      Sinsvalues []= '?';
      Sinsvars [ ] - '$this->'.$field['name'];
    }
  }
  Sinsvars = join( Sinsvars. ". " );
  Sinsvalues - join( Sinsvalues. ". " ):
```

```
Sinsfields = join( $insfields, ". " );
Supdfields []= '$this->'.$pk;
Supdfields - joint Supdfields, ". " );
Supdsets - joint Supdsets. ", " );
?>
class <?php echot ucfirst( Stable['name'] ) ) ?>
{
 <?php
 foreacht $table['fields'] as Sfield ) {
  ?>
  var $<?php echot $field['name'] ); ?>;
  <?php
}
?>
function <?php echot ucfirst( $table['name'] ) ) ? > ( )
{
  $this->id = null;
}
function load($id)
{
  $data = selectOne( "SELECT * FROM <?php echot Stablername'] ) ?> WHERE <?php
  echot $pk ); ?> = ?", arrayt Sid ) );
  <?php
  foreacht $table['fields'] as Sfield ) {
   ?>
   $this-><?php echot $field['name'] ): ?> - $data['<?php echot $field['name']
   ); ?>']:
   <?php
  }
  ?>
}
<?php
foreacht $table['fields'] as Sfield ) {
  ?>
  function get_<?php echot $field['name'] ) ? > ( ) { return $this-><?php echot
    $field['name'] ) ?>: }
  function set_<?php echot $field['name'] ) ?>( $val ) { $this-><?php echot
    $field['name'] ) ?> = $val; }
  <?php
}
?>
function update( )
{
  if ( $this->id != null ) { $this->updateRecord( ): }
  else { $this->insertRecord( ); }
}
function insertRecord( )
{
  return executeCommand( "INSERT INTO <?php echot Stablername'] ) ?> ( <?php
  echo($insfields); ?> ) VALUES ( <?php echo($insvalues); ?> )".
  arrayt <?php echot $insvars ) ; ? > ) ) ;
}
function updateRecordt )
{
  return executeCommandt "UPDATE <?php echot $table['name'] ) ?> SET <?php
```

```
    echo($updsets); ?> WHERE <?php echo( $pk ); ?>-?",
    array( <?php echo( $updfields ); ?> ));
  }
  function deleteRecord( $id )
  {
    return executeCommand( "DELETE FROM <?php echo( $table['name'] ) ?> WHERE
    <?php echo( $pk ); ?>-?", array( $id ) );
  }
}
<?php }
echo( "?>" );
$php = ob_get_clean( );
$fh = fopen( "mydb.php", "w" );
fwrite( $fh, $php );
fclose( $fh );
?>
```

Верите вы или нет, однако нужно будет написать еще немного кода. Это будет сценарий dbwrap.php (показанный в примере 5.16), реализующий соединение с определенной базой данных. Вам нужно создать по одному такому сценарию для соединения с каждой из баз данных.

Пример 5.16. Сценарий для работы с определенной базой данных

```
<?php
require_once( "DB.php" );
$dsn = 'mysql://root:password@localhost/books';
$db =& DB::Connect( $dsn, array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )); }

function selectOne( $sql, $args )
{
  global $db;
         $res = $db->query( $sql, $args );
         $res->fetchInto($row, DB_FETCHMODE_ASSOC);
  return $row;
}

function selectBlock( $sql, $args )
{
  global $db;
         $res = $db->query( $sql, $args );
         $rows = array( );
  while( $res->fetchInto($row, DB_FETCHMODE_ASSOC) ) { $rows[] = $row; }
  return $rows;
}

function executeCommand( $sql, $args )
{
  global $db;
         $sth = $db->prepare( $sql );
  return $db->execute( $sth, $args );
}
?>
```

Сохраните исходный код примера 5.17 в файл insert.php. С его помощью можно проверить, добавляется ли информация в базу данных.

Пример 5.17. Тестовый сценаний, добавляющий информацию в базу данных

```
<?php
require_once( "mydb.php" );

$auth = new Author( );
$auth->set_name( "Jack" );
$auth->update( ):
?>
```

В примере 5.18 производится проверка загрузки данных. Его нужно сохранить в файл load.php.

Пример 5.18. Сценарий, проверяющий, загружаются ли данные из базы данных

```
<?php
require_once( "mydb.php" ):

$auth = new Author( );
$auth->load( 1 );
?>
Name: <?php echo( $auth->get_name( ) ): ?>
```

Сохраните исходный код примера 5.19 как файл с именем delete.php. Он проверяет возможность удаления информации из базы данных при помощи сгенерированных нами классов.

Пример 5.19. Сценарий для проверки возможности удаления данных

```
<?pnp
require_once( "mydb.php" );

$auth = new Author( );
$auth->deleteRecord( 1 ):
?>
```

Больше всего кода в данном трюке в сценарии gen. php. Это исходный код генератора, формирующего исходный код. Причем он формируется с учетом содержимого файла dbwrap. php, а тестируется при помощи сценариев insert.php, 1oad.php и delete.php. Сперва сценарий gen. php считывает XML-документ, отражающий схему базы данных. На самом деле большая часть сценария посвящена размещению XML в расположенной в памяти структуре. Этим занимается код в начале сценария и до строки xml_parse(), которая считывает XML и вызывает обработчик события, предназначенный для анализ тегов. После того как XML был считан, можно приступать к формированию кода. Этот процесс начинается с инициализации выходного буфера, происходящей в функции ob_start(). Затем генератор перебирает в цикле все таблицы и поля, создавая на основе полученной информации классы один за одним. Когда классы созданы, выходной буфер закрывается и сформированный код, предназначенный для доступа к базе данных и к данному моменту уже сохраненный в строке, записывается в файл.

Запуск трюка

При формировании кода для вашей базы данных сперва вам необходимо создать ее схему в XML-формате. Я разместил здесь файл под названием schema.xml,

в котором описана схема простой базы данных книг. Чтобы сформировать код для этой базы данных, я использую PHP-интерпретатор для командной строки:

```
% php gen.php < schema.xml
```

Генератор кода создает файл с названием mydb.php, содержащий классы для доступа к записям, хранимым в базе данных, написанные на PHP. Если вы воспользуетесь предоставленной здесь схемой, выходной файл должен будет выглядеть следующим образом:

```
<?php
require_once( "dbwrap.php" );

class Book
{
  var $id;
  var $title;
  var $publisher_id;
  var $author_id;

  function Book( )
  {
    $this->id = null;
  }

  function load($id)
  {
    $data = selectOne( "SELECT * FROM book WHERE id = ?", array( $id ) );
    $this->id = $data['id'];
    $this->title = $data['title'];
    $this->publisher_id = $data['publisher_id'];
    $this->author_id = $data['author_id'];
  }

  function get_id( ) { return $this->id; }

  function set_id( $val ) { $this->id = $val; }

  function get_title( ) { return $this->title; }

  function set_title( $val ) { $this->title = $val; }

  function get_publisher_id( ) { return $this->publisher_id; }

  function set_publisher_id( $val ) { $this->publisher_id = $val; }

  function get_author_id( ) { return $this->author_id; }

  function set_author_id( $val ) { $this->author_id = $val; }

  function update( )
  {
    if ( $this->id != null ) { $this->updateRecord( ); }
    else { $this->insertRecord( ); }
  }
  function insertRecord( )
```

```
  {
    return executeCommand( "INSERT INTO book ( id, title, publisher_id,
    author_id ) VALUES ( 0, ?, ?, ? )",
    array( $this->title, $this->publisher_id, $this->author_id ) );
  }

  function updateRecord( )
  {
    return executeCommand( "UPDATE book SET title=?, publisher_id=?, author
    id-? WHERE id=?",
    array( $this->title, $this->publisher_id, $this->author_id, $this->
    id ) );
  }

  function deleteRecord( Sid )
  {
    return executeCommand( "DELETE FROM book WHERE id=?", array( Sid ) );
  }
}

class Publisher
{
  var Sid;
  var $name;

  function Publisher( )
  {
    $this->id = null;
  }

  function load($id)
  {
    $data = selectOne( "SELECT * FROM publisher WHERE id = ?", array
    ( Sid ) );
    $this->id - Sdata['id'];
    $this->name • $data['name'];
  }

  function get_id( ) { return $this->id; }

  function set_id( $val ) { $this->id - $val; }

  function get_name( ) { return $this->name; }

  function set_name( Sval ) { $this->name = Sval; }

  function update( )
  {
    if ( Sthis->id !- null ) { $this->updateRecord( ); }
    else { Sthis->insertRecord( ); }
  }
  function insertRecord( )
  {
    return executeCommand( "INSERT INTO publisher ( id, name ) VALUES
    ( 0, ? )",
```

```
    array( $this->name ) );
  }

  function updateRecorcK )
  {
    return executeCommand( "UPDATE publisher SET name=? WHERE id=?",
    array( $this->name, $this->id ) );
  }
  function deleteRecord( $id )
  {
    return executeCommand( "DELETE FROM publisher WHERE id=?", array
    ( Sid ) );
  }
}

class Author
{
  var Sid;
  var $name;

  function Author( )
  {
    Sthis->id = null;
  }

  function load(Sid)
  {
    Sdata - selectOne( "SELECT * FROM author WHERE id - ?". array( Sid ) );
    Sthis->id = $data['id'];
    $this->name = $data['name']:
  }

  function get_id( ) { return Sthis->id; }

  function set_id( Sval ) { Sthis->id = Sval: }

  function get_name( ) { return Sthis->name; }

  function set_name( Sval ) { Sthis->name = Sval; }

  function update( )
  {
    if ( $this->id !- null ) { $this->updateRecord( ); }
    else { $this->insertRecord( ); }
  }

  function insertRecord( )
  {
    return executeCommand( "INSERT INTO author ( id, name ) VALUES
    ( 0, ? )".
    array( Sthis->name ) );
  }

  function updateRecord( )
  {
    return executeCommand( "UPDATE author SET name=? WHERE id=?",
    array( $this->name, Sthis->id ) );
```

```
  }

  function deleteRecord( $id )
  {
    return executeCommand( "DELETE FROM author WHERE id=?", array( $id ) );
  }
}
?>
```

Здесь находятся три класса, по одному для каждой таблицы из базы данных. У каждого есть поля-члены класса, соответствующие каждому полю из XML, конструктор, устанавливающий значение ID равным null, целый набор методов для доступа (get) и для изменения значений полей (set), а также функции для обновления и удаления записей. Для проверки этих классов запустите файл insert.php из командной строки:

```
% php insert.php
```

В нем в базу данных добавляется новый автор. Теперь вы можете запустить сценарий load.php:

```
% php load.php
Name: Jack
```

Результат подтверждает, что новая запись, как и ожидалось, была добавлена. И наконец, удалим запись при помощи сценария delete.php:

```
% php delete.php
```

Используя этот генератор вместе с другими описанными в данной книге, вы сможете формировать дублирующий исходный код для доступа к базе намного быстрее и аккуратнее, чем если бы вы делали это вручную.

Смотрите также

- «Разработка более качественных схем SQL» (смотрите трюк 34).
- «Создание неприступных баз данных» (смотрите трюк 35).
- «Формирование баз данных SQL» (смотрите трюк 41).
- «Формирование кода, выполняющего команду базы данных SELECT» (смотрите трюк 42).

ТРЮК №38

Упрощенная работа с XML при помощи регулярных выражений

При помощи трюка, использующего регулярные выражения, вы можете считывать XML, не затрачивая много времени на создание функций для его анализа.

Вы можете обрабатывать XML, используя всего несколько библиотек PHP. Например, поддержка XML — это всего лишь одна из возможностей, установленных или

нет на вашем сервере, на котором вы запускаете ваши приложения. Чтобы не полагаться на какие-то определенные расширения, зачастую намного проще и правильнее с точки зрения портативности обрабатывать данные из XML при помощи некоторых регулярных выражений, чем пользоваться специальными XML-анализаторами.

Код

Сохраните XML из примера 5.20 как файл books.xml.

Пример 5.20. Небольшой XML-код для примера

```
<books>
  <book name="Pragmatic Programmer" />
  <book name="Code Generation in Action" />
  <book id="8951234" name="Podcasting Hacks" />
</books>
```

Теперь сохраните исходный код из примера 5.21 в файл bookread.php.

Пример 5.21. Простой сценарий, использующий регулярные выражения для считывания XML

```
<?php
$xml = "";
while( !feof(STDIN) ) { $xml .= fgets( STDIN ); }
preg_match_all( "/\<book\s+.*?name=[\"|\']([.*?)[\"|\'].*?\/\>/is", $xml,
$found ):
foreach( $found[1] as $name ) { print( "$name\n" ); }
?>
```

В этом сценарии используется функция preg_match_all () для поиска тегов <book> в XML. Затем она группирует содержимое в поле name и возвращает результат. Модификатор s крайне важен — указав его, мы сообщаем процессору регулярных выражений о необходимости производить поиск в нескольких строчках. Нам подходят все варианты тега <book> из файла books.xml, поэтому это регулярное выражение должно быть достаточно гибким, чтобы обработать их.

Запуск трюка

Воспользуйтесь командой php для запуска файла bookread.php:

```
% php bookread.php < books.xml
Pragmatic Programmer
Code Generation in Action
Podcasting Hacks
```

Улучшаем трюк

В других довольно-таки часто встречающихся ситуациях мы можем располагать сложной вложенной структурой XML, которую нам необходимо обработать. Возьмем для примера список людей:

```
<people>
  <person>
```

```
    <first>Jack</first>
    <last>Herrington</last>
  </person>
  <person>
    <last>Katzen</last>
    <first>Molly</first>
  </person>
</people>
```

В идеале вам, возможно, захочется заполучить массив для каждого человека, состоящий из имени и фамилии. В примере 5.22 используется последовательный вызов регулярных выражений, чтобы сперва найти тег с описанием человека, а затем уже извлечь из этого тега имя и фамилию.

Пример 5.22. Регулярное выражение расправляется со сложным XML

```
<?php
$text - "";
while( !feof( STDIN ) ) { $text .- fgets( STDIN ); }
preg_match_all( "/\<person\>(.*?)\<\/person\>/si", Stext. $people );
$list = array( );
foreach( $people[1] as Sperson )
{
  preg_match( "/\<first\>(.*?)\<\/first\>/is". Sperson, $res ):
  $first = $res[1]:
  preg_match( "/\<last\>(.*?)\<\/last\>/is". Sperson, Sres );
  $last = $res[1];
  $list [ ] - array(
  'first' => $first,
  'last' => $last
  );
}
print_r( Slist ):
?>
```

Поскольку теги <first> и <last> могут располагаться в любой последовательности, этот сценарий более сложный, чем описанный в примере 5.21, использующий только одно регулярное выражение в попытке обнаружить одновременно теги <first> и <last>. Воспользовавшись PHP-интерпретатором для командной строки, выполните следующую команду:

```
php peopleread.php < people.xml
Array
(
  [0] -> Array
  (
    [first] => Jack
    [last] => Herrington
  )
  [1] -> Array
  (
    [first] -> Molly
    [last] -> Katzen
  )
)
```

Функция print_r() выводит содержимое массива в удобочитаемой для программиста форме. Таким образом, вы можете использовать полученную информацию,

как вам заблагорассудится, без необходимости полагаться на библиотеки расширений XML для PHP.

Смотрите также

LJ «Создание обработчиков простых XML-запросов для доступа к базам данных» (см. трюк 40).

• «Создание корректныхXML» (см. трюк 54).

ТРЮК №39

Экспорт схем баз данных в XML

Используйте PHP для считывания схемы из вашей базы данных и ее экспорта в XML для написания документации или для формирования кода.

Иногда по некоторым причинам бывает очень полезно иметь сохраненную схему вашей базы данных. Во-первых, вы можете воспользоваться ею при автоматическом формировании кода на PHP для доступа к базе данных (см. трюк 37). Во-вторых, вы можете использовать ее для сравнения двух версий схемы и разработки сценария, предназначенного для обновления приложения.

Код

В примере 5.23 показано содержимое файла schema.php.

Пример 5.23. Сценарий, который считывает схему базы данных в XML-формате

```
<?php
$dbuser = "root":
$dbpassword = "password':
$dbserver = " local host";
$dbname = "wordpress":

$db = mysql_connect( $dbserver, $dbuser, $dbpassword ):

mysql_select_db( $dbname );
$tables_res = mysql_query( "SHOW TABLES FROM ".$dbname, $db ):
$tables - array( );
while( $tableinfo = mysql_fetch_row($tables_res) ) {
  $tables[] - $tableinfo[ 0 ]:
}
mysql_free_result( $tables_res ):

header( "content-type: text/xml" );
?>
<schema>
<?php foreach( Stables as Stable ) { ?>
<table name="<?php echo( Stable ); ?>">
<?php
```

```
$fields_res = mysql_query( 'SHOW FIELDS FROM '.$table, $db );
while( $fieldinfo = mysql_fetch_row($fields_res) ) {
?>
<field
name="<?php echo( $fieldinfo[0]); ?>"
type="<?php echo( $fieldinfo[1]); ?>"
/>
<?php }
mysql_free_result( $fields_res );
?>
</table>
<?php } ?>
</schema>
```

В этом небольшом сценарии считывается схема из базы данных MySQL и формируется XML, который выводится на консоль (конечно же, вы можете сохранить его в файл). В сценарии сперва настраивается соединение с базой данных при помощи нескольких констант. Затем при использовании команды SHOW TABLES устанавливается связь с базой данных и получаются все используемые в ней таблицы. Далее перебираются все таблицы и при помощи команды SHOW FIELDS получается список всех полей для каждой из таблиц. Вся полученная информация сохраняется в XML, и в сценарии она на лету форматируется соответствующим образом.

Запуск трюка

Чтобы запустить этот сценарий, воспользуйтесь версией PHP-интерпретатора для командной строки:

```
% php schema.php
<schema>
<table name="wp_categories">
<field
name="cat_ID"
type="bigint(20)"
/>
<field
name="cat_name"
type="varchar(55)"
/>
<field
name="category_nicename"
type="varchar(200)"
/>
<field
name="category_description"
type="longtext"
/>
<field
name="category_parent"
type="int(4)"
/>
...
```

В этом примере я указал сценарию на мою базу данных под названием WordPress, которая является довольно сложной. В полученном в итоге XML есть базовый тег <schema>, содержащий теги <table> для каждой из таблиц. Внутри же каждого тега <table> перечислены все поля таблицы, причем для каждого из них используется индивидуальный тег <field>, в котором хранится имя поля и его тип.

ПРИМЕЧАНИЕ

Этот сценарий предполагается использовать только с MySQL. Запросы с точно такими же именами (SHOW TABLES и SHOW FIELDS) доступны и в других базах данных, но они работают слегка по-другому, однако вы можете внести некоторые минимальные изменения, чтобы этот сценарий смог работать с любой выбранной вами базой данных.

Смотрите также

- «Формирование кода, выполняющего команду базы данных CRUD» (смотрите трюк 37).
- «Формирование баз данных SQL» (смотрите трюк 41).
- «Формирование кода, выполняющего команду базы данных SELECT» (смотрите трюк 42).

ТРЮК № 40

Создание обработчиков простых XML-запросов для доступа к базам данных

XSLT может считывать данные XML прямо из URL. Описанный в этом трюке сценарий предназначен для быстрого переноса всей вашей базы данных из URL в формат XSLT.

XSLT — это мощный язык для представления отчетов или для преобразования данных. Он может считывать данные прямо из определенных URL. Тем не менее без установленных расширений изначально он не может работать с базами данных. Этот трюк предоставляет вам полный доступ к вашей базе данных через веб-сервер (отличная, хотя и довольно-таки сомнительная задумка). При помощи его указанные в URL запросы экспортируются в XML.

Код

Сохраните сценарий из примера 5.24 в файл query.php.

Пример 5.24. Небезопасный сценарий

```
<?php
$dbuser   =  "root";
$dbpassword = "password";
$dbserver  -  " localhost";
```

```
$dbname = "test";

$db = mysql_connect( $dbserver, $dbuser, $dbpassword );
mysql_select_db( $dbname );

$query = "SELECT * FROM user";
if ( $_GET["query"] )
  $query = $_GET["query"];

$res = mysql_query( $query, $db );

header( "content-type: text/xml" );
?>
<result>
<?php while( $row = mysql_fetch_assoc($res) ) { ?>
<row>
<?php foreach( $row as $key => $value ) { ?>
<data field="<?php echo( $key ); ?>"><?php echo( htmlentities( $value ) ); ?>
</data>
<?php } ?>
</row>
<?php } ?>
</result>
```

В этом простом и в то же время крайне небезопасном сценарии первым делом настраивается соединение с базой данных. Затем формируется SQL-запрос, который посылается базе данных.

ПРИМЕЧАНИЕ

Вы можете задать дополнительный параметр для SQL-запроса, добавив соответствующий параметр в URL.

Когда запрос послан, в качестве выходного файла генерируется XML, элементами которого выступают полученные из каждого ряда поля.

Запуск трюка

Скопируйте файл на ваш PHP-сервер и проверьте его, открыв URL в браузере.

ВНИМАНИЕ

Прежде чем запустить этот трюк, удостоверьтесь, что ваш клиент и сервер скрыты за брандмауэром, а также не доступны через Интернет. Иначе вы предоставите всему миру доступ к вашей базе данных!

Чтобы послать запрос, добавьте к URL соответствующий аргумент: **http://localhost/phphacks/xmlquery/query.php?query=SELECT%20*%20FROM%20user**. При помощи этого запроса вы полностью получаете таблицу с пользователями из моей тестовой базы данных. XML, полученный в результате, выглядит следующим образом:

```
<result>
<row>
<data field="id">1</data>
```

```
<data field="name">jack</data>
</row>
<row>
<data field="id">2</data>
<data field="name">lori</data>
</row>
<row>
<data field="id">3</data>
<data field="name">megan</data>
</row>
</result>
```

В браузере Internet Explorer отобразится отформатированный XML. В других браузерах вам, возможно, придется просмотреть исходный код полученной в результате запроса страницы. Теперь вы можете указать таблице стилей на этот же URL и получить доступ через XML к любой информации, которая вас интересует.

Смотрите также

J «Возможность использования вашими клиентами контроля над форматированием при помощи XSL» (см. трюк 7).

Формирование баз данных SQL

Используйте PHP для автоматического создания сценариев SQL с помощью схемы базы данных в формате XML.

Одной из наиболее распространенных проблем, с которой сталкиваются разработчики при написании исходного кода для доступа к базе данных, является отсутствие синхронизации PHP со структурой базы данных и т. д. Обычно мы используем сценарий SQL для предварительной загрузки базы данных с соответствующими таблицами и данными в них с целью дальнейшей загрузки приложения. Но поддержка работоспособности этого сценария может стать настоящей головной болью, особенно если одновременно с этим вам требуется делать соответствующие обновления в коде на PHP, связанном с таблицами SQL. В этом трюке рассматривается простой сценарий, предназначенный для автоматического формирования SQL и PHP на базе описания базы данных в XML-формате. Это позволит вам быть уверенными, что SQL и PHP синхронизированы. Это также означает, что в случае изменений сервера базы данных или версии PHP вы все еще сможете использовать тот же самый файл `schema.xml`. Все, что вам надо будет сделать, — это изменить исходный код генератора, чтобы он соответствовал текущей версии сервера или версии PHP. На рис. 5.5 показано программное взаимодействие с файлом `schema.xml`, содержимое которого используется в качестве входной информации в код генератора, написанного на PHP и создающего файл MySQL, который, в свою очередь, создает базу данных.

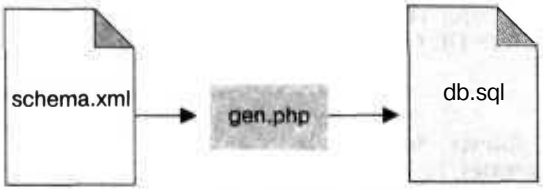


Рис. 5.5. Генератор, создающий SQL из XML-файла со схемой

Код

Сохраните простой XML-документ из примера 5.25, описывающий схему базы данных в файл с именем `schema.xml`.

Пример 5.25. Структура базы данных в XML-формате

```
<schema>
<table name="book">
  <field name="id" type="int" primary-key="true" />
  <field name="title" type="text" />
  <field name="publisher_id" type="int" />
  <field name="author_id" type="int" />
</table>
<table name="publisher">
  <field name-"id" type="int" primary-key="true" />
  <field name="name" type="text" />
</table>
<table name="author">
  <field name="id" type-"int" primary-key="true" />
  <field name="name" type="text" />
</table>
</schema>
```

Сохраните код генератора из примера 5.26 в файл `gen.php`.

Пример 5.26. Сценарий для автоматического формирования базы данных и исходного кода

```
<?php
Stables = array( );

function start_element( Sparser, $name. $attribs )
{
  global Stables;
  if ( $name — "TABLE" )
  {
    Stable - array( ):
    $fields = array( );
    $table['name'] = $attribs['NAME'];
    $table['fields'] - array( ):
    Stables []= Stable;
  }
  if ( Sname -- "FIELD" )
  {
    $field = array( );
    $field['name'] = $attribs['NAME'];
    $field['type'] = $attribs['TYPE'];
```

```
    $field['pk'] = ( $attribs['PRIMARY-KEY'] == "true" ) ? 1 : 0;
    $tables[count($tables)-1]['fields'] []= $field;
  }
}

function end_element( Sparser, $name ) { }
Sparser = xml_parser_create( );
xml_set_element_handler($parser, "start_element", "end_element" );
while( !feof( STDIN ) ) {
  Stext = fgets( STDIN );
  xml_parse( Sparser, Stext );
}
xml_parser_free( Sparser );
ob_start( );
foreach( Stables as Stable ) {
  $pk = null;
?>
DROP TABLE IF EXISTS <?php echo( $table['name'] ) ?>;
CREATE TABLE <?php echo( $table['name'] ) ?> (
<?php
$first - 1;
foreach( $table['fields'] as Sfield ) {
?>
<?php echo( Sfirst ? "" : "," ) ?>
<?php echo( $field['name'] ) ?> <?php echo( Sfield['type'] ) ?>
<?php if ( Sfield['pk'] ) {
Spk = $field['name'];
?> NOT NULL AUTO_INCREMENT<?php } ?>
<?php
Sfirst = 0;
} ?>
<?php if ( Spk ) { ?>
,primary key( <?php echo( Spk ) ?> )
<?php } ?>
);
<?php }
$sql = ob_get_clean( );
$fh - fopen( "db.sql", "w" );
fwrite( $fh, Ssql );
fclose( Sfh );
?>
```

Запуск трюка

Воспользуйтесь PHP-интерпретатором для командной строки, чтобы запустить исходный код:

```
php gen.php < schema.xml
```

В сценарии создается файл db.sql с примерно следующим содержимым (очевидно, что полученные вами в результате файлы будут отличаться в случае различных баз данных и таблиц):

```
DROP TABLE IF EXISTS book;
CREATE TABLE book (
  id int NOT NULL AUTO_INCREMENT
```

```
  .title text
  .publisher_id int
  .author_id int
  .primary key( id )
);

DROP TABLE IF EXISTS publisher;
CREATE TABLE publisher (
  id int NOT NULL AUTO_INCREMENT
  .name text
  .primary key( id )
);

DROP TABLE IF EXISTS author;
CREATE TABLE author (
  id int NOT NULL AUTO_INCREMENT
  .name text
  .primary key( id )
);
```

Этот SQL-код создает таблицу, соответствующую описанной в файле `schema.xml` схеме. В XML-файле хранятся все таблицы и их поля, и он может использоваться как для формирования SQL (см. трюк 42), так и кода на PHP, использующего этот SQL (см. трюк 37). Именно поэтому SQL и PHP всегда будут синхронизированы.

ПРИМЕЧАНИЕ

Никогда не вносите изменения в файл db.sql вручную. Всегда после внесения изменений в XML-файл, содержащий схему, запускайте генератор для создания нового SQL.

Смотрите также

- «Формирование кода, выполняющего команду базы данных CRUD» (смотрите трюк 37).
- «Формирование кода, выполняющего команду базы данных SELECT» (смотрите трюк 42).

Формирование кода, выполняющего команду базы данных SELECT

Используйте PHP для написания исходного кода, предназначенного для доступа к базе **данных** прямо из XML с описанием ее схемы.

Написание классов для доступа к SQL-таблицам баз данных — очень утомительная и подверженная ошибкам работа. В этом трюке при помощи XML-файла с описанной схемой базы данных и написанного на PHP генератора кода PHP-классы создаются автоматически.

ПРИМЕЧАНИЕ

Я использовал тот же файл schema.xml, что и в трюке для формирования соответствующего SQL (см. трюк 41).

На рис. 5.4 продемонстрировано, как в качестве входной информации в генератор передается XML. Генератор же в свою очередь создает классы на PHP и сохраняет их в файл mydb.php.

ПРИМЕЧАНИЕ

Этот выходной файл является временным, и вносить в него изменения вручную не следует.

Код

В примере 5.27 показано содержимое файла schema.xml, отражающего схему базы данных.

Пример 5.27. XML, отражающий схему базы данных

```
<schema>
<table name="book">
  <field name="id" type="int" primary-key="true" />
  <field name="title" type="text" />
  <field name="publisher_id" type="int" />
  <field name="author_id" type="int" />
</table>
<table name="publisher">
  <f1eld name="id" type="int" primary-key="true" />
  <field name="name" type="text" />
</table>
<table name="author">
  <field name="id" type="int" primary-key="true" />
  <field name="name" type="text" />
</table>
</schema>
```

По аналогии с тем, как мы создавали код для реализации команды CRUD (см. трюк 37), вам необходимо будет создать оберточные функции. Сохраните исходный код из примера 5.28 в файл dbwrap.php.

Пример 5.28. Функции, предоставляющие доступ к информации, хранимой в определенной базе данных

```
<?php
require_once( "DB.php" ):
$dsn - 'mysql://root:password@localhost/books':
$db =& DB::Connect( $dsn, array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )); }

function selectOne( $sql, $args )
{
  global $db;
         $res = $db->query( $sql. $args ):
         $res->fetchInto($row, DB_FETCHMODE_ASSOC);
  return $row;
```

```
}

function selectBlock( $sql, $args )
{
  global $db;
         $res = $db->query( $sql, $args );
         $rows = array( );
  while( $res->fetchInto($row, DB_FETCHMODE_ASSOC) ) { $rows []= $row; }
  return $rows;
}
?>
```

Далее сохраните исходный код из примера 5.29 в файл gen.php. Здесь на самом деле и происходит автоматическое формирование кода.

Пример 5.29. Сценарий, формирующий исходный код

```
<?php
$tables = array( );

function start_element( Sparser, Sname, Sattribs )
{
  global Stables;
  if ( Sname — "TABLE" )
  {
    $table = array( );
    $fields = array( );
    Stablername'] = $attribs['NAME'];
    $table['fields'] - array( ):
    Stables []= Stable;
  }
  if ( $name == "FIELD" )
  {
    Sfield = array( );
    $field['name'] = SattribsC'NAME'];
    $field['type'] = $attribs['TYPE'];
    $field['pk'] = ( $attribs['PRIMARY-KEY'] == "true" ) ? 1 : 0;
    $tables[count($tables)-1]['fields'] []= Sfield;
  }
}

function end_element( Sparser, Sname ) { }
Sparser = xml_parser_create( );
xml_set_element_handler($parser, "start_element", "end_element" );
while( !feof( STDIN ) ) {
  Stext = fgets( STDIN );
  xml_parse( Sparser, Stext );
}
xml_parser_free( Sparser );
ob_start( );
echo( "<?php\n" );
?>

require_once( "dbwrap.php" );

<?php

foreach( Stables as Stable ) {
```

```
  $pk = null;
  foreach( $table['fields'] as $field ) {
    if ( $field['pk'] )
      $pk = $field['name'];
  }
?>
class <?php echo( ucfirst( $table['name'] ) ) ?>
{
  function getOne( $id )
  {
    return selectOne( "SELECT * FROM <?php echo( $table['name'] ) ?> WHERE
    <?php echo( $pk ); ?> = ?". array( $id ) );
  }
  function getAll( )
  {
    return selectBlock( "SELECT * FROM <?php echo( $table['name'] ) ?>".
    array( )
  }
}
<?php }
echo( "?>" );

$php - ob_get_clean( );

$fh = fopen( "mydb.php", "w" );
fwrite( $fh. $php );
fclose( $fh );
?>
```

Сперва код, отвечающий за генерацию, считывает содержимое XML в структуру, хранимую в памяти. Затем при помощи стандартных механизмов использования шаблонов начинают формироваться классы для каждой из таблиц, и выходная информация буферизуется в строке. Когда весь необходимый для классов исходный код создан, буферизация прекращается, вся полученная выходная информация сохраняется в строке, которая, в свою очередь, записывается в файл при помощи функций fopenO, `fwrite()` и fcloseO.

Для запуска генератора, использующего XML-файл со схемой базы данных, воспользуйтесь PHP-интерпретатором для командной строки:

```
php gen.php < schema.xml
```

В результате вы получите в той же директории файл `mydb.php` примерно следующего содержания:

```
<?php
require_once( "dbwrap.php" );

class Book
{
  function getOne( $id )
  {
    return selectOne( "SELECT * FROM book WHERE id = ?", array( $id ) );
  }
  function getAll( )
```

```
  {
    return selectBlock( "SELECT * FROM book". array( ) );
  }
}

class Publisher
{
  function getOne( $id )
  {
    return selectOne( "SELECT * FROM publisher WHERE id - ?". array( $id )
    );
  }
  function getAll( )
  {
    return selectBlock( "SELECT * FROM publisher". array( ) );
}

class Author
{
  function getOne( $id )
  {
    return selectOne( "SELECT * FROM author WHERE id - ?". array( Sid ) );
  }
  function getAll( )
  {
    return selectBlock( "SELECT * FROM author". array( ) ):
  }
)
?>
```

Согласитесь, круто! PHP создает PHP!

В сценарии сперва анализируется схема, а затем при помощи довольно простого кода на PHP формируется исходный код, использующий функции из файла dbwrap.php, предназначенные для получения информации из базы данных. У каждого класса есть две функции: одна получает все элементы таблицы, а другая — только одну запись из таблицы. Чтобы проверить это, создайте файл index.php (содержимое которого показано в примере 5.30) и воспользуйтесь им для создания объекта типа Publisher, предназначенного для отправки запросов в таблицу publisher (издатель).

Пример 5.30. Сценарий для проверки кода, работающего с базой данных

```
<?php
require_once( "mydb.php" );
$pub = new Publisher( ):
?>
<html>
<body>
<table>
<?php
$rows = $pub->getAll( );
foreach( $rows as $row ) {
?>
```

```
<tr><td><?php echo( $row['id'] ); ?></td>
<td><?php echo( $row['name'] ); ?></td></tr>
<?php } ?>
</table>
</body>
</html>
```

Откройте созданную вами страницу в браузере, и вы увидите содержимое таблицы publisher. На рис. 5.6 вы можете увидеть эту таблицу, отображенную в браузере.

Рис. 5.6. Таблица publisher, отображенная в браузере

Смотрите также

- «Формирование кода, выполняющего команду базы данных CRUD» (смотрите трюк 37).
- ❑ «Формирование баз данных SQL» (смотрите трюк 41).

ТРЮК №43

Преобразование CSV в PHP

Используйте PHP для создания массивов данных на базе файлов данных со значениями, разделенными запятыми (CSV).

Я частенько работаю со списком статических значений, который мне не хочется размещать в базе данных, но который в то же время я хотел бы использовать в своих PHP-приложениях. Эти статические данные могут храниться где угодно, но чаще

всего они содержатся в таблицах. В этом полезном трюке реализуется возможность преобразования любой информации, представленной в виде значений, разделенных запятыми (один из наиболее простых форматов для извлечения данных из таблицы), в код на PHP, который в дальнейшем вы сможете скопировать и вставить в вашу PHP-станицу.

Код

Сохраните исходный код из примера 5.31 в файл index.php.

Пример 5.31. Страница с требующими преобразования данными, разделенными запятыми

```
<html>
<body>
<form method="post" action="commaconv.php" />
<table>
<tr><td>CSV Data:</td>
<td><textarea name="data" cols="40" rows="10">
"Alabama",4530182
"Alaska",655435
"Arizona",5743834
"Arkansas",2752629
"California",35893799
"Colorado",4601403
"Connecticut",3503604
"Delaware",830364
"District of Columbia",553523
"Florida",17397161
"Georgia",8829383
"Hawaii",1262840
"Idaho",1393262
"Illinois",12713634
"Indiana",6237569
"Iowa",2954451
"Kansas",2735502
"Kentucky",4145922
"Louisiana",4515770
"Maine",1317253
"Maryland",5558058
"Massachusetts",6416505
"Michigan",10112620
"Minnesota",5100958
"Mississippi",2902966
"Missouri",5754618
"Montana",926865
"Nebraska",1747214
"Nevada",2334771
"New Hampshire",1299500
"New Jersey",8698879
"New Mexico",1903289
"New York",19227088
"North Carolina",8541221
"North Dakota",634366
"Ohio",11459011
```

```
"Oklahoma",3523553
"Oregon",3594586
"Pennsylvania",12406292
"Rhode Island",1080632
"South Carolina",4198068
"South Dakota",770883
"Tennessee",5900962
"Texas",22490022
"Utah",2389039
"Vermont",621394
"Virginia",7459827
"Washington",6203788
"West Virginia",1815354
"Wisconsin",5509026
"Wyoming",506529</texta rea></td></tr>
<tr><td> Field Name 1:</td><td><input name="field0" value="state" /></td></tr>
<tr><td> Field Name 2:</td><td><input name="field1" value="population" /></td>
</tr>
<tr><td> Field Name 3:</td><td><input name="field2" value="" /></td></tr>
<tr><td> Field Name 4:</td><td><input name="field3" value="" /></td></tr>
<tr><td> Field Name 5:</td><td><input name="field4" value="" /></td></tr>
</table>
<input type="submit" />
</form>
</body>
</html>
```

Исходный код из примера 5.32, который требуется сохранить в файл commaconv. php, реализует саму возможность преобразования данных.

Пример 5.32. Код на PHP, с легкостью преобразующий данные из CSV-формата, как и в случае с XML или SQL

```
<html><body>
<div style="font-family:courier: font-size:small;">
$data - array(<br/>
<?
$fieldnames = array(
  $_POST['field0' ].
  $_POST['field1' ],
  $_POST['field2' ],
  $_P0ST['field3' ],
  $_POST['field4' ] ):
  $rows = split( "\n". $_POST['data'] ):
  $index = 0:
  foreach( $rows as $row )
  {
    if ( $index != 0 )
      printC ".<br/>" );
    $index++:
    print( " array(" );
    $fields = split( ".". $row );
    for( $f = 0: $f < count( $fields ): $f++ )
    {
      $data = $fields[ $f ];
      $data = preg_replace( "/WWW". "\"". $data );
      if ( $f > 0 )
```

```
        print( ". " );
      print( $fieldnames[ $f ] );
      printC " => " );
      print( $data );
    }
    print( " )" );
  }
  ?><br/>
);
</div>
</body></html>
```

Запуск трюка

Первым делом откройте предварительно сохраненную на сервере страницу index.php. Вы увидите показанную на рис. 5.7 форму. Вставьте данные в CSV-формате в поле ввода **CSV Data** (или оставьте уже введенную туда по умолчанию информацию). Затем введите имена полей, находящихся в данных, в поля **Field Name,** расположенные ниже.

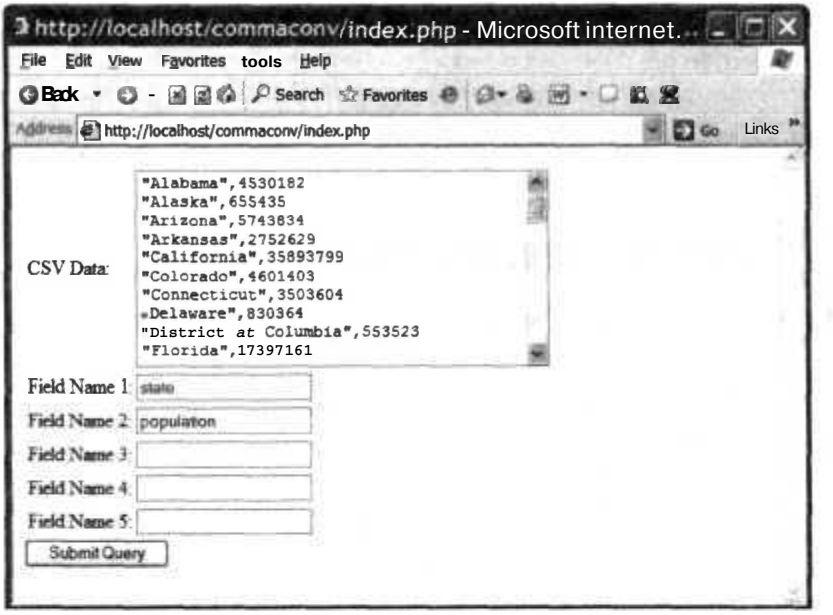


Рис. 5.7. Форма ввода данных в CSV-формате с именами полей

ПРИМЕЧАНИЕ

На данный момент у страницы есть только пять полей ввода, но увеличить их количество в HTML — крайне простая задача.

Когда вы введете данные и имена колонок, нажмите кнопку Submit Query, и форма передаст эти данные в сценарий commaconv.php. Результат работы сценария

показан на рис. 5.8. Выделите весь отображенный на странице текст и воспользуйтесь командой копирования, чтобы поместить его в буфер обмена. Затем вы можете воспользоваться командой вставки в вашем любимом текстовом редакторе, чтобы разместить полученный код на вашей странице.

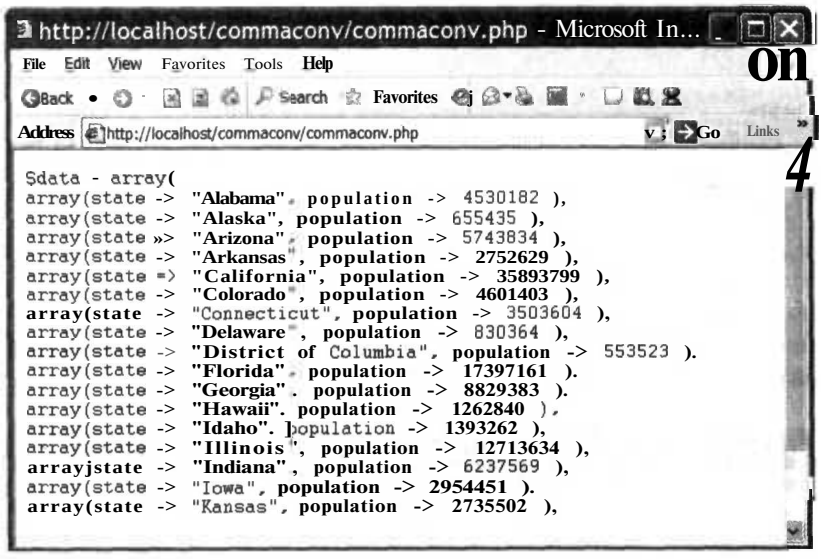


Рис. 5.8. Сгенерированный PHP-сценарий

ТРЮК №44

Импорт данных с веб-страниц

Используйте регулярные выражения для импорта данных из таких источников, как Metacritic.

Как вы поступите, если вам необходимы некоторые данные с сайта, но он не поддерживает их экспорт в каком-либо понятном формате (например, в XML (см. трюк 38) или CSV (см. трюк 43))? Одним из вариантов решения этой задачи является извлечение данных при помощи обработки отображенной на экране информации (screen scrape) прямо из HTML. Для начала содержимое страницы с интересующей нас информацией загружается либо в строковую переменную, хранимую в памяти, либо в файл. Затем из этой строки или файла при помощи регулярных выражений извлекаются соответствующие данные.

Вы можете обработать таким образом практически любой веб-сайт и получить из него необходимые данные. Для примера в этом трюке я выбрал страницу с обзорами DVD Metacritic (http://www.metacritic.com/video/).

Metacritic — это сайт, где размещаются обзоры и оценки фильмов, музыки и видеоигр. На рис. 5.9 показана страница Metacritic, которую я использовал для импорта данных для трюка.

Рис. 5.9. Страница Metacritic с представленными на ней обзорами DVD и видео

В левой части окна находится отсортированный по названиям список фильмов с оценками. Исходя из размера страницы я могу сказать, что мне нужна лишь небольшая ее HTML-часть. Я использую команду View Source (Просмотр исходного кода страницы), чтобы изучить исходный код страницы, и вижу, что на самом деле есть отдельная небольшая секция для оценок, выделенная в тег , в котором-то и содержатся нужные мне данные.

```
</TR>
</TABLE>
<DIV ID="sortbyname1">
<P CLASS="listing">
<SPAN CLASS="yellow">51</SPAN>
<A HREF="/video/titles/800bullets">800 Bullets</A><BR>
<SPAN CLASS="yellow">58</SPAN>
<A HREF="/video/titles/actsofworship">Acts of Worship</A><BR>
<SPAN CLASS="green">81</SPAN>
<A HREF="/video/titles/badeducation"><B>Bad Education</B></A><IMG
SRC="/_images/scores/star.gif" WIDTH="11" HEIGHT="11" ALIGN="absmiddle"><BR>
...
```

Первым делом мы извлечем именно этот тег . Затем нам понадобится воспользоваться другим регулярным выражением, чтобы выбрать запись о каждом фильме из текста, находящегося в теге . Заметьте, что перечисление каждого фильма начинается с тега и заканчивается тегом
. Этого вполне

достаточно, чтобы выделить каждый фильм. В третьем списке содержится некоторая дополнительная информация о фильме, которую я извлекаю при помощи других регулярных выражений. Я очень рекомендую вам использовать технику «разделяй-и-властвуй» при написании кода для извлечения данных. Не пытайтесь проделать всю работу при помощи одного регулярного выражения, иначе это приведет к тому, что вы создадите абсолютно нечитаемый код, внести изменения в который не сможете даже сами.

Код

Сохраните исходный код из примера 5.33 в файл scrapecritic.php.

Пример 5.33. Исходный код на PHP для загрузки URL и получения содержимого из него

```
<html>
<?
// Инициализируем объект типа CURL
$ch - curl_init( "http://www.metacritic.com/video/" ):

// Подделываем пользовательский программный агент
curl_setopt( $ch. CURLOPT_USERAGENT, "Internet Explorer" );

// Начинаем буферизацию
ob_start( ):

// Получаем HTML из MetaCritic
curl_exec( $ch ):
curl_close( $ch ):

// Получаем содержимое выходного буфера
$str - ob_get_contents( );
ob_end_clean( );

// Получаем отсортированный по именам список
preg_match( '7\<DIV ID=\"sortbyname1\"\>(.*?)\<\/DIV\>/is".
$str. $byname ):

// Получаем каждый фильм отдельно
preg_match_all( "/\<SPAN.*?>(.*?)\<\/SPAN\>.*?\<A.*?\>(.*?)\<BR\>/is",
$byname[0], $moviedata );

// Обрабатываем данные о фильмах
$movies = array( );
for( $i = 0; $i < count( $moviedata[1] ); $i++ )
{
  // Оценки готовы
  $score = $moviedata[1][$i];
  // Нам необходимо очистить заголовок от тегов
  // и декодировать HTML
  $title - $moviedata[2][$i]:
  $title = preg_replace( "/<.*?>/", "". $title ):
  $title = html_entity_decode( Stitle );
  // Список фильмов добавляется в массив
```

```
  $movies [ ] - array( $score. $title );
}
?>
<body>
<table>
<tr>
<th>Name</th><th>Score</th>
</tr>
<? foreach( $movies as $movie ) { ?>
<tr>
<td><? echo( $movie[1] ) ?></td>
<td><? echo( $movie[0] ) ?></td>
</tr>
<? } ?>
</table>
</body>
</html>
```

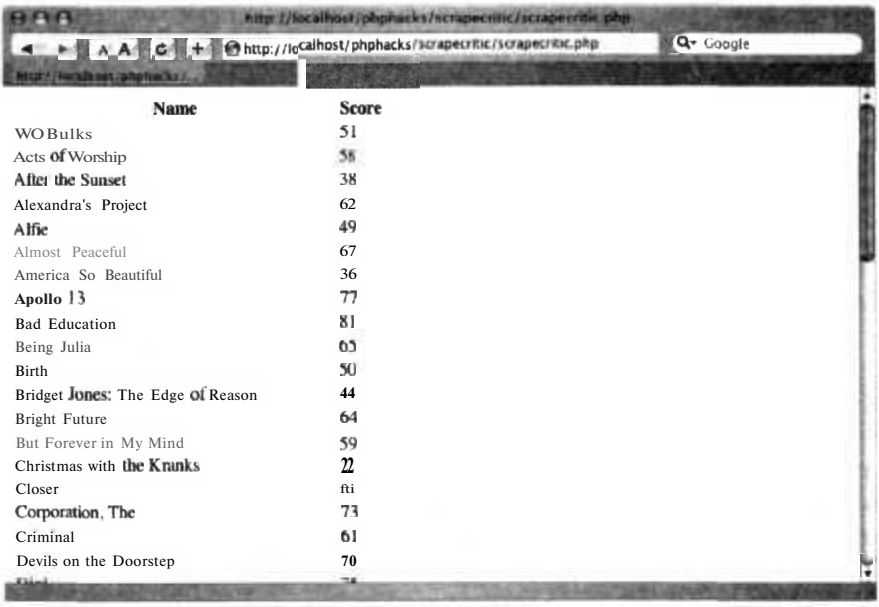
Для начала в сценарии `scrapecritic.php` содержимое страницы Metacritic DVD сохраняется в строке. Функции `ob_start()`, ob_get_contents() и `ob_end_clean()` используются для перехвата текста, который функция `curl_exec()` разместила бы на странице, и вместо этого копируют его в строку. Следующим шагом сценария при помощи функции `preg_match()` и регулярного выражения для конкретной страницы извлекается содержимое только тега , в котором находится отсортированный по именам список. Здесь-то и проявляется основная техническая проблема извлечения данных: если на сайте, который вы обрабатываете, хотя бы самую малость изменится форматирование, это может привести (и, скорее всего, приведет) к неработоспособности кода для импорта данных. Если это возможно, следует всегда размещать данные в XML, так как он более гибок в случае изменения формата. Получив отсортированный по именам список, сценарий использует функцию `preg_match_all` () для получения названий фильмов и их оценок и размещения их в массиве. Ну и, наконец, сценарий получает названия фильмов из этого массива, освобождая записанные в него имена от каких-либо дополнительных ненужных тегов и форматирования. В итоге мы получаем очищенные от избыточной информации данные, готовые к дальнейшему использованию. При помощи простого цикла `foreach` в сценарии создается таблица, в которой содержатся названия фильмов и оценки, соответствующие им с учетом обзоров.

Запуск трюка

Чтобы запустить этот трюк, скопируйте файл на ваш PHP-сервер и откройте его в браузере (рис. 5.10).

ПРИМЕЧАНИЕ

Импорт данных из отображенной на экране информации также используется для преобразования типа содержимого. Вы можете взять содержимое HTML-страницы и преобразовать ее в WML-формат для отображения в телефонах с поддержкой Интернета или в формат RSS-полей для формирования новостей.



Name	Score
WO Bulks	51
Acts of Worship	58
After the Sunset	38
Alexandra's Project	62
Alfie	49
Almost Peaceful	67
America So Beautiful	36
Apollo 13	77
Bad Education	81
Being Julia	63
Birth	50
Bridget Jones: The Edge of Reason	44
Bright Future	64
But Forever in My Mind	59
Christmas with the Kranks	22
Closer	fti
Corporation, The	73
Criminal	61
Devils on the Doorstep	70

Рис. 5.10. Результат извлечения данных из страницы

Проблемы, возникающие при извлечении информации

При импорте данных чаще всего возникают две основные проблемы. Первая технического плана, а вторая — правового. С технической стороны, при извлечении данных часто возникают ошибки, когда изменяется формат обрабатываемого вами сайта. В дополнение ко всему исходный код для импорта данных с одного сайта, очевидно, не сможет работать с другими по причине различий в форматировании. И, наконец, процесс извлечения информации может быть крайне медленным или даже прерываться, когда обрабатываемый вами сайт не отвечает на запросы по истечении определенного промежутка времени. Следует также с умом подходить к выбору сайта, данные из которого вы хотите импортировать. Ищите сайты, сгенерированные при помощи веб-приложений, а не написанные вручную.

Разметка созданных самостоятельно страниц может размещаться в коде абсолютно случайно, в то время как в сгенерированных при помощи приложений страницах она будет представлена в предсказуемом формате, что позволит намного проще работать с ней при помощи регулярных выражений соответствующего формата.

ПРИМЕЧАНИЕ

Сгенерированные приложениями страницы, как правило, имеют расширения PHP, JSP, ASP или созвучные с ними. Созданные же вручную страницы обычно имеют расширения HTM или HTML.

С правовой же точки зрения, прежде чем вы добавите информацию себе на сайт, вам следует заранее удостовериться, имеете ли вы право таким образом ее использовать. Нет ничего хуже, чем узнать после написания огромного кода, предназна-

ченного для импорта данных, что информация была получена неправомерно и не может быть использована.

Смотрите также

- «Проверка вашего приложения при помощи роботов» (см. трюк 83).
- «Следите за вашим сайтом» (см. трюк 84).

ТРЮК №45

Получение данных из загруженных таблиц Excel

Используя XML из Excel 2003, вы можете получать данные прямо из таблиц, которые пользователи загрузили на ваш сайт.

Вы можете получать информацию от ваших пользователей, используя множество различных способов. Если вы позволите пользователям легко добавлять информацию в вашу систему, то избежите того, что они уйдут от вас для поиска новых ресурсов, позволяющих удовлетворить их потребности по работе с данными, необходимыми для бизнеса (и, естественно, вы лишитесь доходов от них). Поддержка импорта данных с таких общедоступных источников, как Excel, может быть очень привлекательным для пользователей. В этом трюке будет показано, как сохранять таблицы Excel в новый формат XML, поддерживаемый как Excel, так и Microsoft Office 2003, а также как считывать данные с этого формата и выводить пользователю на экран. На рис. 5.11 показано взаимодействие между браузером (отображенном на рисунке в виде компьютера) и системой импорта данных. Первая страница — index.php, на которой расположена кнопка Browse (Обзор). Затем пользователь выбирает XML-файл Excel, который передается на страницу import.php, а она, в свою очередь, возвращает готовый HTML с данными, хранимыми в выбранном файле.

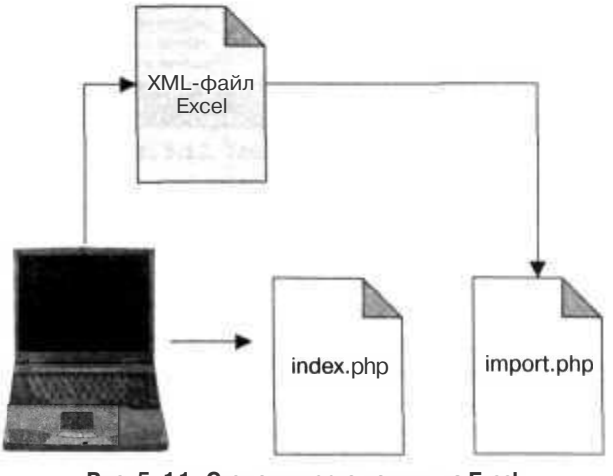


Рис. 5.11. Схема импорта данных из Excel

КОД

Файл index.php (показанный в примере 5.34) отвечает за передачу данных из Excel в ваши PHP-сценарии.

Пример 5.34. Код на PHP для получения данных из Excel в ваши сценарии

```
<html>
<body>
<form enctype="multipart/form-data" action="import.php" method="post">
Excel XML file:
  <input type="hidden" name="MAX_FILE_SIZE" value="2000000" />
  <input type="file" name="file" /><br/>
  <input type="submit" value="Upload" />
</form>
</body>
</html>
```

Сохраните исходный код из примера 5.35 в файл import.php. Он предназначен для импорта данных.

Пример 5.35. PHP, предназначенный для импорта данных из Excel

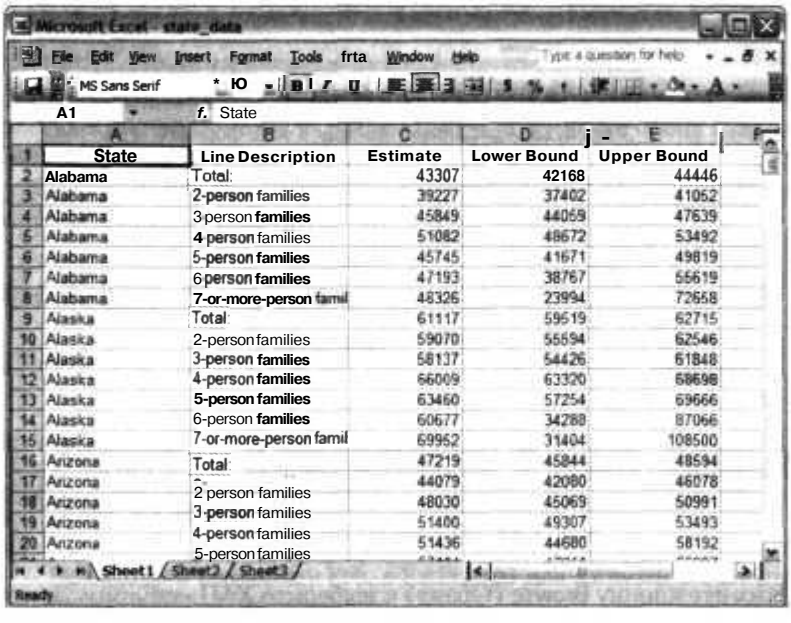
```
<html>
<body>
<?php
$data = array( ):
if ( $_FILES['file']['tmp_name'] )
{
  $dom = DOMDocument::load( $_FILES['file']['tmp_name'] );
  $rows - $dom->getElementsByTagName( 'Row' ):
  foreach ($rows as $row)
  {
    $cells - $row->getElementsByTagName( 'Cell' ):
    $datarow = array( );
    foreach ($cells as $cell)
    {
      $datarow [ ]- $cell->nodeValue:
    }
    Sdata [ ]- Sdatarow:
  }
}
?>
<table>
<?php foreach( Sdata as $row ) { ?>
<tr>
<?php foreach( $row as $item ) { ?>
<td><?php echo( $item ); ?></td>
<?php } ?>
</tr>
<?php } ?>
</table>
</body>
</html>
```

Страница import.php, которая является основной в этом трюке, первым делом открывает загруженный файл при помощи XML DOM. Затем она перебирает все

элементы Row, а для каждого из них — все элементы Cell. В каждом элементе Cell сценарий находит текущую дату, которая сохраняется в массив с именем $datarow. Затем сохраненная информация выводится на экран в виде HTML с использованиемя стандартной техники текстовых шаблонов PHP (в конце сценария).

Запуск трюка

Для запуска данного трюка сперва создадим таблицу Excel. Я, как обычно, посетил в Интернете сайт U.S. Census Bureau **(http://www.census.gov),** чтобы получить некоторые данные. В данном конкретном случае я использовал средний семейный доход, расположенный в отдельных скобках для каждого из 50 штатов. (Кстати, среди семей, состоящих из двух человек, самый большой средний доход у тех, которые живут в штате Аляска, так что дерзайте!) Тем не менее данные вы можете видеть на рис. 5.12.



	A	B	C	D	E
1	State	Line Description	Estimate	Lower Bound	Upper Bound
2	Alabama	Total:	43307	42168	44446
3	Alabama	2-person families	39227	37402	41052
4	Alabama	3-person families	45849	44059	47639
5	Alabama	4-person families	51082	48672	53492
6	Alabama	5-person families	45745	41671	49819
7	Alabama	6-person families	47193	38767	55619
8	Alabama	7-or-more-person famil	48326	23994	72658
9	Alaska	Total:	61117	59519	62715
10	Alaska	2-person families	59070	55594	62546
11	Alaska	3-person families	58137	54426	61848
12	Alaska	4-person families	66009	63320	68698
13	Alaska	5-person families	63460	57254	69666
14	Alaska	6-person families	60677	34288	87066
15	Alaska	7-or-more-person famil	69952	31404	108500
16	Arizona	Total:	47219	45844	48594
17	Arizona	2-person families	44079	42080	46078
18	Arizona	3-person families	48030	45069	50991
19	Arizona	4-person families	51400	49307	53493
20	Arizona	5-person families	51436	44680	58192

Рис. 5.12. Таблица Excel с данными

Теперь мне необходимо преобразовать ее в таблицу XML-формата, выбрав команду Save As (Сохранить как), как это показано на рис. 5.13. Интересно отметить, что я могу всегда хранить таблицу в XML-формате — нет зависимости между бинарной версией и версией XML, а также мы не столкнемся с потерей данных. Если ваш клиент использует Office 2003 и вы предоставляете ему таблицу для работы, просто дайте ему ее XML-версию! В этом случае он сможет продолжить работать с ней и никогда даже не узнает о каких-либо различиях. Теперь, получив XML, мы можем загрузить страницы для нашего сайта; используя браузер, откройте в нем файл index.php **(рис. 5.14).**

Рис. 5.13. Сохраняем таблицу в XML-формате

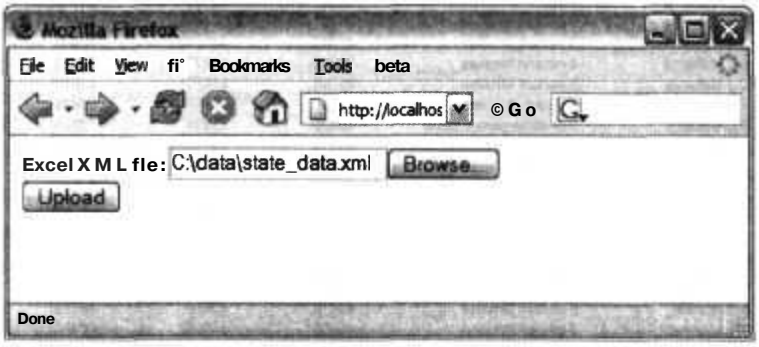
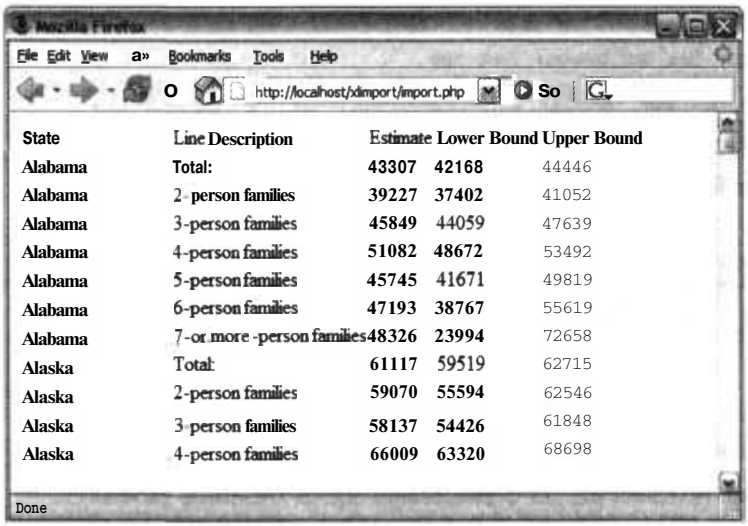


Рис. 5.14. Загрузка таблицы на сервер

Теперь нажмите кнопку **Browse** (Обзор) и выберите XML-таблицу. Затем нажмите кнопку **Upload** (Загрузить) и передайте таким образом XML-данные в файл `import.php`. В нем используются поддерживаемые PHP-объекты типа XML DOM и Xpath, используемые для обработки XML и извлечения из него табличных данных. Затем при помощи небольшого кода на PHP эти данные представляются в виде HTML-таблицы. Результат работы с взятыми для примера статистическими данными, которые я изобразил на рис. 5.12, показан на рис. 5.15.

ПРИМЕЧАНИЕ

Очевидно, если вы хотите разместить данные в базе данных, то следует внести в исходный код соответствующие изменения и не формировать HTML, а вызвать команду SQL INSERT.



State	Line Description	Estimate	Lower Bound	Upper Bound
Alabama	Total:	43307	42168	44446
Alabama	2-person families	39227	37402	41052
Alabama	3-person families	45849	44059	47639
Alabama	4-person families	51082	48672	53492
Alabama	5-person families	45745	41671	49819
Alabama	6-person families	47193	38767	55619
Alabama	7-or more -person families	48326	23994	72658
Alaska	Total:	61117	59519	62715
Alaska	2-person families	59070	55594	62546
Alaska	3-person families	58137	54426	61848
Alaska	4-person families	66009	63320	68698

Рис. 5.15. Данные из таблицы в HTML-формате

Такие полезные для пользователя возможности могут быть очень важны. Вместо того чтобы тратить часы, вводя данные в HTML-формы и работать с неудобными веб-интерфейсами, ваш пользователь может просто воспользоваться привычными для него средствами, например Excel.

Смотрите также

- «Загрузка информации в базу данных из Excel» (см. трюк 46).

ТРЮК №46

Загрузка информации в базу данных из Excel

Используйте поддержку XML в Excel 2003 для загрузки в базу данных SQL информации из таблицы.

Очень часто у меня в распоряжении была таблица Excel, заполненная данными, которые мне необходимо было разместить в базе данных. До того как вышел Office 2003, мне приходилось преобразовывать каждую таблицу в CSV-формат, а затем использовать стандартный обработчик для добавления записей в базу данных. С появлением же в Excel 2003 возможностей сохранять таблицы, макросы и даже форматирование в XML я отправил этот загрузчик в корзину. Сценарий в этом трюке преобразовывает данные из XML-формата Excel в SQL, и затем вы можете размещать их в вашей базе данных. На рис. 5.16 показано, как сгенерированный при помощи Excel XML-файл передается в сценарий `gen.php` и преобразовывается в SQL, а затем размещается в базе данных.

Рис. 5.16. Взаимодействие базы данных с XML-файлом Excel

Код

Сохраните исходный код примера 5.36 в файл gen.php.

Пример 5.36. Исходный код для формирования SQL из XML-файла Excel

```
<?php
Stables = array( );
Sindata = 0;
function encode( $text )
{
  $text = preg_replace( "/'/", "''", Stext );
  return $text;
}

function start_element( Sparser, $name, Sattribs )
{
  global Stables, $indata;
  if ( Sname == "WORKSHEET" )
  {
    Stables []= array(
      'name' => $attribs['SS:NAME'],
      'data' => array( )
    );
  }
  if ( Sname == "ROW" )
  {
    $tables[count($tables)-1]['data'] []= array( );
  }
  if ( Sname == "DATA" )
  {
    Sindata = 1;
  }
}

function text( Sparser, Stext )
{
  global Stables, Sindata;
  if ( Sindata )
  {
    Sdata =& $tables[count($tables)-1]['data'];
    Sdata[count($data)-1] []= Stext;
```

```
  }
}

function end_element( Sparser, $name )
{
  global $indata;
  •if ( $name -- "DATA" )
    Sindata = 0:
}
Sparser - xml_parser_create( ):
xml_set_element_handler( Sparser. "start_element". "end_element" );
xml_set_character_data_handler( $parser. "text" ):
while( !feof( STDIN ) ) {
  Stext = fgets( STDIN );
  xml_parse( Sparser. Stext );
}
xml_parser_free( Sparser ):
foreach( Stables as Stable ) {
  $name - $table['name'];
  Sdata =& Stabler data'];
  $cols = implode( ". ". Sdata[0] );
  for( Sin = 1; Sin < count( Sdata ); $in++ ) {
    $sqldata = implode( ". ", array_map( "encode". Sdata[$in] ) );
 ?>
 INSERT INTO <?php echo( Sname )?> ( <?php echo( Scols ) ?> ) VALUES ( <?php echo(
$sqldata ): ?> ):
<?php } } ?>
```

В основном этот сценарий используется в качестве обработчика XML. Большая часть кода предназначена для анализа XML, в частности для поиска тегов <Data>, в которых содержатся данные из таблицы. Когда данные найдены, они помещаются в хранимый в памяти список таблиц. Для каждой из них есть соответствующие строки <Data>. В них содержатся данные из каждой ячейки таблицы. Во второй части сценария происходит преобразование полей и данных, размещенных в массиве $tables, в команды INSERT INTO (для простоты мы просто выведем их в консоль). Вы же можете запросто использовать полученные команды mysql для загрузки данных прямо в базу или сохранить их в файле для дальнейшего повторного использования.

Запуск трюка

Для начала создайте таблицу в Excel 2003. Я создал простейшую таблицу, которую вы можете видеть на рис. 5.17. Она предназначена для предварительной загрузки списка издателей в мою базу данных.

Я удалил листы, которые мне были не нужны, — эти пустые листы только увеличивают размер XML, — и затем переименовал первый лист со списком издателей.

ПРИМЕЧАНИЕ

Сценарий в примере 5.36 использует имя листа, содержащего данные, в качестве имени таблицы, в которую загружаются данные.

Рис. 5.17. Простая таблица Excel с информацией для базы данных

В первой строке листа содержатся имена полей. Во всех последующих строках содержатся данные для загрузки в таблицу. Далее сохраните этот файл как XML, воспользовавшись командой Save As (Сохранить как) в меню File (Файл) в Excel. Рассмотрим пример того, как выглядят данные в формате XML в этом файле:

```
<?xml version="1.0"?>
<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
  xmlns:o="urn:schemas-microsoft-com:office:office"
  xmlns:x="urn:schemas-microsoft-com:office:excel"
  xmlns:html="http://www.w3.org/TR/REC-html40"
  xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet">
  <DocumentProperties xmlns="urn:schemas-microsoft-com:office:office">
    <Author>Jack Herrington</Author>
    <LastAuthor>Jack Herrington</LastAuthor>
    <Created>2005-05-23T03:36:24Z</Created>
    <Company>MM</Company>
    <Version>11.257</Version>
  </DocumentProperties>
  <OfficeDocumentSettings xmlns="urn:schemas-microsoft-com:office:office">
    <AllowPNG/>
  </OfficeDocumentSettings>
  <ExcelWorkbook xmlns="urn:schemas-microsoft-com:office:excel">
    <WindowHeight>15080</WindowHeight>
    <WindowWidth>24840</WindowWidth>
    <WindowTopX>80</WindowTopX>
    <WindowTopY>-20</WindowTopY>
    <Date1904/>
```

```
    <AcceptLabelsInFormulas/>
    <ProtectStructure>False</ProtectStructure>
    <ProtectWindows>False</ProtectWindows>
  </ExcelWorkbook>
  <Styles>
    <Style ss:ID="Default" ss:Name="Normal">
      <Alignment ss:Vertical="Bottom"/>
      <Borders/>
      <Font ss:FontName="Verdana"/>
      <Interior/>
      <NumberFormat/>
      <Protection/>
    </Style>
  </Styles>
  <Worksheet ss:Name="Publisher">
  <Table ss:ExpandedColumnCount="2" ss:ExpandedRowCount="7" x:
  FullColumns="1"
  x:FullRows="1">
<Row>
  <Cell><Data ss:Type="String">ID</Data></Cell>
  <Cell><Data ss:Type="String">Name</Data></Cell>
</Row>
<Row>
  <Cell ss:Formula="=&quot;0&quot;"><Data ss:Type="String">0</Data></Cell>
  <Cell><Data ss:Type="String">O'Reilly</Data></Cell>
</Row>
<Row>
  <Cell ss:Formula="=&quot;0&quot;"><Data ss:Type="String">0</Data></Cell>
  <Cell><Data ss:Type="String">Manning</Data></Cell>
</Row>
```

Я хочу получить название листа, а также строки с данными из этого XML. Воспользуемся PHP-интерпретатором для командной строки для запуска сценария, генерирующего SQL на основе XML-данных из Excel:

```
% php gen.php < data.xml
INSERT INTO Publisher ( ID, Name ) VALUES ( '0', 'O''Reilly' );
INSERT INTO Publisher ( ID, Name ) VALUES ( '0', 'Manning' );
INSERT INTO Publisher ( ID, Name ) VALUES ( '0', 'Wiley' );
INSERT INTO Publisher ( ID, Name ) VALUES ( '0', 'Addison-Wesley' );
INSERT INTO Publisher ( ID, Name ) VALUES ( '0', 'Pragmatic Press' );
INSERT INTO Publisher ( ID, Name ) VALUES ( '0', 'APress' );
```

Вы также можете перенаправить выходную информацию прямо в файл, например следующим образом:

```
% php gen.php < data.xml > publishers.sql
```

Смотрите также

- «Получение данных из загруженных таблиц Excel» (см. трюк 45).
- «Динамическое создание таблиц Excel» (см. трюк 49).

ТРЮК №47

Организация поиска в документах Microsoft Word

Ищите текст в документах Microsoft Word при помощи анализа файлов WordML

Множество нужных данных бывает скрыто в документах Microsoft Word. В частности, такие документы, как резюме, отчасти очень подходят для приложений, импортирующих данные из каких-либо документов. Разделы с объявлениями о поиске работы часто нуждаются в коде, который обрабатывает документы Word и находит ключевые слова или фразы для распределения по категориям кандидатов на работу. В этом трюке показано, как организовать поиск текстовых строк в документах Word, сохраненных как WordML.

Код

Сохраните исходный код из примера 5.37 в файл index.php.

Пример 5.37. HTML, реализующий загрузку данных

```
<html>
<body>
<form enctype="multipart/form-data" action="search.php" method="post">
WordML file: <input type="hidden" name="MAX_FILE_SIZE" value="2000000" />
  <input type="file" name="file" /><br/>
  <input type="submit" value="Upload" />
</form>
</body>
</html>
```

Сохраните код из примера 5.38 в файл search.php. Этот сценарий просматривает загруженный файл WordML и обнаруживает определенные фрагменты.

Пример 5.38. Сценарий, реализующий поиск по документу

```
<html>
<body>
<?php
$wordlist = array( );
$dom = new DOMDocument( );
if ( $_FILES['file']['tmp_name'] )
{
  $dom->load( $_FILES['file']['tmp_name'] );
  $found = $dom->getElementsByTagName( "t" );
  foreach( $found as $element )
  {
    $words = split( ' ', $element->nodeValue );
    foreach( $words as $word )
    {
      $word = preg_replace( '7[.]|[.]/'. '', $word );
      $word = preg_replace( '/^\s+/', '', $word );
      $word = preg_replace( '/\s+$/', '', $word );
      if ( strlen( $word ) > 0 )
```

```
        {
          $word = strtolower( $word );
          $wordlist[ $word ] = 0;
        }
      }
    }
}
$words = array_keys( $wordlist );
sort( $words );
foreach( $words as $word ) {
?>
<?php echo( $word ); ?><br/>
<?php } ?>
</body>
</html>
```

В сценарии search.php работа начинается с того, что открывается загруженный файл WordML при использовании объектов типа XML DOM. Затем находятся все узлы t. В них-то и хранится текст документа. В этих разделах удаляется вся пунктуация. Затем оставшийся текст разбивается на слова, и они сохраняются их в хеш-таблицу $wordlist. И наконец, этот список слов выводится на экран.

Запуск трюка

Создайте простой документ в Microsoft Word 2003 и сохраните его как файл WordML где-нибудь у себя на диске. Затем загрузите эти файлы к себе на веб-сервер и откройте страницу index.php. Она должна выглядеть так, как показано на рис. 5.18. Нажмите кнопку **Browse** (Обзор) и выберите файл WordML. Затем нажмите кнопку **Upload** (Загрузить). Таким образом, вы передадите файл в сценарий search.php. Он воспользуется объектами типа XML DOM, чтобы считать этот файл. Данные, хранимые в файле WordML, будут отсортированы и переданы HTML-странице, как это показано на рис. 5.19. Здесь вы уже можете отыскать определенные слова или подсчитать, сколько раз они встречались в документе (см. трюк 24).

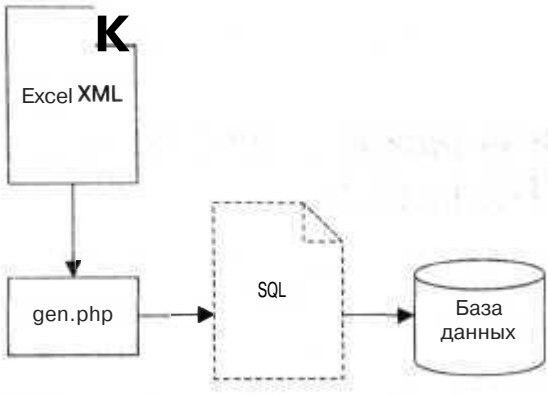


Рис. 5.18. Структура страницы, используемой для загрузки файла

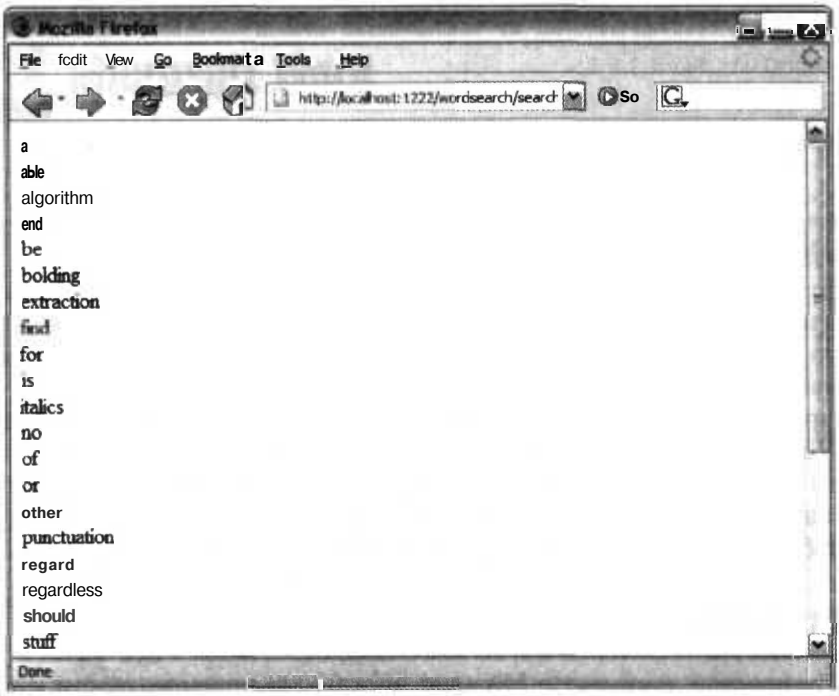


Рис. 5.19. Слова, встречаемые в загруженном документе

ПРИМЕЧАНИЕ

Формат WordML поддерживается только в Microsoft Word 2003 и более поздних версиях. На данный момент он еще не поддерживается Macintosh, хотя я надеюсь на это в более поздних версиях. Для работы в более старых версиях Microsoft Word вам, возможно, захочется переписать исходный код трюка, обрабатывающего документы в RTF-формате вместо WordML. RTF-формат поддерживают все недавние версии Microsoft Word.

Смотрите также

- «Динамическое создание документов в RTF-формате » (см. трюк 48).

ТРЮК №48

Динамическое создание документов в RTF-формате

Используйте **PHP** для динамического формирования документов в формате Rich Text Format (RTF).

Формат Rich Text Format (RTF) — это текстовый формат, используемый текстовыми редакторами, особенно Microsoft Word, а также программами для просмотра и предназначенный для сохранения документов в более современной форме.

Если вы хотите формировать документы динамически, используя все возможности текстовых процессоров, в этом вам поможет RTF-формат.

Для начала создайте документ в текстовом редакторе, например в Microsoft Word. На рис. 5.20 показан используемый в этом трюке документ.

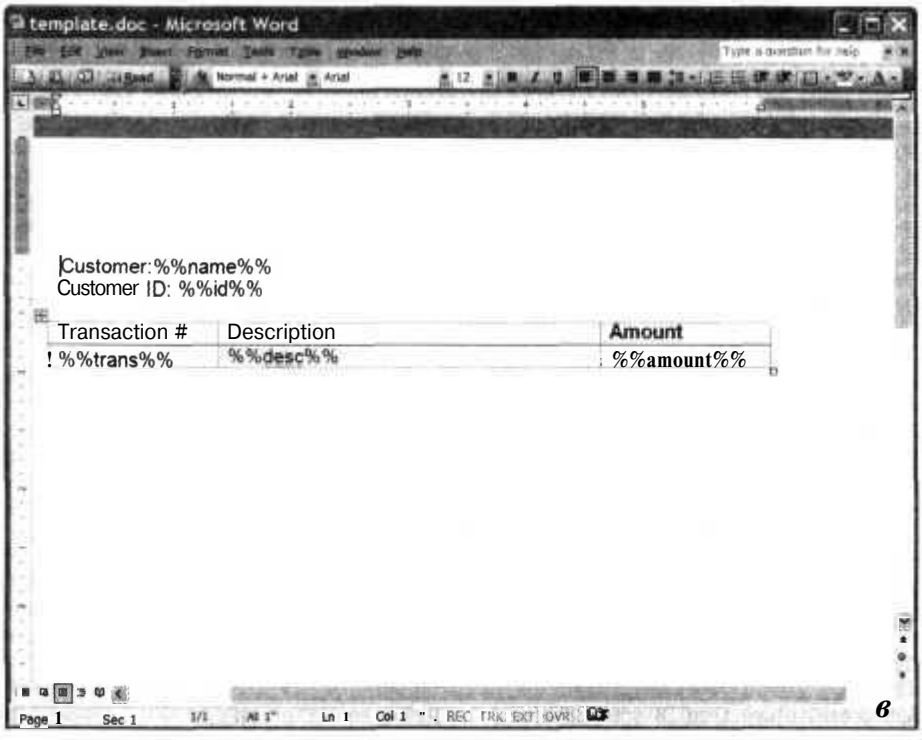


Рис. 5.20. Исходный документ Microsoft Word

Часть документа, помеченная символами %%, — это то место, где я хочу размещать динамические данные. Я мог бы подобрать другие специальные символы, но символы %% более отчетливые и необычные; к тому же знаки процентов не преобразуются в RTF. Получив готовый текстовый документ, воспользуйтесь командой Save As (Сохранить как), чтобы сохранить его в RTF-формате. Затем, используя RTF-файл в качестве шаблона, вы можете создать код на PHP, который будет формировать новые RTF-документы.

Код

Сохраните текст из примера 5.39 в файл rtf.php.

Пример 5.39. Использование escape-последовательностей для отображения документа в RTF-формате

```
<? header( "content-type: application/msword" );
$customerName = "First customer";
```

```
$customerID = "cust_0001";
$data = array(
  array( trans => "123". desc => "Books", amount => '$123.25' ).
  array( trans => "345". desc => "Stamps", amount => '$22.93' ).
  array( trans => "1531". desc -> "Candles", amount => '$56.27' )
);
?>
{\rtf1\ansi\ansicpg1252\uc1\deff0\stshfdbch0\stshfloch0\stshfhich0\stshfbi0\
def1ang1033\deflangfe1033
{\fonttbl{\f0\froman\fcharset0\fprq2{\*\panose 02020603050405020304}
Times New Roman;}{\f1\fswiss\fcharset0\fprq2{\*\panose
020b0604020202020204}Arial;}
{\f180\froman\fcharset238\fprq2 Times New Roman CE;}{\f181\froman\fcharset204\
fprq2 Times New Roman Cyr;}
{\f183\froman\fcharset161\fprq2 Times New Roman Greek;}{\f184\froman\fcharset162\
fprq2 Times New Roman Tur;}
{\f185\froman\fcharset177\fprq2 Times New Roman (Hebrew);}
{\f186\froman\fcharset178\fprq2 Times New Roman (Arabic);}
{\f187\froman\fcharset186\fprq2 Times New Roman Baltic;}
{\f188\froman\fcharset163\fprq2 Times New Roman (Vietnamese);}
{\f190\fswiss\fcharset238\fprq2 Arial CE;}{\f191\fswiss\fcharset204\fprq2 Arial
Cyr;}
{\f193\fswiss\fcharset161\fprq2 Arial Greek;}{\f194\fswiss\fcharset162\fprq2
Arial Tur;}
{\f195\fswiss\fcharset177\fprq2 Arial (Hebrew);}
{\f196\fswiss\fcharset178\fprq2 Arial (Arabic);}
{\f197\fswiss\fcharset186\fprq2 Arial Baltic;}{\f198\fswiss\fcharset163\fprq2
Arial (Vietnamese);}
}{\colortbl;\red0\green0\blue0;\red0\green0\blue255;\red0\green255\blue255;\red0\
green255\blue0;
\red255\green0\blue255;\red255\green0\blue0;\red255\green255\blue0;\red255\
green255\blue255;
\red0\green0\blue128;\red0\green128\blue128;\red0\green128\blue0;\red128\green0\
blue128;
\red128\green0\blue0;\red128\green128\blue0;\red128\green128\blue128;
\red192\green192\blue192;}{\stylesheet{\ql
\li0\ri0\widctlpar\aspalpha\aspnum\faauto\adjustright\rin0\lin0\itap0
\fs24\lang1033\langfe1033\cgrid\langnp1033\langfenp1033 \snext0 Normal;}
{\*\cs10 \additive \ssemihidden Default Paragraph Font;}{\*
\ts11\tsrowd\trftsWidthB3\trpaddl108\trpaddr108\trpaddfl3\trpaddft3\trpaddfb3\
trpaddfr3\tscellwidthfts0\tsvertalt\tsbrdrt\tsbrdrl\tsbrdrb\tsbrdrr\tsbrdrdgl\
tsbrdrdgr\tsbrdrh\tsbrdrv
\ql \li0\ri0\widctlpar\aspalpha\aspnum\faauto\adjustright\rin0\lin0\itap0
\fs20\lang1024\langfe1024\cgrid\langnp1024\langfenp1024 \snext11
\ssemihidden Normal Table;}{\*\ts15\tsrowd\trbrdrt\brdrs\brdrw10
\trbrdrl\brdrs\brdrw10 \trbrdrb\brdrs\brdrw10
\trbrdrr\brdrs\brdrw10 \trbrdrh\brdrs\brdrw10 \trbrdrv\brdrs\brdrw10
\trftsWidthB3\trpaddl108\trpaddr108\trpaddfl3\trpaddft3\trpaddfb3\trpaddfr3\
tscellwidthfts0\tsvertalt\tsbrdrt\tsbrdrl\tsbrdrb\tsbrdrr\tsbrdrdgl\tsbrdrdgr\
tsbrdrh\tsbrdrv
\ql \li0\ri0\widctlpar\aspalpha\aspnum\faauto\adjustright\rin0\lin0\itap0
\fs20\lang1024\langfe1024\cgrid\langnp1024\langfenp1024 \sbasedon11 \snext15
\styrsid6312866 Table Grid;}}{\*\latentstyles\lsdstimax156\lsdlockeddef0}{\*\
rsidtbl \rsid6312866}
{\*\generator Microsoft Word 11.0.6359;}{\info{\title Customer <? print(
$customerName ) ?> }
```

```
{\author jherring}{\operator jherring}{\creatim\yr2005\mo4\dy30\hr21\min46}
{\revtim\yr2005\mo4\dy30\hr21\min46}{\version2}{\edmins0}{\nofpages1}{\
nofwords15}{\nofchars91}
{\*\company Macromedia Inc.}{\nofcharsws105}{\vern24703}}
\widowctrl\ftnbj\aenddoc\noxlattoyen\expshrtn\noultrlspc\dntblnsbdb\nospaceforul\
formshade\horzdoc\dgmargin\dghspace180\dgvspace180\dghorigin1800\dgvorigin1440\
dghshowl\dgvshowl
\jexpand\viewkind1\viewscale125\pgbrdrhead\pgbrdrfoot\splytwnine\ftnlytwnine\
htmautsp\nolnhtadjtbl\useltbaln\alntblind\lytcalctblwd\lyttblrtgr\lnbrkrule\
nobrkwrptbl\snaptogridincell\allowfieldendsel\wrppunct\asianbrkrule\nojkernpunct\
rsidroot6312866 \fet0
\sectd \linex0\endnhere\sectlinegrid360\sectdefaultcl\sftnbj
{\*\pnseclvl1\pnucrm\pnstart1\pnindent720\pnhang {\pntxta .}}
{\*\pnseclvl2\pnucltr\pnstart1\pnindent720\pnhang {\pntxta .}}
{\*\pnseclvl3\pndec\pnstart1\pnindent720\pnhang {\pntxta .}}
{\*\pnseclvl4\pnlcltr\pnstart1\pnindent720\pnhang {\pntxta )}}
{\*\pnseclvl5\pndec\pnstart1\pnindent720\pnhang {\pntxtb (}{\pntxta )}}
{\*\pnseclvl6\pnlcltr\pnstart1\pnindent720\pnhang {\pntxtb (}{\pntxta )}}
{\*\pnseclvl7\pnlcrm\pnstart1\pnindent720\pnhang
{\pntxtb (}{\pntxta )}}{\*\pnseclvl8\pnlcltr\pnstart1\pnindent720\pnhang {\pntxtb
(}
{\pntxta )}}{\*\pnseclvl9\pnlcrm\pnstart1\pnindent720\pnhang {\pntxtb (}{\pntxta
)}}
\pard\plain \ql \li0\ri0\widctlpar\aspalpha\aspnum\faauto\adjustright\rin0\lin0\
itapO
\fs24\lang1033\langfe1033\cgrid\langnp1033\langfenp1033
{\f1\insrsid6312866\charrsid6312866 Customer: <? print( $customerName) ?>
\par Customer ID: <? print( $customerID ) ?>
\par
\par }\trowd \irow0\irowband0\ts15\trgaph108\trleft-108\trftsWidth1\trftsWidthB3\
trftsWidthA3\trautofit1\trpaddl108\trpaddr108\trpaddfl3\trpaddft3\trpaddfb3\
trpaddfr3\tbl1khdrrows\tbl1klastrow\tbl1khdrcols\tbl1klastcol \clvertalt\clbrdrt\
brdrtbl \clbrdrl
\brdrtbl \clbrdrb\brdrs\brdrw10 \clbrdrr\brdrtbl \cltxlrtb\clftsWidth3\
clwWidth2088\clshdrawnil
\cellx1980\clvertalt\clbrdrt\brdrtbl \clbrdrl\brdrtbl \clbrdrb\brdrs\brdrw10 \
clbrdrr\brdrtbl
\cltxlrtb\clftsWidth3\clwWidth4680\clshdrawnil \cellx6660
\clvertalt\clbrdrt\brdrtbl \clbrdrl\brdrtbl \clbrdrb\brdrs\brdrw10
\clbrdrr\brdrtbl \cltxlrtb\clftsWidth3\clwWidth2088\clshdrawnil \cellx8748\pard \
ql
\li0\ri0\widctlpar\intbl\aspalpha\aspnum\faauto\adjustright\rin0\lin0 {
\b\f1\insrsid6312866\charrsid6312866 Transaction #\cell Description\cell Amount\
cell }\pard
\ql \li0\ri0\widctlpar\intbl\aspalpha\aspnum\faauto\adjustright\rin0\lin0
{\b\f1\insrsid6312866\charrsid6312866 \trowd \irow0\irowband0
\ts15\trgaph108\trleft-108\trftsWidth1\trftsWidthB3\trftsWidthA3\trautofit1\
trpaddl108\trpaddr108\trpaddfl3\trpaddft3\trpaddfb3\trpaddfr3\tbllkhdrrows\
tbl1klastrow\tbl1khdrcols\tbl1klastcol \clvertalt\clbrdrt\brdrtbl \clbrdrl\
brdrtbl \clbrdrb
\brdrs\brdrwl0 \clbrdrr\brdrtbl \cltxlrtb\clftsWidth3\clwWidth2088\clshdrawnil
\cellxl980\clvertalt\clbrdrt\brdrtbl \clbrdrl\brdrtbl
\clbrdrb\brdrs\brdrw10 \clbrdrr\brdrtbl \cltxlrtb\clftsWidth3\clwWidth4680\
clshdrawnil
\cellx6660\clvertalt\clbrdrt
\brdrtbl \clbrdrl\brdrtbl \clbrdrb\brdrs\brdrwl0 \clbrdrr\brdrtbl
```

```
\cltxlrtb\clftsWidth3\clwWidth2088\clshdrawnil \cellx8748\row }\trowd
\irow1\irowband1\lastrow
<? foreach( $data as $row ) { ?>
\ts15\trgaph108\trleft-108\trftsWidth1\trftsWidthB3\trftsWidthA3\trautofit1\
trpaddl108\trpaddr108\trpaddfl3\trpaddft3\trpaddfb3\trpaddfr3\tbllkhdrrows\
tbllklastrow\tbllkhdrcols\tbl1klastcol\clvertalt\clbrdrt\brdrs\brdrw10\clbrdrl\
brdrtbl \clbrdrb
\brdrtbl \clbrdrr\brdrtbl \cltxlrtb\clftsWidth3\clwWidth2088\clshdrawnil
\cellx1980\clvertalt\clbrdrt\brdrs\brdrw10 \clbrdrl\brdrtbl \clbrdrb\brdrtbl
\clbrdrr\brdrtbl \cltxlrtb\clftsWidth3\clwWidth4680\clshdrawnil \cellx6660\
clvertalt\clbrdrt
\brdrs\brdrw10 \clbrdrl\brdrtbl \clbrdrb\brdrtbl \clbrdrr\brdrtbl
\cltxlrtb\clftsWidth3\clwWidth2088\clshdrawnil \cellx8748\pard \ql
\li0\ri0\widctlpar\intbl\aspalpha\aspnum\faauto\adjustright\rin0\lin0
{\f1\insrsid6312866 <? print( $row['trans'] ) ?>\cell <? print( $row['desc'] ) ?>
\cell
<? print( $row['amount'] ) ?>\cell }\pard \ql
\li0\ri0\widctlpar\intbl\aspalpha\aspnum\faauto\adjustright\rin0\lin0 {\fl\
insrsid6312866
\trowd \irow1\irowband1\lastrow
\ts15\trgaph108\trleft-108\trftsWidth1\trftsWidthB3\trftsWidthA3\trautofit1\
trpaddl108\trpaddr108\trpaddfl3\trpaddft3\trpaddfb3\trpaddfr3\tbllkhdrrows\
tbllklastrow\tbllkhdrcols\tbllklastcol \clvertalt\clbrdrt\brdrs\brdrw10 \clbrdrl\
brdrtbl \clbrdrb
\brdrtbl \clbrdrr\brdrtbl \cltxlrtb\clftsWidth3\clwWidth2088\clshdrawnil
\cellx1980\clvertalt\clbrdrt\brdrs\brdrw10 \clbrdrl\brdrtbl \clbrdrb\brdrtbl
\clbrdrr\brdrtbl \cltxlrtb\clftsWidth3\clwWidth4680\clshdrawnil \cellx6660\
clvertalt\clbrdrt
\brdrs\brdrwlO \clbrdrl\brdrtbl \clbrdrb\brdrtbl \clbrdrr\brdrtbl
\cltxlrtb\clftsWidth3\clwWidth2088\clshdrawnil \cellx8748\row }
<? } ?>
\pard \ql \li0\ri0\widctlpar\aspalpha\aspnum\faauto\adjustright\rin0\lin0\itap0
{\f1\insrsid6312866\charrsid6312866
\par } }
```

Код, который вы видите, — это всего лишь содержимое нашего файла Word RTF с функцией header сверху и небольшим дополнительным исходным кодом на PHP для добавления этого содержимого туда, где оно потребуется. Функция header сообщает браузеру, что перед ним RTF-документ, а не плохо отформатированный HTML или текст.

RTF — это сложный файловый формат, что особенно заметно, когда дело доходит до вывода на экран содержимого документа с насыщенным форматированием. Вам не надо полностью понимать RTF-формат для формирования документов. Вы можете просто поступить, как я, то есть использовать ваш текстовый процессор как и обычно, но расставлять в документе маркеры, чтобы определять, где следует размещать динамическое содержимое. Другими словами, в этом трюке используется чисто механическая работа, а не какой-то мистический процесс преобразования. Это скучно, согласен, но в то же время очень полезно!

Запуск трюка

Когда я открываю сценарий в Internet Explorer, первое, что я вижу, — это окно, как и на рис. 5.21.

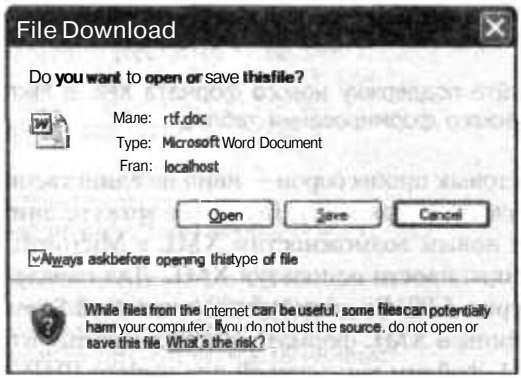


Рис. 5.21. Сообщение от службы безопасности, появляющееся при загрузке страницы

Это окно позволяет пользователю выбрать, как обработать сгенерированный сервером RTF-файл. Я выбрал вариант открытия файла, после чего отобразился встроенный в браузер компонент для управления документами Word, как это показано на рис. 5.22. Часть документа, ранее помеченная символами %%, заменена данными с сервера, причем можно было запросто поместить здесь информацию и из базы данных.

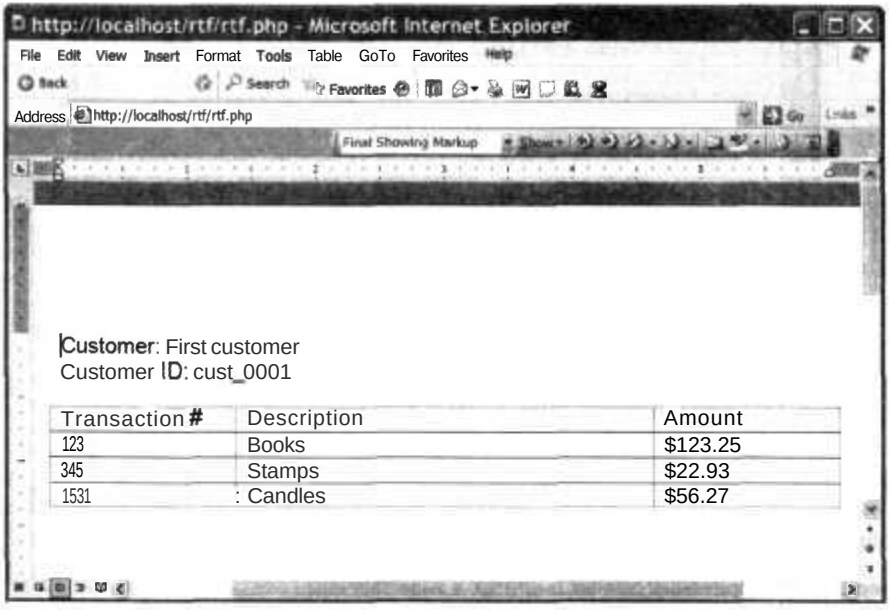


Рис.5.22. Динамически созданный RTF-файл, отображенный в Internet Explorer

Смотрите также

- «Динамическое создание таблиц Excel» (см. трюк 49).

ТРЮК №49

Динамическое создание таблиц Excel

Используйте поддержку нового формата XML в Microsoft Office 2004 для динамического формирования таблиц.

Документы для текстовых процессоров — явно не единственное, что вы захотите создавать динамически (см. трюк 48). Вы также можете динамически создавать таблицы. Благодаря новым возможностям XML в Microsoft Office 2004 мы можем создавать таблицы, просто используя XML. Для начала создайте документ, как это показано на рис. 5.23. Воспользуйтесь командой Save As (Сохранить как) для сохранения таблицы в XML-формате. Затем воспользуйтесь созданным приложением Excel XML-файлом как основой для вашего PHP-файла.

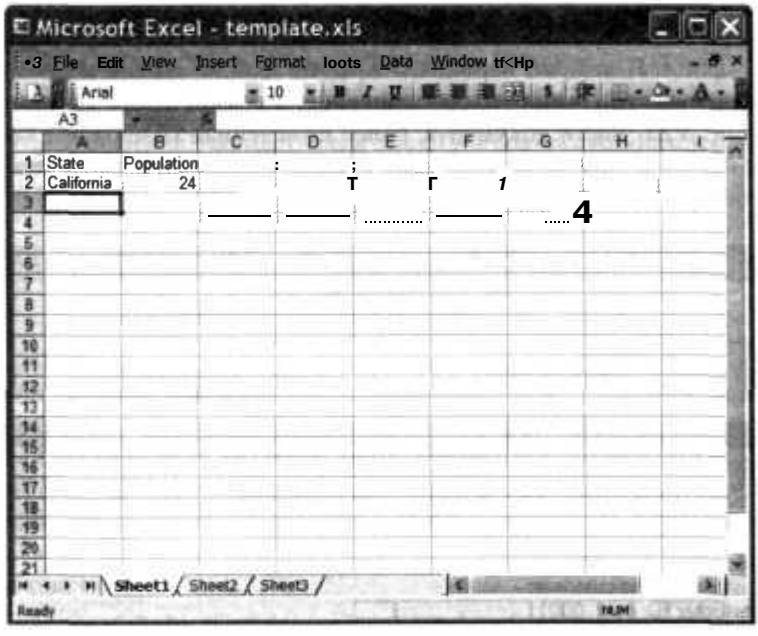


Рис. 5.23. Простая таблица Excel, которой я воспользовался в качестве шаблона

Код

Сохраните исходный код из примера 5.40 в файл `spreadsheet.php`.

Пример 5.40. Отображение данных о штатах в виде таблицы

```
<?
header( "content-type: text/xml" );
$data = array(
  array(state => "Alabama", population => 4530182 ),
  array(state => "Alaska", population -> 655435 ),
  array(state => "Arizona", population => 5743834 ),
  array(state => "Arkansas", population => 2752629 ),
```

```
  array(state -> "California", population => 35893799 ),
  ...
  array(state => "Washington", population -> 6203788 ),
  array(state => "West Virginia", population => 1815354 ),
  array(state -> "Wisconsin", population => 5509026 ),
  array(state => "Wyoming", population => 506529 )
);

echo( "<?xml version=\"1.0\"?>\n" );
echo( "<?mso-application progid=\"Excel.Sheet\"?>\n" );
?>
<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
  xmlns:o="urn:schemas-microsoft-com:office:office"
  xmlns:x="urn:schemas-microsoft-com:office:excel"
  xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
  xmlns:html="http://www.w3.org/TR/REC-html40">
  <DocumentProperties xmlns="urn:schemas-microsoft-com:office:office">
    <Author>Jack Herrington</Author>
    <LastAuthor>Jack Herrington</LastAuthor>
    <Created>2005-04-30T14:08:07Z</Created>
    <LastSaved>2005-04-30T14:09:14Z</LastSaved>
    <Company>Myself</Company>
    <Version>11.6360</Version>
  </DocumentProperties>
  <OfficeDocumentSettings xmlns="urn:schemas-microsoft-com:office:office">
    <DownloadComponents/>
    <LocationOfComponents HRef="file:///C:\APPINSTALL\Microsoft\Office_Pro_2003\"/>
  </OfficeDocumentSettings>
  <ExcelWorkbook xmlns="urn:schemas-microsoft-com:office:excel">
    <WindowHeight>15930</WindowHeight>
    <WindowWidth>20025</WindowWidth>
    <WindowTopX>480</WindowTopX>
    <WindowTopY>105</WindowTopY>
    <ProtectStructure>False</ProtectStructure>
    <ProtectWindows>False</ProtectWindows>
  </ExcelWorkbook>
  <Styles>
    <Style ss:ID="Default" ss:Name="Normal">
      <Alignment ss:Vertical="Bottom"/>
      <Borders/>
      <Font/>
      <Interior/>
      <NumberFormat/>
      <Protection/>
    </Style>
  </Styles>
  <Worksheet ss:Name="States">
    <Table ss:ExpandedColumnCount="2"
    ss:ExpandedRowCount="<? print( count( $data ) + 1 ) ?>"
    x:FullColumns="1"
    x:FullRows="1">
    <Row>
      <Cell><Data ss:Type="String">State</Data></Cell>
      <Cell><Data ss:Type="String">Population</Data></Cell>
    </Row>
    <? foreach( $data as $row ) { ?>
    <Row>
```

```
      <Cell><Data ss:Type="String"><? print( $row["state"] ) ?></Data></Cell>
      <Cell><Data ss:Type="Number"><? print( $row["population"] ) ?>
      </Data></Cell>
    </Row>
    <? } ?>
    </Table>
    <WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
      <Selected/>
      <Panes>
        <Pane>
          <Number>3</Number>
          <ActiveRow>2</ActiveRow>
        </Pane>
      </Panes>
      <ProtectObjects>False</ProtectObjects>
      <ProtectScenarios>False</ProtectScenarios>
    </WorksheetOptions>
  </Worksheet>
</Workbook>
```

Я воспользовался исходной XML-таблицей в качестве базы для сценария, а затем добавил функцию header() в самом начале, чтобы указать, что это данные в XML-формате (чтобы браузер корректно их обработал). Данные, строго запрограммированные и используемые в этом сценарии, — это информация о количестве населения в штатах США исходя из переписи в 2000 году. Я использовал цикл foreach для создания строк таблицы.

Очень важно получить правильное значение ss:ExpandedRowCount. Если это значение будет некорректным, то Excel откажется загружать таблицу, а только сообщит о непонятной ошибке. В целом, я и создаю таблицы соответствующим образом, так как я заметил, что Excel чересчур сильно печется о целостности данных. Я пытаюсь делать небольшие корректировки в XML, пока не достигну того, что мне требуется. Сделав же сразу **много изменений,** вы рискуете испытать огромное разочарование в случае ошибки, так как Excel проигнорирует некорректные данные, толком и не сообщив вам, где же произошла ошибка.

Запуск трюка

Скопируйте файлы на сервер и откройте PHP-сценарий в Internet Explorer (рис. 5.24).

Браузер Internet Explorer открывает окно службы безопасности, когда видит, что загружается отличное от HTML содержимое. Нажав кнопку Open (Открыть), вы запустите элемент управления Excel, встроенный в Internet Explorer, и в его окне отобразятся данные из XML-таблицы. Пример этого вы можете видеть на рис. 5.25 — как раз то, что мы и хотели сделать!

Смотрите также

- «Получение данных из загруженных таблиц Excel» (см. трюк 45).
- «Динамическое создание документов в RTF-формате» (см. трюк 48).

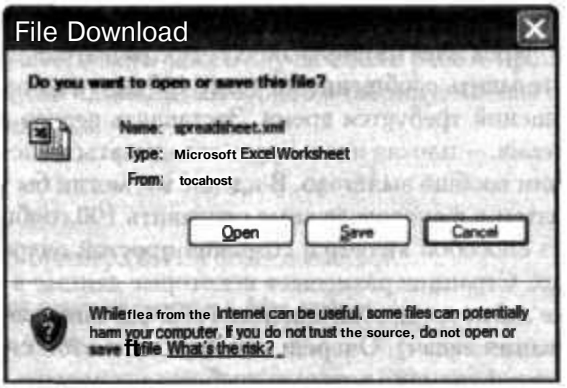


Рис. 5.24. Сообщение от службы безопасности, появляющееся при загрузке страницы

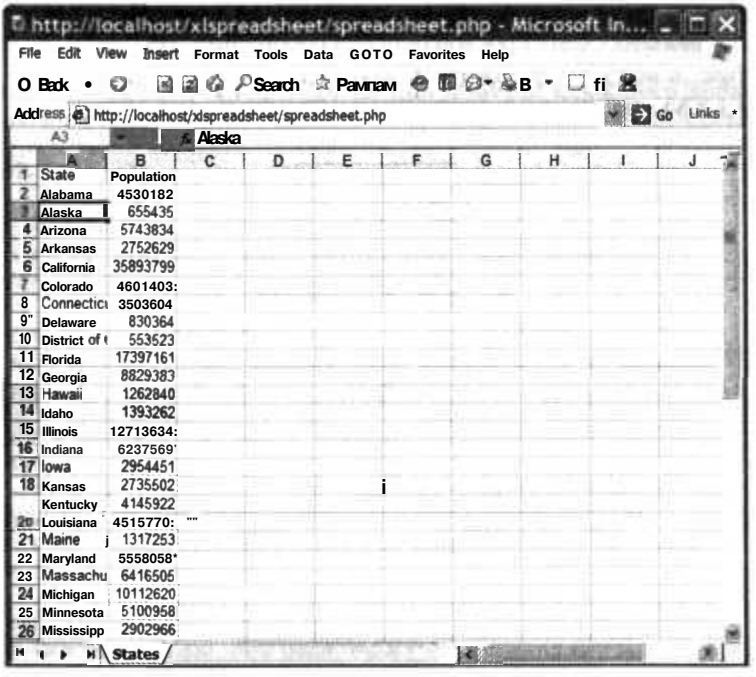


Рис. 5.25. Готовая таблица с динамическими данными

ТРЮК №50

Создание очереди сообщений

Используйте таблицы MySQL для создания простых очередей сообщений при использовании отложенных уведомлений.

Фоновые процессы всегда были проблемой в веб-приложениях. Пользователям нравится мгновенно получать ответ на запрос от веб-страниц, но иногда обработка

данных может занимать время. Классический пример — отправка сообщений при помощи рассылки через электронную почту. Пользователь запускает процесс, который должен отправить сообщение по почте 100 другим пользователям, но на отправку 100 сообщений требуется время. Заставлять сервер ждать, пока будут отправлены все письма, — плохая идея. Будет складываться впечатление, что приложение зависло или вообще вылетело. В идеале мы могли бы вернуть пользователю страницу, а затем в фоновом режиме отправить 100 сообщений. Но как это сделать? Одним из способов является создание простой очереди сообщений на основе базы данных. Страница размещает некоторые данные в очереди, обработкой которых позже займется другой процесс (обычно запускаемый при помощи системы планирования задач). Очередь сообщений в этой системе принимает в себя два параметра: функцию, которую требуется запустить, и аргументы для этой функции. Таким образом, вы можете отложить выполнение любого процесса, какого захотите.

На рис. 5.26 показана схема простой очереди сообщений. На самом деле в ней есть два поля: `func`, в котором содержится имя функции, и `args`, в котором хранятся аргументы в XML-формате.

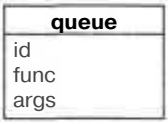


Рис. 5.26. Схема очереди

На рис. 5.27 показан статус таблицы queue. Сперва она пуста, в ней нет сообщений. Затем добавляется пара сообщений. По мере обработки сообщений они удаляются.

Простая очередь сообщений

id	func	args

После добавления двух сообщений

id	func	args
1	mail_notification	<xml>...</xml>
2	mail_notification	<xml>...</xml>

После удаления одного сообщения

id	func	args
2	mail_notification	<xml>...</xml>

Рис. 5.27. Очередь после нескольких операций

Одна из самых важных особенностей очереди сообщений заключается в том, что она продолжает существовать даже в случае сбоя в работе процесса, считывающего из таблицы очереди или записывающего в таблицу, так как очередь сообщений находится в базе данных.

Код

Сохраните SQL из примера 5.41 в файл queue.sql.

Пример 5.41. SQL для создания очереди в базе данных

```
DROP TABLE IF EXISTS queue:
CREATE TABLE queue (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  func TEXT.
  args TEXT.
  PRIMARY KEY ( id )
);
```

Управление же очередью производится при помощи исходного кода из примера 5.42 (сохраните его в файл с именем queue.php).

Пример 5.42. Сценарий, управляющий очередью

```
<?php
require( "DB.php" );
$db =& DB::connect("mysql://root:password@localhost/queue", array( )):
if (PEAR::isError($db)) { die($db->getMessage( )): }

function add_to_queue( $func. $args )
{
  global $db;
         $dom = new DomDocument( );
         $root = $dom->createElement( "arguments" );
  foreach( $args as $argtext )
  {
    $arg = $dom->createElement( "argument" );
    $arg->appendChild( $dom->createTextNode( Sargtext ) );
    $root->appendChild( $arg );
  }
  $dom->appendChild( $root );
  $sth = $db->prepare( "INSERT INTO queue VALUES ( 0. ?. ? )" );
  $db->execute( $sth. array( $func. $dom->saveXML( ) ) ):
}

function run_queue( )
{
  global $db:
         $delsth = $db->prepare( "DELETE FROM queue WHERE id = ?" );
         $res = $db->query( "SELECT id. func. args FROM queue" );
  while( $res->fetchInto( $row ) )
  {
    $id - $row[0]:
    $func » $row[1];
    $argxml =$row [2 ];
```

```
    $dom =new DomDocument();
    $dom->loadXML($argxml );
    $args =array();
    foreach($dom->getElementsByTagName("argument")as $node )
    {
      $args[]=$node->nodeValue:
    }
    call_user_func_array($func,$args );
    $db->execute($delsth.array($id ));
  }
}
?>
```

Файл add.php, исходный код которого вы можете видеть в примере 5.43, предназначен для проверки возможности добавления письма в очередь.

Пример 5.43. Простой сценарий, отправляющий сообщение по почте

```
<?php
require_once( 'queue.php' );

add_to_queue( 'mail_notification',
  array(
    'jack@oreilly.com',
    'You owe us some money. Pay up.'
  )
);
?>
```

Сценарий run.php (пример 5.44) выдает сообщение о статусе очереди.

Пример 5.44. Сценарий, сообщающий (частично) пользователю, что происходит в данный момент

```
<?pnp
require_once( "queue.php" );
 function mail_notification( $user, $text )
{
  print "Mailing $user:\n$text\n\n":
}
run_queue( ):
?>
```

Основной код в этом примере расположен в библиотеке queue.php, используемой обоими сценариями: сценарием add.php, добавляющим элементы в очередь, и сценарием run.php, запускающим на выполнение сохраненные в очереди элементы.

В библиотеке для работы с очередью определены две функции: addtoqueueO, которая добавляет запись в очередь, и run_queueO, которая получает все строки из таблицы и запускает их одну за другой. Функция run_queue() — самая интересная, так как она никогда не знает, что будет делать! Она просто считывает из базы данных имя функции и аргументы для нее. Затем она вызывает PHP-функцию call_user_func_array(), которая принимает в качестве параметров имя функции и ее аргументы. Это означает, что вы можете расширить функциональность вашей очереди, не внося никаких изменений в базовые функции добавления и запуска.

Запуск трюка

Запуск трюка начинается с создания таблицы для очереди в базе данных:

```
% mysqladmin --user=root --password=password create queue
% mysql --user=root --password=password queue < queue.sql
```

Запустив сценарий add.php при помощи PHP-интерпретатора для командной строки, вы добавите сообщение в очередь (этот код будет запущен на веб-странице). В нашем случае это будет используемое в качестве примера сообщение по электронной почте. Вы можете запустить сценарий несколько раз, если хотите создать несколько тестовых сообщений для электронной почты:

```
% php add.php
```

Затем, запустив сценарий run.php, вы отправите на выполнение сохраненные в очереди сообщения:

```
% php run.php
Mailing jack@oreilly.com:
You owe us some money. Pay up.
```

Каждое сообщение выводится на консоль, но его также можно запросто отправить и по электронной почте.

Смотрите также

- «Отслеживание ваших объектов» (см. трюк 67).

Дизайн приложений

Трюки 51-66

Засев на верхушку базы данных и укутавшись в HTML, вы только создадите логику приложения. В этой же главе мы сосредоточимся на трюках, при помощи которых можно сделать эту логику более стабильной и гибкой. В раскрытых здесь темах мы обсудим проблемы безопасности и ролей, управления паролями, авторизации и управления сеансами, а также некоторые вопросы, связанные с электронной коммерцией.

ТРЮК №51

Создание модульных интерфейсов

Используйте динамическую загрузку, чтобы позволить вашим пользователям самостоятельно разрабатывать встраиваемые модули для вашего приложения.

Большинство действительно популярных PHP-приложений с открытым кодом обладают механизмом подключения расширений, что позволяет программистам на PHP создавать небольшие фрагменты кода, которые динамически загружаются в приложение. В этом трюке продемонстрирован созданный на базе XML сценарий для рисования, возможности которого вы можете расширить, просто добавив в директорию для модулей новые классы, написанные на PHP, но, конечно, же, рисование точки — это не тот код, который можно с легкостью усовершенствовать.

Код

Сохраните исходный код из примера 6.1 в файл modhost.php.

Пример 6.1. Код, реализующий модульную архитектуру на PHP

```
<?php
class DrawingEnvironment
{
  private $img = null;
  private $x = null;
  private $y = null;
  private $colors = array( );
  public function___constructs $x. $y )
```

```
  {
    $this->img = imagecreatetruecolor( $x, $y );
    $this->addColor( 'white', 255, 255, 255 );
    $this->addColor( 'black', 0, 0, 0 );
    $this->addColor( 'red', 255, 0, 0 );
    $this->addColor( 'green', 0, 255, 0 );
    $this->addColor( 'blue', 0, 0, 255 );
    imagefilledrectangle( $this->image( ),
    0, 0, $x, $y, $this->color( 'white' ) );
  }
  public function image( ) { return $this->img; }
  public function size_x( ) { return $this->x; }
  public function size_y( ) { return $this->y; }
  public function color( $c ) { return $this->colors[$c]; }
  public function save( $file )
  {
    imagepng( $this->img, $file );
  }
  protected function addColor( $name, $r, $g, $b )
  {
    $col = imagecolorallocate($this->img, $r, $g, $b);
    $this->colors[ $name ] = $col;
  }
}

interface DrawingObject
{
  function drawObject( $env );
  function setParam( $name, $value );
}

function loadModules( $dir )
{
  $classes = array( );
  $dh = new DirectoryIterator( $dir );
  foreach( $dh as $file )
  {
    if( $file->isDir( ) == 0 && preg_match( "/[.]php$/", $file ) )
    {
      include_once( $dir."/".$file );
      $class = preg_replace( "/[.]php$/", "", $file );
      $classes []= $class;
    }
  }
  return $classes;
}

$classes = loadModules( "mods" );
$dom = new DOMDocument( );
$dom->load( $argv[1] );
$nl = $dom->getElementsByTagName( "image" );
$root = $nl->item( 0 );
$size_x = $root->getAttribute( 'x' );
$size_y = $root->getAttribute( 'y' );
$file = $root->getAttribute( 'file' );
$de = new DrawingEnvironment( $size_x, $size_y );
```

```
$obs_spec = array( );
$el = $root->firstChild;
while( $el != null )
{
  if ( $el->tagName != null )
  {
    $params = array( ):
    for( $i - 0: $i < $el->attributes->length; $i++ )
    {
      $p = $el->attributes->item( $i )->nodeName;
      $v = $el->attributes->item( $i )->nodeValue;
      Sparams[ $p ] - $v;
    }
    $obs_spec []= array(
      'type' => $el->tagName.
      'params' => $params
    );
  }
  Sel - $el->nextSibling;
}

foreachC $obs_spec as $os )
{
  $ob = null;
  eval( '$ob = new '.$os['type'].'( ):' ):
  foreachC $os[ 'params' ] as Skey => lvalue )
  $ob->setParam( Skey. Svalue ):
  $ob->drawObject( Sde );
}
Sde->save( $file );
?>
```

Сохраните исходный код из примера 6.2 в файл `mods/Circle.php`.

Пример 6.2. Пример модуля, рисующего окружности

```
<?pnp
class Circle implements DrawingObject
{
  private Sradius = null;
  private $color = null;
  private Sx - null;
  private $y = null;
  function drawObject( Senv )
  {
    $r2 = $this->radius / 2;
    imagefilledellipse( $env->image( ),
      Sthis->x - Sr2. Sthis->y - Sr2.
      $this->radius, $this->radius.
      $env->color( $this->color )
    );
  }
  function setParam( $name. Svalue )
  {
    if ( Sname == "radius" ) Sthis->radius = Svalue;
    if ( Sname == "color" ) $this->color = Svalue;
    if ( Sname == "x" ) Sthis->x = Svalue;
```

```
    if ( $name — "y" ) $this->y = $value;
  }
}
?>
```

Запуск трюка

Запуск этого трюка производится из командной строки. Первым делом необходимо создать тестовый файл XML:

```
<image x='100' y='100' file='out.png'>
  <Circle x='20' y-'40' color='red' radius='15' />
  <Circle x='60' y-'30' color='green' radius='30' />
  <Circle x='70' y='75' color='blue' radius='35' />
</image>
```

В этом файле указано, что изображение должно быть размером 100 x 100 пикселов и иметь название out.png, а также что на нем должны быть нарисованы три окружности, причем все различных цветов и размеров.

Создав этот XML, запустите сценарий следующим образом:

```
% php raodhost.php test.xml
```

Первый аргумент, который передается в сценарий, — это имя XML-файла, содержащего характеристики изображения. Полученный в результате файл изображен на рис. 6.1.

Рис. 6.1. Полученное изображение

Чтобы немного объяснить, что здесь происходит, позвольте мне сперва рассказать о файле modhost.php. В его начале я определил класс DrawingEnvironment, который на самом деле является всего лишь оберточным классом с несколькими ссылками на объекты более низкого уровня. Эта среда будет передаваться в любые другие объекты, предназначенные для рисования, чтобы они могли размещать результат прямо в изображении. Следующее, что нас может заинтересовать, это интерфейс DrawingObject, объекты которого должны быть адаптированы для рисования. Самое хитрое происходит именно в функции loadModules(), которая загружает все модули из указанной директории в среду PHP. Затем сценарий считывает предоставляемый ему XML-файл и производит его анализ, разместив содержимое файла в объекте $obs_spec — массиве, выступающем в роли хранилища содержимого XML-файла. Следующим шагом создается среда для рисования и объекты для рисования исходя из значений, хранимых в $obs_spec. На базе этих значений затем строится изображение. И наконец, изображение сохраняется в файл. На рис. 6.2 показаны связи между объектами типа DrawingEnvironment и DrawingObjects, а также то, как динамически загружаемый класс Circle реализует интерфейс DrawingObject.

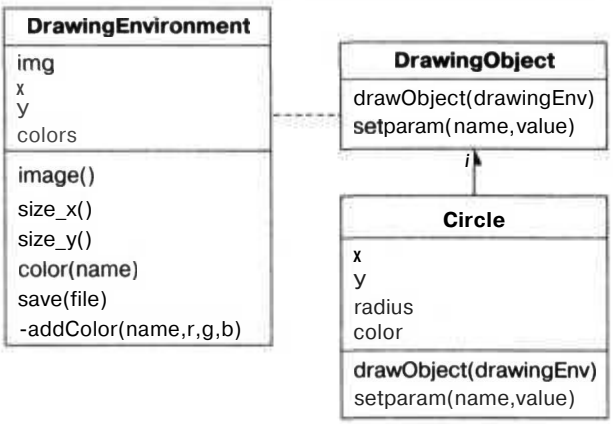


Рис. 6.2. Структура системы рисования

Здесь предоставлена только простая реализация этой техники.

ПРИМЕЧАНИЕ

Из-за спецификации интерфейса данный сценарий предназначен для PHP 5, но функции include_once() и eval() были и в PHP 4. Придется внести некоторые изменения, но нет причин, по которым вы не сможете создать что-то похожее на PHP 4.

Я очень рекомендую добавлять механизм расширений, подобный вышеописанному, во все PHP-приложения, которые есть смысл в дальнейшем улучшать, особенно если вы ожидаете, что они будут использоваться в различных средах, которые вы не контролируете. Это позволит конечным пользователям настроить программу исходя из их требований, причем вам не придется заново заходить прямо в код для создания каждой из новых возможностей или типов объектов, в которых возникла необходимость.

Смотрите также

- «Отслеживание ваших объектов» (см. трюк 67).
- «Создание объектов при помощи Абстрактной фабрики» (см. трюк 68).
- «Выделение кода создания структур при помощи Строителя» (см. трюк 70).

ТРЮК №52

Поддержка кода из Вики

Позвольте вашим пользователям вводить стилизованный текст в ваши приложения, используя поддержку синтаксиса из Вики.

Новая форма системы управления содержанием в Интернете, также известная как *Вики,* — это набор страниц, каждая из которых озаглавлена как WikiWord —

два или больше слова, написанные с заглавной буквы и без пробелов. Простота, с которой вы можете обновить Вики, сделала ее очень популярной как во внутренних сетях, так и в Интернете. Наверное, одной из самых известных Вики является Википедия (Wikipedia) (http://www.wikipedia.org/). Это сетевая энциклопедия, каждый из посетителей которой может добавить в нее новые записи, используя всего лишь свой веб-браузер. Другая причина, по которой Вики такие популярные, — способ форматирования страниц, используемый в них. Оно осуществляется намного проще, чем при написании эквивалентного кода на HTML. Например, вы оканчиваете параграф, просто нажав два раза кнопку ввода, причем нет необходимости добавлять теги <p>. На самом деле в Вики теги вообще почти никогда не используются. Например, для создания списка с маркерами вы просто помещаете знак звездочки в начале каждой строки, что намного проще использования эквивалентных тегов <ul> и <li>. В этом трюке продемонстрирован модуль PEAR для использования форматирования в стиле Вики в приложениях на PHP.

Код

Сохраните исходный код из примера 6.3 в файл index.php.

Пример 6.3. Страница, позволяющая редактировать текст в формате Вики

```
<html>
<body>
<form method="post" action="render.php">
<textarea name="text" cols="80" rows="20">
+ Header Level 1
Here's a paragraph and a link to AnotherPage.
* list item 1
* list item 2
Link to a NewPage like this.
</textarea><br/>
<input type="submit" />
</form>
</body>
</html>
```

Затем сохраните исходный код из примера 6.4 в файл render.php.

Пример 6.4. Код на PHP, выводящий на экран текст в формате Вики

```
<html>
<body>
<?php
// Подключаем библиотеку Wiki Text Pear
require_once( "Text/Wiki.php" );
// Создаем объект Wiki
$wiki = new Text_Wiki( ):
// Выводим поле с текстом, которое нам прислали в форме
echo( $wiki->transform( $_POST["text" ], 'Xhtml' ) ):
?>
</body>
</html>
```

Запуск трюка

Для этого кода вам понадобится модуль PEAR `Text_Wiki` (см. трюк 2). После установки модуля и создания PHP-файлов откройте в браузере страницу `index.php`, как это показано на рис. 6.3. Введите какой-либо Вики-текст в форму и нажмите кнопку Submit (Отправить). При использовании такого же текста, как и в этом примере, результат будет похож на изображенный на рис. 6.4.

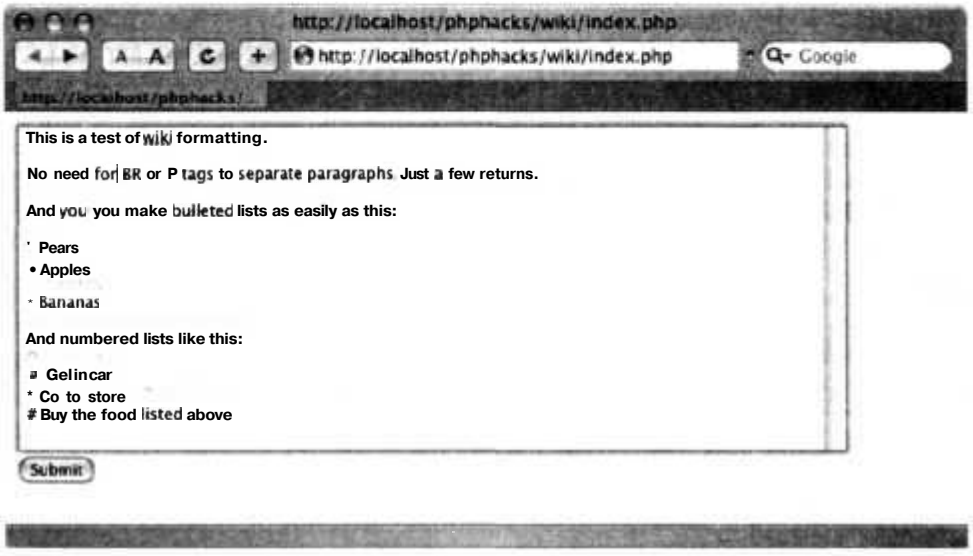


Рис. 6.3. Страница ввода текста для отображения его в формате Вики

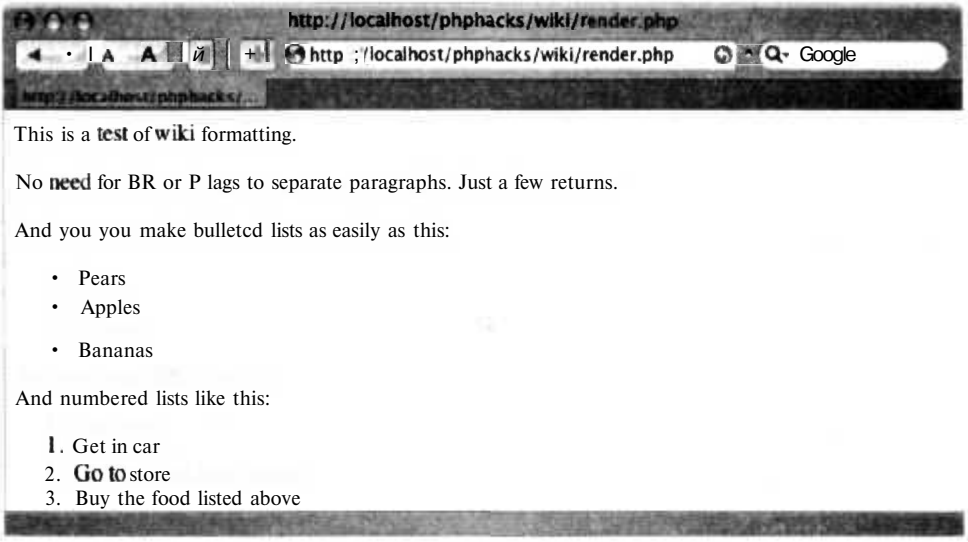


Рис. 6.4. Отображенный в формате Вики текст

ПРИМЕЧАНИЕ

Полный список правил форматирования для Вики расположен на домашней странице компонента Text_Wiki PEAR по адресу http://wiki.ciaweb.net/yawiki/index.php?area=Text_Wiki&page=WikiRules.

Улучшаем трюк

Модуль Text_Wiki очень хорошо спроектирован и написан. Можно добавлять новые правила форматирования текста, а уже существующие можно включать и отключать в зависимости от потребности приложения. Методы enableRule() и disableRule() служат для подключения или отключения встроенных правил форматирования текста. Метод же addRule() используется для добавления новых правил в средства для форматирования.

Преобразование любого объекта в массив

Используйте интерфейс Iterator в PHP 5, чтобы преобразовать любой объект в массив.

Если вы когда-либо использовали интерфейс DOM для чтения или записи XML в PHP, то уже знакомы с интерфейсом DOMNodeList. При использовании DOM многие из его методов возвращают в качестве результата массив узлов. Этот массив заполняется объектом типа DOMNodeList. Чтобы прочитать список узлов, вам необходимо написать код, похожий на следующий:

```
$dl - $doc->getElementsByTagName( "foo" ):
for( $i - 0; $i < $dl->length; $i++ )
{
  $n = $dl->item( $i ):
  ...
}
```

Это выглядит как-то неудачно, особенно с учетом того, что у PHP есть красивый оператор foreach, предоставляющий доступ к массивам практически без риска запутаться в них. Разве было бы плохо, если бы интерфейс DOM выглядел так?

```
foreach($doc->getElementsByTagName( "foo" ) as $n ) {
...
}
```

Такая запись намного понятнее и менее подвержена ошибкам. Благодаря дополнениям в PHP 5, теперь мы можем использовать оператор foreach применительно к любым объектам, просто разместив этот класс в интерфейсе Iterator. В этом трюке я покажу вам, как внедрить шаблон Наблюдатель (см. трюк 67) при помощи интерфейса Iterator.

Код

Сохраните исходный код из примера 6.5 в файл iterator.php.

Пример 6.5. Класс, использующий новый интерфейс Iterator в PHP 5

```
<?pnp
interface Listener
{
  public function invoke( Scalier, Sdata ):
}

class ListenerList implements Iterator
{
  private Slisteners = array( );
  public function _ _construct( )
  {
  }
  public function add( Slistener )
  {
    $this->listeners [] - Slistener:
  }
  public function invoke( Scalier. Sdata )
  {
    foreach( $this as Slistener )
    {
      $listener->invoke( Scalier, Sdata ):
    }
  }
  public function rewind( )
  {
    reset($this->listeners);
  }
  public function current( )
  {
    return current($this->listeners);
  }
  public function key( )
  {
    return key(Sthis->listeners):
  }
  public function next( )
  {
    return next($this->listeners):
  }
  public function valid( )
  {
    return ( $this->current( ) !- false );
  }
}

class SimpleListener implements Listener
{
  private Sv;
  public function___construct( Sv ) { $this->v = Sv; }
```

```
  public function invoke( Scalier, $data )
  {
    echo( $this->v." invoked with with '$data'\n" );
  }
  public function___tostring( ) { return "Listener ".$this->v; }
}

$11 = new ListenerList( );
$ll->add( new SimpleListener( "a" ) );
$ll->add( new SimpleListener( "b" ) );
$ll->add( new SimpleListener( "c" ) );
print("Listeners:\n\n");
foreach( $11 as $listener )
{
  print( $listener );
  print( "\n" );
}
print("\nInvoking Listeners:\n\n");
$ll->invoke( null, "Some data" );
?>
```

В первой части кода определяется интерфейс Listener для объектов, которые будут зарегистрированы в ListenerList. Во второй части определяется ListenerList, который на самом деле является всего лишь надстройкой над массивом с дополнительными методами addO и invokeO.

Все другие методы реализованы в интерфейсе Iterator. SimpleListener — это всего лишь реализация интерфейса Listener, которая просто предназначена для вывода сообщений. На рис. 6.5 показана модель исходного кода из этого трюка. ListenerList может быть либо пустым, либо содержать в себе несколько объектов, реализующих интерфейс Listener, в SimpleListener реализует интерфейс Listener, который при обращении к нему просто выводит сообщения.

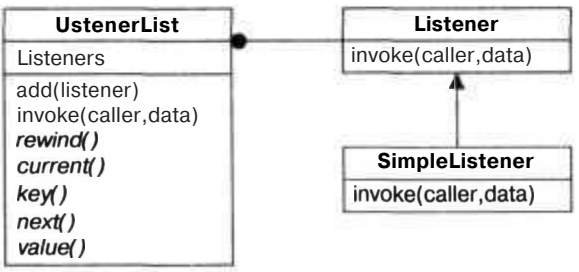


Рис. 6.5. Схема UML для UstenerList

Запуск трюка

Этот трюк следует запускать при помощи интерпретатора для командной строки:

```
% php iterator.php
Listeners:
Listener a
```

```
Listener b
Listener c
Invoking Listeners:
a invoked with with 'Some data'
b invoked with with 'Some data'
c invoked with with 'Some data'
%
```

Если вы посмотрите на код в самом конце примера 6.5, то увидите тестовые данные, которые здесь выводятся на экран. Первый тест предназначен для перебора списка при помощи оператора foreach. Его результат вы можете видеть в верхней части командной строки сразу после запуска. Далее следует результат вызова метода invokeO объектов из ListenerList.

Самое полезное в интерфейсе Iterator то, что теперь вы всегда можете передавать в него сложные интерфейсы, даже в тех случаях, в которых раньше вы могли использовать только массивы. Эти интерфейсы в виде массивов будут работать, как и раньше, но теперь они еще обладают дополнительными методами.

Смотрите также

- «Отслеживание ваших объектов» (см. трюк 67).

ТРЮК №54

Создание корректных XML

Используйте XML DOM для создания XML, не содержащих ошибки.

Создавая XML из ваших интернет-приложений на PHP, очень просто допустить ошибку. Вы можете перепутать кодировку, и в результате некоторые специальные символы будут отформатированы неправильно, или вы можете пропустить начальный или завершающий теги. Обе эти проблемы, которые частенько возникают даже в самых простых PHP-приложениях, в результате приведут к созданию некорректного XML, и потребитель не сможет правильно его обработать. Почти все проблемы такого рода — это результат работы с XML как с потоком символов, а не при помощи таких XML API, как DOM. В этом трюке будет показано, как создавать объекты типа XML DOM в памяти и затем экспортировать их в текстовом виде. Этот метод создания XML помогает избегать всех этих тонкостей с кодировками и форматированием, и благодаря этому ваш XML код всегда будет сформирован корректно.

На рис. 6.6 показано размещенное в памяти дерево XML, которое мы и создадим в этом трюке. Каждый его элемент является объектом. Основой системы является объект типа DOMDocument, на который указывает корневой узел дерева. Далее, каждый элемент узла типа DOMElement может обладать либо своими собственными детьми, либо узлами с атрибутами.

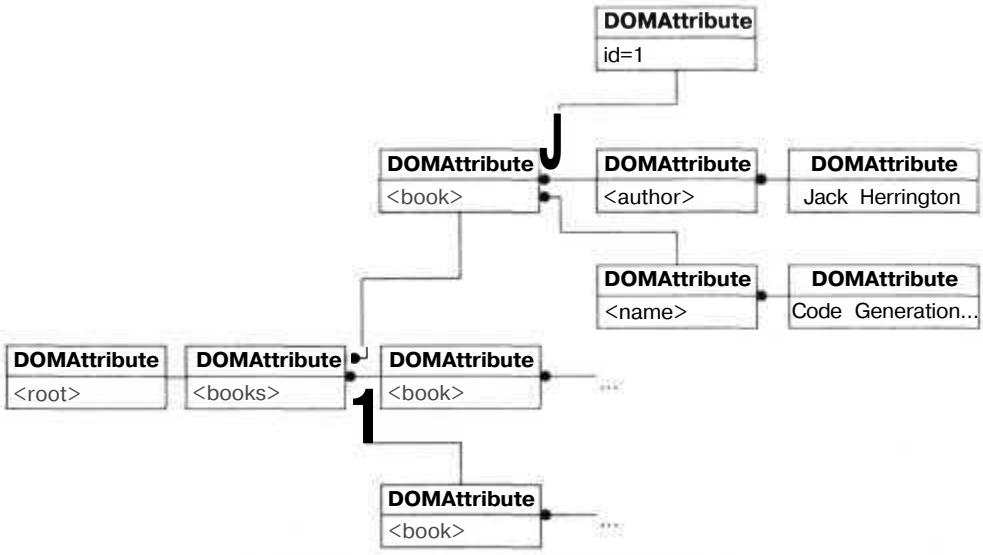


Рис. 6.6. Расположенное в памяти дерево XML

Код

Сохраните исходный код из примера 6.6 в файл `xmldom.php`.

Пример 6.6. Образец кода, корректно формирующего XML

```
<?php
$books - array(
  array (
    id => 1.
    author => "Jack Herrington".
    name -> "Code Generation in Action"
  ).
  array (
    id => 2.
    author -> "Jack Herrington".
    name -> "Podcasting Hacks"
  ).
  array (
    id => 3.
    author => "Jack Herrington".
    name => "PHP Hacks"
  )
);

$dom - new DomDocument( ):
$dom->formatOutput - true;

$root = $dom->createElement( "books" ):
$dom->appendChild( $root );

foreach( $books as $book )
```

```
{
  $bn = $dom->createElement( "book" );
  $bn->setAttribute( 'id', $book['id'] );
  $author = $dom->createElement( "author" );
  $author->appendChild( $dom->createTextNode( $book['author'] ) );
  $bn->appendChild( $author );
  $name = $dom->createElement( "name" );
  $name->appendChild( $dom->createTextNode( $book['name'] ) );
  $bn->appendChild( $name );
  $root->appendChild( $bn );
}

header( "Content-type: text/xml" );
echo $dom->saveXML();
?>
```

Запуск трюка

Загрузите этот файл на ваш сервер и перейдите к странице `xmldom.php` (рис. 6.7). Перед вами четко отформатированный и корректно сформированный XML, причем мне не пришлось добавлять в него вручную ни одного тега или значения какого-либо атрибута. Вместо этого созданием объекта занимался DOM, причем он же организовывал и связи между ними при помощи метода appendChildO. И наконец, для экспорта XML в виде текста используется метод `saveXML()`. Перед нами простой объектно-ориентированный способ создания корректно работающего в любых ситуациях XML.

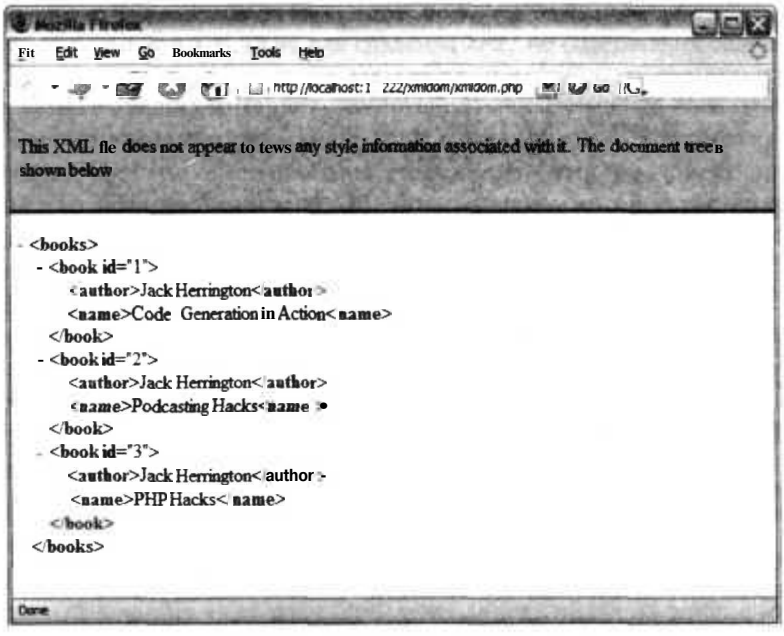


Рис. 6.7. XML-книга в браузере Firefox

Смотрите также

- «Разработка более качественных схем SQL» (см. трюк 34).
- «Создание неприступных баз данных» (см. трюк 35).

ТРЮК №55

Исправление проблемы повторной передачи данных

Используйте таблицу транзакций в вашей базе данных, чтобы избавиться от классической проблемы повторной передачи данных.

Я слегка раздражаюсь, когда сталкиваюсь с плохим дизайном интернет-приложений. И больше всего это касается изобилия плохо написанного кода для исправления проблемы повторной передачи данных. Как часто вы сталкивались с тем, что на сайтах, предоставляющих коммерческие услуги, вас просят не нажимать дважды кнопку Submit (Отправить)? Такого рода проблемы возникают, когда браузер дважды отправляет содержимое веб-формы на сервер. Тем не менее, если пользователь дважды нажимает кнопку Submit (Отправить), браузер как раз и должен это сделать, а выявлять ошибочность такого действия — дело сервера.

Рассмотрим проблему повторной отправки данных (рис. 6.8). Браузер отправляет два запроса, так как пользователь дважды нажимает кнопку. Первый запрос принимается, и еще до того, как будет возвращен ответ, отправляется еще один запрос. После этого приходит первый ответ, а за ним второй. На рис. 6.9 проиллюстрировано, как избавиться от проблемы повторной передачи данных: во время первого запроса в той странице, которая передается, сохраняется его уникальный ID. В этом случае, когда приходит второй ответ с тем же ID, дублирующая транзакция отменяется.

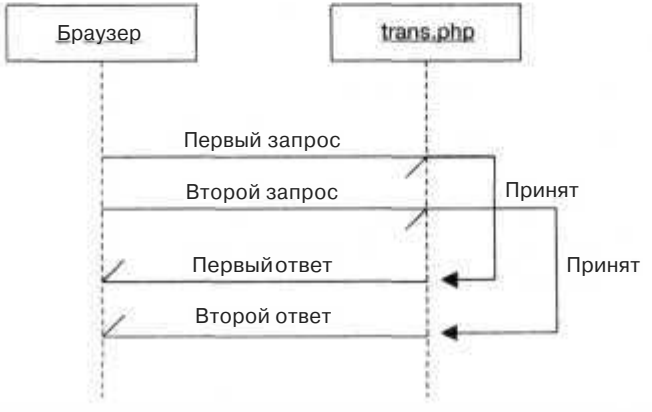


Рис. 6.8. Диаграмма последовательности действий при возникновении проблемы повторной передачи данных

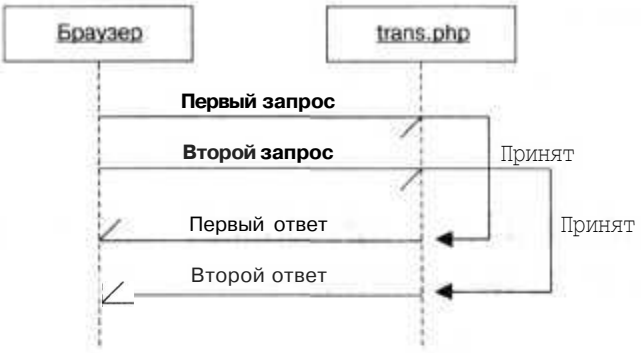


Рис. 6.9. Для решения проблемы повторной отправки данных требуется отменить второй запрос

Код

Сохраните исходный код из примера 6.7 в файл db.sql.

Пример 6.7. Код, используемый базой данных для проверки транзакций

```
DROP TABLE IF EXISTS transcheck;
CREATE TABLE transcheck (
  transid TEXT,
  posted TIMESTAMP
);
```

Сохраните исходный код из примера 6.8 в файл index.php.

Пример 6.8. HTML-форма, содержащая ID-транзакции

```
<? require_once( "trans.php" ); ?>
<html>
<body>
<form action="handler.php" method="post">
<input type="hidden" name="transid" value="<?php echo( get_transid( ) ); ?>" />
Name: <input type="text" /><br/>
Amount: <input type="text" size="5" /><br/>
<input type="submit" />
</format>
```

Сохраните исходный код из примера 6.9 в файл handler.php.

Пример 6.9. Код, получающий данные из формы и проверяющий транзакцию

```
<? require_once( "trans.php" ); ?>
<html>
<body>
<?php if ( check_transid( $_POST["transid"] ) ) { ?>
This form has already been submitted.
<?php } else {
add_transid( $_POST["transid"] );
?>
Ok, you bought our marvelous product. Thanks!
<?php } ?>
</body>
</html>
```

Сохраните исходный код из примера 6.10 в файл trans.php.

Пример 6.10. Библиотека для проверки транзакций

```
<?php
require_once( "DB.php" );
$dsn = 'mysql://root:password@localhost/transtest';
$db =& DB::Connect( $dsn, array( ) );
if (PEAR::isError($db)) {
  die($db->getMessage( ));
}

function check_transid( $id )
{
  global $db;
  $res = $db->query( "SELECT COUNT(transid) FROM transcheck WHERE transid=?",
  array($id) );
  $res->fetchInto($row);
  return $row[0];
}

function add_transid( $id )
{
  global $db;
        $sth = $db->prepare( "INSERT INTO transcheck VALUES( ?, now( ) )" );
        $db->execute( $sth, array( $id ) );
}

function get_transid( )
{
  $id = mt_rand( );
  while( check_transid( $id ) ) { $id = mt_rand( ); }
  return $id;
}
?>
```

Запуск трюка

Загрузите файлы на сервер, а затем воспользуйтесь командой mysql для загрузки схемы db.sql в вашу базу данных:

```
mysql --user=myuser --password=mypassword mydb < db.sql
```

Далее откройте в браузере страницу index.php, и вы увидите простейшую форму, часто используемую на коммерческих сайтах, как это показано на рис. 6.10.

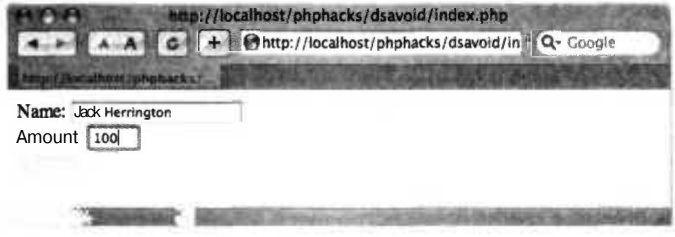


Рис. 6.10. Форма, используемая на коммерческих страницах

Введите какие-либо фиктивные данные и нажмите кнопку **Submit** (Отправить). Результат вы можете видеть на рис. 6.11, где показана удачная транзакция. Для начала это очень хорошо, ведь мы удостоверились, что можем удачно выполнять транзакции. Теперь перейдем к отмене дублирующей транзакции.

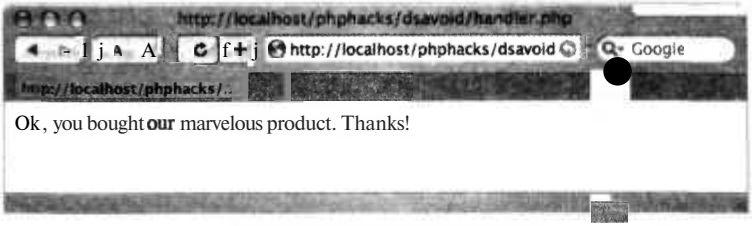


Рис. 6.11. Удачная покупка

Нажмите кнопку **Back** (Назад), а затем снова кнопку Submit (Отправить). Результат вы можете видеть на рис. 6.12.

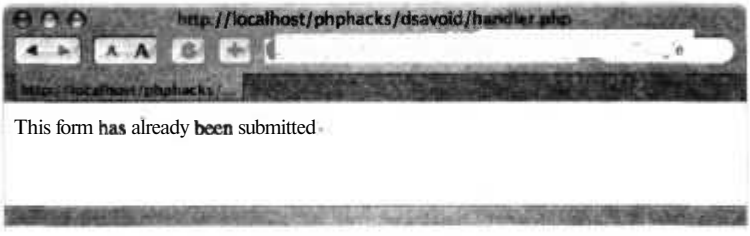


Рис. 6.12. Результат повторной передачи данных

Произошло следующее: файл `index.php` запросил уникальный ID из сценария `trans.php`. Сценарий `handler.php`, получающий значения переменных из формы, первым делом проверяет ID при помощи функции check_transid(), чтобы удостовериться, что он еще не был использован. Результат работы кода, когда ID уже был использован, вы можете увидеть на рис. 6.12. Если в базе данных этот ID отсутствует, мы используем функцию `add_transid()`, чтобы добавить в базу новый ID и сообщить пользователю, что передача данных была завершена удачно, как это показано на рис. 6.11. Хитрый читатель, возможно, возразит мне. Если другая форма отправит данные в промежутке между работой функций `check_transid()` и `add_transid()`, то вы можете столкнуться со случаем повторной передачи данных, причем он будет корректным. Если ваша база данных поддерживает хранимые процедуры, то можете написать одну транзакцию, которая будет проверять, была ли транзакция завершена, и добавлять ее в список выполненных транзакций. Это поможет избежать наложения транзакций друг на друга и удостовериться, что у вас не будет случаев повторной передачи данных.

ПРИМЕЧАНИЕ

Ко времени написания этой книги MySQL не поддерживал хранимых процедур, однако эта возможность была в списке предусмотренных в следующих релизах.

ТРЮК №56

Создание отчетов с использованием пользовательских настроек

Используйте механизм PHP для создания отчетов, который на базе XML-файлов с определениями создает указанный отчет.

Механизмы для создания отчетов позволяют настраивать генерируемые ими отчеты в соответствии с требованиями пользователя. Эта возможность очень важна для корпоративных приложений, так как эти системы редко полностью удовлетворяют пользовательские нужды. Возможность слегка подправить отчеты, сообщения либо другие способы вывода информации на экран очень важна с точки зрения удовлетворения всех пожеланий пользователя.

Механизм создания отчетов позволяет пользователям применять описательную методику при их формировании. Главная страница формирует запрос, получает данные и затем запускает механизм создания отчетов для форматирования данных. Некоторые механизмы создания отчетов, такие как **RLIB** (http://rlib.sf.net/), могут экспортировать отчеты не только **в HTML,** но также в PDF, XML и другие форматы. В этом трюке для создания простого отчета о книгах я использую систему PHPReports (http://phpreports.sf.net/).

Код

Сохраните исходный код из примера 6.11 в файл index.php.

Пример 6.11. Код на PHP, предназначенный для запуска генератора отчетов

```
<?php
require_once( "PHPReportMaker.php" );
$rep = new PHPReportMaker( );
$rep->setUser( "root" );
$rep->setPassword( "" );
$rep->setDatabaseInterface( "mysql" );
$rep->setConnection( "localhost" );
$rep->setDatabase( "books" );
$rep->setSQL( "SELECT NAME,AUTHOR FROM BOOK ORDER BY NAME" );
$rep->setXMI_( "bookreport.xml" );
$rep->run( );
?>
```

Сохраните код из примера 6.12 в файл bookreport.xml.

Пример 6.12. Спецификация отчета в XML-формате

```
<REPORT MARGINWIDTH="5" MARGINHEIGHT="5">
<TITLE>Book Report</TITLE>
<CSS>report.css</CSS>
<PAGE BORDER="0" SIZE="10" CELLSPACING="0" CELLPADDING="5">
</PAGE>
<GROUPS>
<GROUP NAME="author" EXPRESSION="AUTHOR">
<HEADER>
```

```
<ROW>
  <COL CELLCLASS="header"><XHTML><i>Name</i></XHTML></COL>
  <COL CELLCLASS="header">Author</COL>
</ROW>
</HEADER>
<FIELDS>
<ROW>
  <COL TYPE="FIELD">NAME</COL>
  <COL TYPE="FIELD">AUTHOR</COL>
</ROW>
</FIELDS>
<FOOTER>
</FOOTER>
</GROUP>
</GROUPS>
</REPORT>
```

Сохраните код из примера 6.13 в файл `report.css`.

Пример 6.13. CSS-таблица стилей для отчета

```
.header { font-weight: bold; }
```

Запуск трюка

Загрузите систему формирования отчетов PHPReports и установите ее согласно поставляющейся с ней инструкции. Загрузите файлы на сервер и откройте в браузере страницу `index.php` (рис. 6.13).

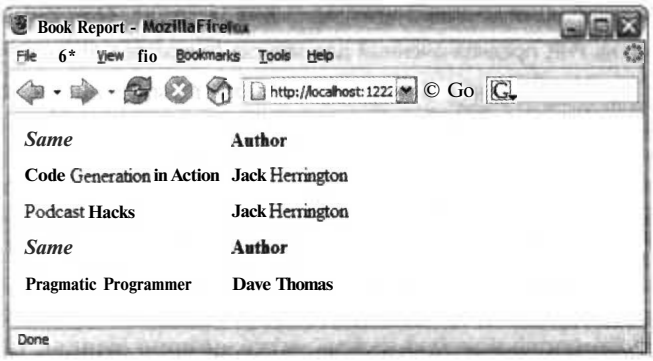


Рис. 6.13. Отформатированный отчет о книгах

Изменить внешний вид отчета просто, так как для этого надо всего лишь внести изменения в XML-код в файле `bookreport.xml`. Система PHPReports позволяет контролировать через HTML и CSS цвет, шрифты и компоновку отчета. Она также позволяет группировать данные при помощи элементов динамического HTML, таких как якорные теги и сценарии. Система PHPReports обладает гибким внутренним интерфейсом, позволяющим отображать на одной единственной HTML-странице текст или даже многостраничный отчет на HTML (HTML разбивается на несколько страниц, и у сгенерированной разметки внизу страницы появляется

возможность навигации по ним). Вы также можете создать свой собственный плагин, позволяющий вам формировать выходные документы любого вида.

Смотрите также

- «Дайте вашим пользователям доступ к управлению форматированием при помощи XSL» (см. трюк 7).

ТРЮК №57

Создание систем авторизации

В прочной системе авторизации нуждается любое сложное многопользовательское приложение.

Любое многопользовательское интернет-приложение нуждается в системе авторизации пользователей. Вы можете использовать механизм авторизации Apache, который при попытке получить доступ к странице отображает всплывающее окно с полями, предназначенными для введения имени пользователя и пароля, но это означает, что вам придется внедрять этот механизм авторизации в ваше приложение и в вашу базу данных. И, к сожалению, это означает, что вы не сможете управлять процедурой входа пользователя в систему и не сможете добавить возможность выбора пункта Я забыл свой пароль либо ссылку на контакты. На рис. 6.14 показано взаимодействие страницы с системой авторизации. Пользователь начинает работу со страницы входа в систему `index.php`. Далее в сценарии `login.php` проверяются предоставленные пользователем данные для авторизации.

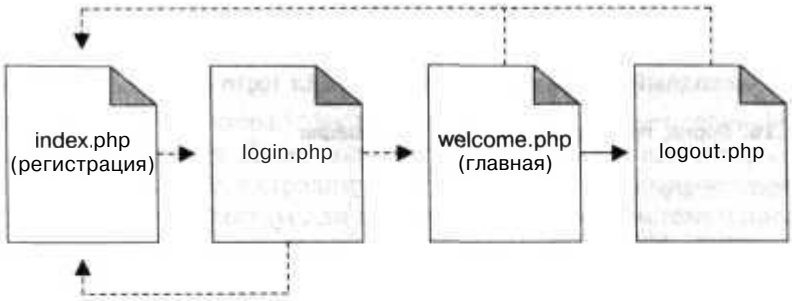


Рис. 6.14. Механизм работы системы авторизации

Если проверка при помощи файла `login.php` выявила корректность введенных данных, то пользователь начинает сеанс работы и перенаправляется на страницу `welcome.php`. Там он может выбрать ссылку logout (Выход из системы), которая вернет его к сценарию `logout.php`, закончит его рабочий сеанс и, наконец, отобразит исходную страницу `index.php`. Если пользователь попытается напрямую без авторизации через URL получить доступ к странице `welcome.php`, то она это обнаружит и перенаправит хитрого пользователя назад к странице авторизации `index.php`.

Код

Сохраните исходный код из примера 6.14 в файл users.sql.

Пример 6.14. База данных с заданными в ней пользователями

```
DROP TABLE IF EXISTS users;
CREATE TABLE users (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  name TEXT,
  password TEXT,
  PRIMARY KEY( id )
);

INSERT INTO users VALUES ( 0, 'jack', MD5( 'toronto' ) );
INSERT INTO users VALUES ( 0, 'megan', MD5( 'seattle' ) );
```

Сохраните исходный код из примера 6.15 в файл с именем index.php.

Пример 6.15. Страница авторизации

```
<html>
<head><title>Login</title></head>
<body>
<?php if ( $_GET['bad'] = 1 ) { ?>
<font color="red">Bad login or password, please try again<br/></font>
<?php } ?>
<form action="login.php" method="post">
<table width="300" border="0" cellspacing="0" cellpadding="2">
<tr><td>User name:</td><td><input type="text" name="user" /></td></tr>
<tr><td>Password:</td><td><input type="password" name="password" /></td></tr>
<tr><td colspan="2"><center><input type="submit" value="Login" /></center></td></
tr>
</table>
</form>
</body>
</html>
```

Сохраните исходный код из примера 6.16 в файл login.php.

Пример 6.16. Форма, предназначенная для авторизации

```
<?php
require_once( "DB.php" );
$dsn = 'mysql://root:password@localhost/time';
$db =& DB::Connect( $dsn, array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )); }
$res = $db->query( "SELECT id FROM users WHERE name=? AND password=MD5(?)",
array( $_POST['user'], $_POST['password'] ) );
$row = array( null );
if ( $res != null )
  $res->fetchInto( $row );
if ( $row[0] != null )
{
  session_start( );
  $_SESSION['user'] = $row[0];
  header( "Location: welcome.php" );
}
else
```

```
{
  header( "Location: index.php?bad=1" );
}
?>
```

Сохраните исходный код из примера 6.17 в файл welcome.php.

Пример 6.17. Домашняя страница для пользователей

```
<?php
session_start( );
if ( $_SESSION['user'] == null || $_SESSION['user'] < 1 )
{
  header( "Location: index.php" );
  exit:
}
require_once( "DB.php" ):
$dsn - 'mysql://root:password@localhost/time';
$db =& DB::Connect( $dsn. array( ) ):
if (PEAR::isError($db)) { die($db->getMessage( )): }
$res = $db->query( "SELECT name FROM users WHERE id=?".
array( $_SESSION['user'] ) );
$res->fetchInto( $row );
?>
<html>
<head><title>Welcome</title></head>
<body>
Welcome <?php echo( $row[0] ); ?><br/><br/>
<a href="logout.php">Logout</a>
</body>
</html>
```

Сохраните исходный код из примера 6.18 в файл с именем logout.php.

Пример 6.18. Обработчик выхода из системы

```
<?php
session_destroy( ):
header( "Location: index.php" ):
?>
```

Трюк начинается со страницы index. php, которая представляет собой форму для авторизации пользователя. Далее пользователь вводит свое имя и пароль, и данные из формы передаются в страницу login.php, которая посылает запрос в базу данных для проверки, существует ли такой пользователь в системе и подходит ли пароль. Если введенные данные верны, сценарий создает сеанс и направляет пользователя на страницу welcome. php, которая ведет себя как его домашняя страница. Оттуда пользователь может выйти из системы, щелкнув кнопкой мыши на ссылке, указывающей на страницу 1 ogout. php. Она и прекращает сеанс.

Запуск трюка

После загрузки файлов на сервер первым делом создайте базу данных пользователей:

```
% mysqladmin --user-root --password=password create time
% mysql --user=root--password=password time< users.sql
```

Первая команда создает базу данных. Вторая загружает SQL-сценарий в базу данных, создавая таким образом таблицу пользователей, и добавляет несколько учетных записей. Следующим шагом перейдем к странице `index.php`. Она должна выглядеть так, как на рис. 6.15.

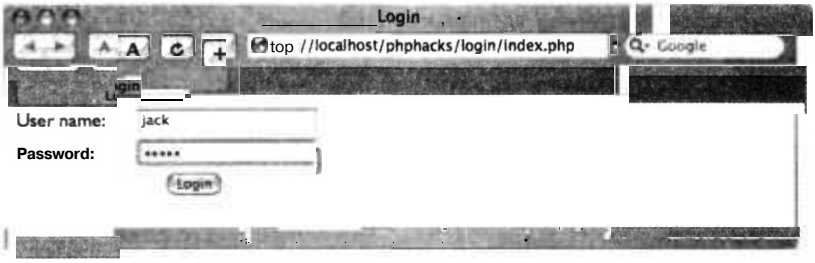


Рис. 6.15. Страница для авторизации

Для проверки страницы авторизации сперва введите неправильный пароль. В поле User name введите jack, а в качестве пароля введите слово `hel l o`, после чего нажмите кнопку Login. Страница `login.php` проверяет запрос на авторизацию, выясняет, что он некорректен, и вновь направляет пользователя на страницу `index.php`, третий параметр у которой равен 1. На странице также красным цветом отображается сообщение об ошибке, как это показано на рис. 6.16.

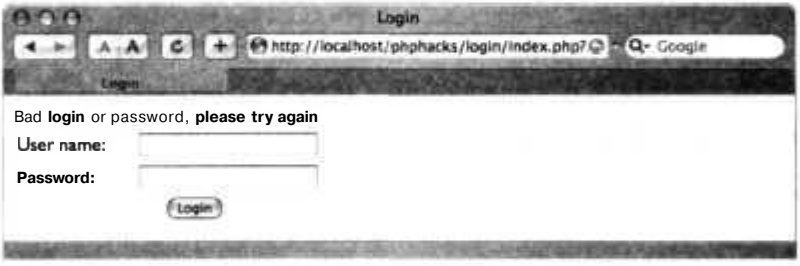


Рис. 6.16. Внешний вид страницы авторизации после того, как было введено неправильное имя пользователя или пароль

В этот раз в качестве имени пользователя введите jack, а в качестве пароля - toronto, после чего снова нажмите кнопку Login. Теперь страница `login.php` выяснила, что введенные данные верны, и она настраивает сеанс, учитывая корректный ID пользователя. Затем вас перенаправляют на страницу `welcome.php`. На ней вы можете видеть информацию об учетной записи пользователя, а также можете выйти из системы, как это показано на рис. 6.17. Если вы щелкнете кнопкой мыши на ссылке Logout, то страница `logout.php` завершит ваш сеанс и снова отправит вас к странице авторизации.

На базе этого простого каркаса системы авторизации вы можете создавать многопользовательские приложения, используя страницу `welcome.php` в качестве шаблона для других ваших страниц. Если вы будете использовать эту страницу в качестве отправной точки, то каждая следующая страница будет также проверять,

Рис. 6.17. Домашняя страница после удачной авторизации

авторизовался ли пользователь, а также будет переадресовывать его назад на страницу index.php в случае отсутствия с ним сеанса на данный момент.

Смотрите также

- «Применение систем безопасности на основе ролей» (см. трюк 58).

ТРЮК №58

Применение систем безопасности на основе ролей

Используйте системы безопасности на основе ролей для обеспечения варьирующегося уровня доступа к вашим интернет-приложениям.

Не все пользователи, которые пытаются получить доступ к системе, обладают в ней одинаковыми правами. Например, одни из них могут добавлять или удалять пользователей, другие могут отправлять, третьи — только читать сообщения, а четвертые могут делать все вышеперечисленное. Правильно организованная система безопасности на основе ролей не только запрещает доступ к некоторым частям системы, но также уменьшает сложность страниц для пользователей с ограниченными правами. Пользователю не следует видеть ссылки, которые ему не доступны исходя из его статуса в системе, но он должен иметь доступ к ссылкам, выполняющим разрешенные для него функции. В этом трюке вам будет продемонстрирована довольно прямолинейная система безопасности на основе ролей.

На рис. 6.18 показано взаимодействие между различными страницами в этом трюке. Пользователь начинает работу со страницы index.php, на которой размещается вход в систему. Она передает данные в страницу login.php, которая проверяет данные авторизации. Если данные корректны, то пользователь авторизуется и переходит к странице welcome.php. Если же данные неверны, то пользователь вновь возвращается к странице index.php. Находясь на странице welcome.php, пользователь может сделать следующее. Он может выйти из системы, щелкнув кнопкой мыши на ссылке к странице logout.php, которая прекращает сеанс и возвращает пользователя к странице index.php, или же перейти прямо к странице manage.php. Там проверяются введенные им для авторизации данные, и если пользователь не обладает достаточными правами, то он возвращается к странице welcome.php.

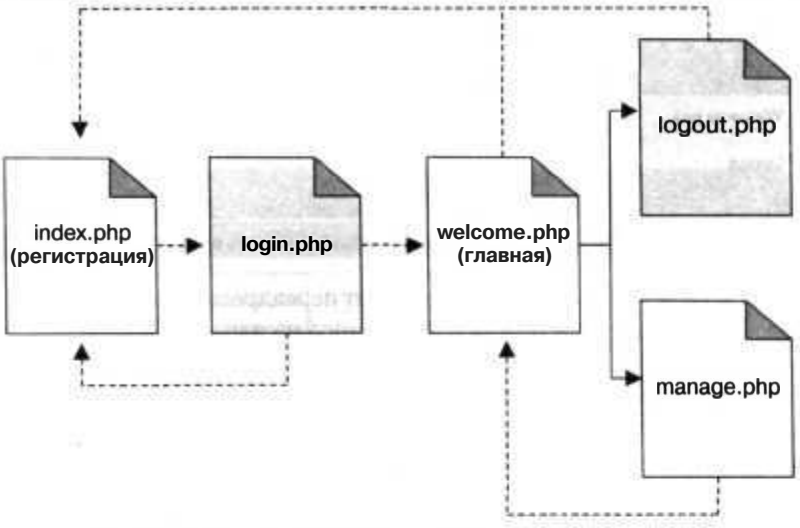


Рис. 6.18. Взаимодействие между страницами в этом трюке

Страница welcome.php также проверяет, действителен ли сеанс с пользователем. Если это не так, то пользователь переадресовывается назад к странице index.php, чтобы заново пройти авторизацию. Другими словами, пользователи не могут обойти авторизацию и получить доступ к сайту, просто набрав прямой URL.

Код

Сохраните исходный код из примера 6.19 в файл с именем dblib.php.

Пример 6.19. Библиотека функций для работы с базой данных, предназначенной для организации системы безопасности на основе ролей

```
<?php
require_once( "DB.php" );
$dsn - 'mysql://root:password@localhost/roles';
$db =& DB::Connect( $dsn, array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )); }

function get_db( ) { global $db; return $db; }
function get_role_id( $role )
{
  global $db;
   $res - $db->query( "SELECT id FROM roles WHERE name=?", array( $role ) );
  if ( $res !- null )
  {
    $res->fetchInto( $row );
    return $row[0];
  }
} return null;

function has_role( $user, $role )
```

```
{
  global $db;
    $role_id = get_role_id( $role );
    $res - $db->query( "SELECT user_id FROM user_role WHERE user_id=?
      AND role_id=?",
  array( $user, $role_id ) );
  if ( $res != null )
  {
    $res->fetchInto( $row );
    return ( (int)$row[0] == (int)$user ) ? true : false;
  }
  return false;
}
?>
```

Сохраните исходный код из примера 6.20 в файл с именем index.php.

Пример 6.20. Страница, предназначенная для авторизации

```
<html>
<head><title>Login</title></head>
<body>
<?php if ( $_GET['bad'] == 1 ) { ?>
<font color="red">Bad login or password, please try again<br/></font>
<?php } ?>
<form action="login.php" method="post">
<table width="300" border="0" cellspacing="0" cellpadding="2">
<tr><td>L)ser name:</td><td><input type="text" name="user" /></td></tr>
<tr><td>Password:</td><td><input type="password" name="password" /></td></tr>
<tr><td colspan="2"><center><input type="submit" value="Login" /></center></td></
tr>
</table>
</form>
</body>
</html>
```

Сохраните исходный код из примера 6.21 в файл с именем login.php.

Пример 6.21. Обработчик введенных при авторизации данных

```
<?php
require_once( "dblib.php" );
$db = get_db( );
$res - $db->query( "SELECT id FROM users WHERE name=? AND password=MD5(?)",
array( $_POST['user'], $_POST['password'] ) );
$row - array( null );
if ( ires != null )
  $res->fetchInto( $row );
if ( $row[0] != null )
{
  session_start( );
  $_SESSION['user'] = $row[0];
  header( "Location: welcome.php" );
}
else
{
  header( "Location: index.php?bad=1" );
}
?>
```

Сохраните исходный код из примера 6.22 в файл с именем logout.php.

Пример 6.22. Код, предназначенный для выхода из системы

```
<?php
session_destroy( ):
header( "Location: index.php" ):
?>
```

Сохраните исходный код из примера 6.23 в файл с именем manage.php.

Пример 6.23. Страница, видимая только менеджерам

```
<?php
require_once( "security.php" ):
session_start( );
if ( $_SESSION['user'] — null || $_SESSION['user'] < 1 )
{ header( "Location: index.php" ): exit: }
check_roles( $_SESSION['user'], array( 'manager' ) );
?>
<html>
<body>
From here you manage the users.<br/><br/>
Back to the <a href="welcome.php">home page</a>.
</body>
</html>
```

Сохраните исходный код из примера 6.24 в файл с именем security.php.

Пример 6.24. Библиотека функций, обеспечивающих безопасность

```
<?php
require_once( "dblib.php" ):

function check_roles( $user, $roles )
{
  foreach( $roles as $role )
  {
    if ( !has_role( $user, $role ) )
    {
      ?>
      You do not have permission to access this page.<br/><br/>
      Return to the <a href="welcome.php">home page</a>.
      <?php
      exit;
    }
  }
}
?>
```

Сохраните исходный код из примера 6.25 в файл с именем users.sql.

Пример 6.25. Код на SQL, используемый в этом примере

```
DROP TABLE IF EXISTS roles:
CREATE TABLE roles (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  name TEXT,
  PRIMARY KEY( id )
);

DROP TABLE IF EXISTS users:
```

```
CREATE TABLE users (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  name TEXT,
  password TEXT,
  PRIMARY KEY( id )
);

DROP TABLE IF EXISTS user_role;
CREATE TABLE user_role (
  user_id MEDIUMINT,
  role_id MEDIUMINT
);

INSERT INTO roles VALUES ( 0, 'user' );
INSERT INTO roles VALUES ( 0, 'manager' );
INSERT INTO users VALUES ( 0, 'jack', MD5( 'toronto' ) );
INSERT INTO users VALUES ( 0, 'megan', MD5( 'Seattle' ) );
INSERT INTO user_role VALUES ( 1 , 1 ) ;
INSERT INTO user_role VALUES ( 2, 1 );
INSERT INTO user_role VALUES ( 2, 2 );
```

Сохраните исходный код из примера 6.26 в файл с именем welcome.php.

Пример 6.26. Домашняя страница для пользователей, вошедших в систему

```
<?php
require_once( "dblib.php" );
session_start( );
if ( $_SESSION['user'] ==null || $_SESSION['user'] < 1 )
{ header( "Location: index.php" ); exit; }
$db - get_db( ):
$res = $db->query( "SELECT name FROM users WHERE id-?". array( $_SESSION['user']
) ):
$res->fetchInto( $row );
?>
<html>
<head><title>Welcome</title></head>
<body>
Welcome <?php echo( $row[0] ): ?><br/><br/>
<?php
if ( has_role( $_SESSION['user'], 'manager' ) ) {
?>
<a href="manage.php">Manage the users</a><br/><br/>
<?php } ?>
<a href="logout.php">Logout</a>
</body>
</html>
```

Для начала система загружает страницу index.php, которая представляет собой форму для авторизации пользователей. Когда пользователь вводит свое имя и пароль, форма передает эти данные на страницу login.php, которая продолжает авторизацию. Если авторизация прошла успешно, то создается сеанс с пользовательским ID. Далее пользователь переходит к странице welcome.php, которая после прохождения авторизации является его домашней страницей. На странице welcome.php есть ссылка на другую страницу с именем manage.php. Она должна быть доступна, если вы являетесь менеджером, поэтому код проверяет наличие у вас соответствующих прав. Если пользователь не выступает в роли менеджера, то он

переадресовывается назад на домашнюю страницу. Последнее, что здесь расположено, это ссылка на страницу для выхода из системы. Это происходит при помощи сценария logout.php, который завершает сеанс с пользователем и возвращает его назад к странице для авторизации.

Код вверху каждой из страниц системы должен проверять, авторизован ли пользователь. Если это так, то страница должна отображаться нормально. Иначе пользователь должен быть переадресован назад на страницу авторизации. Очевидно, что на странице авторизации этого кода нет. Иначе пользователь просто попадет в бесконечный цикл в попытке авторизоваться.

Запуск трюка

Для запуска этого трюка первым делом необходимо загрузить базу данных:

```
% mysqladmin --user=root --password=password create roles
% mysql --user=root --password=password roles < users.sql
```

Когда это будет сделано, загрузите остальные файлы с исходным кодом на сервер и откройте в браузере страницу index.php. Результат должен выглядеть так, как на рис. 6.19.

Рис. 6.19. Авторизация менеджера с именем пользователя **megan**

Авторизуйтесь, используя в качестве имени пользователя слово megan, а в качестве пароля — seattle, и в результате вы увидите домашнюю страницу, показанную на рис. 6.20.

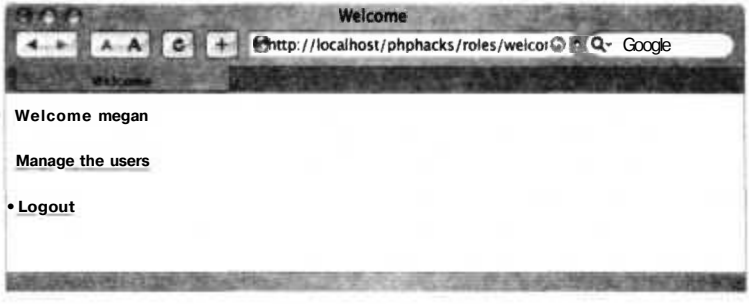


Рис. 6.20. Домашняя страница Megan со ссылкой на управление пользователями

Ссылка на страницу управления учетными записями пользователей (`management.php`) отображается, так как учетная запись `megan` выступает в роли менеджера, и эта страница проверяет роль, используя для этого функцию `has_role()`.

Щелкните кнопкой мыши на ссылке **Manage the users** (рис. 6.21).

Рис. 6.21. Доступна ссылка для управления учетными записями

Эта страница удостоверяется, что тот, кто видит ее, является менеджером. В данном случае пользователь **Megan,** который авторизовался, является менеджером, поэтому он имеет доступ к странице. Теперь выйдем из системы, а затем снова авторизуемся, введя имя пользователя jack, как это показано на рис. 6.22. Пароль для этой учетной записи — `toronto`.

Рис. 6.22. Авторизация пользователя jack, не являющегося менеджером

В этот раз на домашней странице нет ссылки на страницу управления учетными записями пользователей, что вы можете видеть на рис. 6.23.

Рис. 6.23. Домашняя страница без возможностей управления

Даже сообразительный пользователь, который попытается перейти прямо к странице `management.php`, набрав прямой URL вручную, получит в результате страницу

с сообщением о том, что у него недостаточно прав для просмотра этой страницы. Вы можете увидеть это сообщение об ошибке на рис. 6.24.

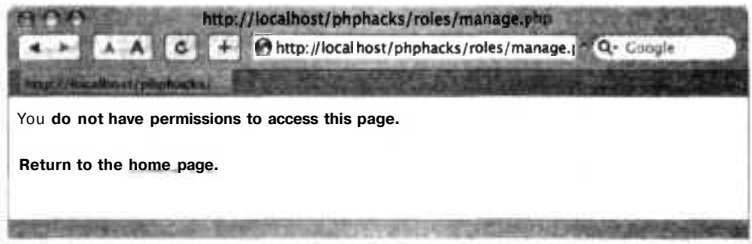


Рис. 6.24. Отказ в прямом доступе к странице управления

Это очень грубый пример системы обеспечения безопасности на основе ролей. Если ваше приложение маленькое или уровни доступа разделяются всего лишь на две или три роли, то система такого типа может вам подойти.

Следующим шагом надо создать объектную модель внутри приложения с учетом ролевой системы. Затем вы можете организовать различные варианты доступа: для чтения, добавления, удаления и обновления каждого из объектов.

Исходя из этого добавьте возможность назначать группы вариантов доступа определенным именам ролей. Для примера, пользователь в роли administrator будет иметь возможность читать, добавлять, удалять и обновлять пользовательские учетные записи. Роль manager должна будет иметь возможность доступа к обновлению учетных записей. Исходя из логики работы система безопасности будет уточнять, имеет ли данный пользователь какие-то определенные права доступа, и, изучив роль пользователя, будет видеть, обладает ли она этими возможностями доступа. Приложение не ищет какую-то определенную роль, так как на самом деле она является всего лишь набором из нескольких вариантов доступа.

Смотрите также

- «Создание систем авторизации» (см. трюк 57).

ТРЮК №59

Переход к паролям MD5

Используйте переходный сценарий для преобразования ваших паролей, хранимых в виде открытого текста, в зашифрованные при помощи MD5.

Имея за плечами годы работы в качестве консультанта, я могу вам с уверенностью сказать, что, хотя люди и говорят вам, что в их интренет-приложениях используются зашифрованные пароли, на самом деле чаще всего это не так. Хотя в действительности сделать это не так уж и сложно. Более того, сайты, которые могут выслать вам ваш пароль в виде текста, если вы щелкнете кнопкой мыши на ссылке Я забыл ваш пароль, хранят их где-нибудь в незашифрованном виде. Нет необходимости говорить, что это плохо.

Так почему же шифрование паролей так важно? Во-первых, любой, кто получит доступ к базе данных через дыру в безопасности, может получить доступ ко всей системе. Во-вторых, многие говорят о необходимости использования различных паролей для различных учетных записей, но в итоге используются либо такие же, либо очень похожие пароли, так как это намного проще. Получив пароль для одной из машин, вы сможете получить доступ к другим, возможно, более важным учетным записям. В этом трюке рассказано, как зашифровать при помощи MD5 таблицу пользователей и паролей, хранимую в открытом виде.

Код

Сохраните исходный код из примера 6.27 в файл schema.sql.

Пример 6.27. Файл с исходной базой данных

```
DROP TABLE IF EXISTS users;
CREATE TABLE users (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  name TEXT,
  pass TEXT,
  PRIMARY KEY( id )
);
```

Сохраните исходный код из примера 6.28 в файл с именем users.sql.

Пример 6.28. Пароли в незашифрованном виде

```
INSERT INTO users VALUES ( 0, "jack", "toronto" );
INSERT INTO users VALUES ( 0, "megan", "omaha" );
```

Сохраните исходный код из примера 6.29 в файл с именем migrate.php.

Пример 6.29. Сценарий преобразовывает пароли, хранимые в виде открытого текста, в зашифрованные при помощи MD5

```
<?php
require_once( "DB.php" );

$dsn = 'mysql://root:password@localhost/migpass';
$db =& DB::Connect( $dsn, array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )); }
$res = $db->query( "SELECT id, pass FROM users", array( ) );
$sth = $db->prepare( "UPDATE users SET pass=MD5(?) WHERE id=?" );
while( $res->fetchInto( $row ) )
{
  $db->execute( $sth, array( $row[1], $row[0] ) );
}
?>
```

Сохраните исходный код из примера 6.30 в файл с именем list.php.

Пример 6.30. Сценарий, перечисляющий пользователей

```
<?php
require_once( "DB.php" );
$dsn = 'mysql://root:password@localhost/migpass';
$db =& DB::Connect( $dsn, array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )); }
$res = $db->query( "SELECT id, name, pass FROM users", array( ) );
```

```
$sth - $db->prepare( "UPDATE users SET pass-MD5(?) WHERE id-?" ):
while( $res->fetchInto( $row ) )
{
  print( $row[0]." - ".$row[l]." - ".$row[2]."\n" );
}
?>
```

Сохраните исходный код из примера 6.31 в файл с именем check.php.

Пример 6.31. Сценарий, проверяющий, зашифрованы ли пароли

```
<?pnp
require_once( "DB.php" ):
$dsn = 'mysql://root:password@localhost/migpass';
$db =& DB::Connect( $dsn. array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )): }
$user = "jack":
$pass = "toronto";
$res = $db->query( "SELECT id. name FROM users WHERE name=? AND pass=MD5(?)",
array( $user. $pass ) ):
while( $res->fetchInto( $row ) )
{
  print( $row[0]." - ".$row[l]."\n" ):
}
?>
```

В этом коде на самом деле нет ничего сверхъестественного — в PHP есть поддержка преобразования в функции MDB(), которая и используется в сценарии migrate.php.

Запуск трюка

В этом трюке используется PHP-интерпретатор для командной строки, поэтому работу начнем с загрузки через командную строку базы данных со схемой, а также добавим в нее несколько пользователей:

```
% mysqladmin --user=root--password=password create migpass
% mysql --user=root--password=password migpass < schema.sql
% mysql --user=root--password=password migpass < users.sql
```

Первой командой мы создаем базу данных. Вторая команда предназначена для создания таблицы пользователей в созданной базе данных, а в последней строке мы добавляем несколько тестовых учетных записей пользователей.

Затем нам следует убедиться, что данные были добавлены корректно:

```
% php list.php
1 - jack - toronto
2 - megan - omaha
```

В результате мы видим две учетных записи из таблицы. Первая принадлежит пользователю **jack** с паролем **toronto,** она хранится в незашифрованном виде. Вторая учетная запись — пользователя **megan** с паролем **omaha**. Следующим шагом преобразуем пароли, используя PHP-сценарии, и посмотрим, что у нас получится:

```
% php migrate.php
% php list.php
1 - jack - 79cca97018f48e834a46f1b634e9a427
2 - megan - c365303299c8e35dbd443faa065feb5f
```

Первой командой мы преобразуем текст паролей в таблицах при помощи функции `MD5()` в MySQL. Сценарий же `list.php` после этого выводит содержимое базы данных с зашифрованными паролями. И, наконец, необходимо удостовериться, что мы по-прежнему можем авторизовать пользователей, используя в SQL-запросах функцию MD5:

```
% php check.php
1 - jack
```

Сценарий `check.php` пытается авторизоваться, используя имя пользователя jack и пароль toronto. Вместо использования пароля `toronto` в незашифрованном виде необходимо преобразовать его при помощи функции MD5 и сравнить с хранимым в базе паролем. При обработке одного и того же текста функцией MD5 результат всегда будет одинаковым, и благодаря этому можно сравнивать зашифрованные при помощи MD5 пароли. Поскольку в открытом виде пароли пользователей больше не хранятся ни в приложении, ни в базе данных, то, если клиент забудет пароль, приложение больше не сможет выслать ему пароль в виде обычного текста. В дополнение к сценарию, реализующему преобразование, в интернет-приложении придется также предусмотреть возможность работы службы **Я забыл свой пароль.** Обычно в таких случаях пароль сбрасывается, устанавливается другой — безопасный (и случайный) — и отсылается на почтовый ящик, указанный в данной учетной записи. Почтовые сообщения такого рода должны предусмотрительно содержать в себе ссылку, щелкнув кнопкой мыши на которой пользователь сможет сменить пароль на другой, более простой для запоминания (доступный хотя бы в течение пары дней), причем срок действия высланного пароля также должен прекращаться через несколько дней.

Смотрите также

- «Создание систем авторизации» (см. трюк 57).
- «Применение систем безопасности на основе ролей» (см. трюк 58).

ТРЮК №60

Создавайте рабочие URL при помощи mod_rewrite

Используйте модуль Apache mod_rewrite для создания простых в понимании и использовании адресов URL.

Модуль сервера Apache `mod_rewrite` позволяет вам прозрачно переадресовывать URL один на другой, причем пользователь не будет знать об этом. Это открывает богатые возможности, начиная от простой переадресации старых URL на новые адреса и заканчивая очисткой «грязных» URL (заполненных дополнительными параметрами и данными, которые приложение никогда не будет использовать), оставшихся в наследство от не очень качественных издательских систем, что в итоге делает URL более читабельным как для пользователей, так и для поисковых систем.

Кратко о переназначении

Читабельные адреса URL — это очень хорошо. Интернет-сайт с хорошо продуманным дизайном обладает логичной файловой системой с подходящими именами файлов и папок, а большинство тонкостей реализации должно оставаться за кадром. На еще более продуманных сайтах посетители могут даже догадаться, какое имя файла следует добавить к адресу, причем вероятность того, что они угадают, очень велика.

Тем не менее иногда, даже если дизайн вашего сайта находится на очень высоком уровне, практически невозможно использовать его URL. Например, вам понадобится воспользоваться системой управления содержанием применительно к URL, который выглядит следующим образом: http://www.site.com/viewcatalog.php?category=hats&prodID=53. Это просто ужасный URL, но в наши дни динамически генерируемых страниц они и их собратья начинают встречаться все чаще и чаще. Рассмотрим список проблем, которые часто встречаются при использовании URL такого рода.

- Раскрывается технология, использованная при создании этого веб-сайта (в нашем случае это PHP). Это может позволить потенциальным хакерам приблизительно понять, какого типа данные им следует отсылать в строке запроса, чтобы реализовать атаку на сайт прямо с парадного входа. Вам не следует разглашать информацию такого рода, если можно этого избежать.

ПРИМЕЧАНИЕ

Даже если вы не сильно озабочены обеспечением безопасности сайта, не следует выставлять напоказ использованные вами технологии. В худшем случае это сбивает с толку ваших посетителей, поэтому вам следует стараться скрывать все лишнее. Если по какой-то причине в будущем вы решите изменить язык программирования, на котором основан ваш сайт (на ASP, например), то все ваши старые URL перестанут работать. Это довольно-таки серьезная проблема, и любой человек, столкнувшийся с необходимостью полностью переписать сайт, вам это подтвердит.

- URL включает в себя трудночитаемую пунктуацию: знаки вопроса и амперсанды. В частности, символы & и являются проблемой — если другой веб-мастер создаст ссылку на страницу, используя этот URL, и не уберет символы амперсандов, его XHTML не будет работать.
- Некоторые поисковые системы не будут учитывать страницы, которые, как они считают, сгенерированы автоматически. Из-за опасности столкнуться с бесконечным количеством страниц, получаемых изменением строки запроса URL, многие поисковые механизмы задуманы так, чтобы избегать добавления таких страниц в каталог.

К счастью, при помощи переименования можно легко конвертировать эти URL во что-нибудь легко управляемое. Например, вы можете преобразовать URL в http://www.site.com/catalog/hats/53/. Не правда ли, намного лучше? Этот URL более логичный, читабельный и запоминающийся, к тому же его примет любая поисковая система. Ложные директории используют короткие имена плюс ко всему

они более информативные. В качестве дополнительного преимущества такая запись выглядит более долговечной. Для использования mod_rewrite вам необходимо передать ему на обработку адреса URL, которым вы хотите организовать замену (это так называемые «грязные» URL, о которых шла речь выше), и настоящие URL, на которые первые будут переадресовываться. URL, замену которым следует подобрать, могут быть путем к файлу, что будет соответствовать одному файлу, или это могут быть регулярные выражения, что может соответствовать сразу нескольким файлам.

Основы переназначения

На некоторых серверах модуль mod_rewrite по умолчанию не запущен. Однако этот модуль содержится в установке Apache, и вы можете запустить его, просто создав файл с настройками для Apache. Создайте файл с именем .htaccess (или откройте его, если он уже существует) и поместите его в корневую директорию, позволив таким образом использовать переназначение для всего вашего сайта. После того как вы создадите этот файл, добавьте в него следующую строку:

```
RewriteEngine on
```

Простая переадресация. Начнем с прямой переадресации. Она выглядит так, как если бы вы переместили файл и хотели бы, чтобы все ссылки на его старое местоположение перенаправлялись на его новый адрес. Код, реализующий простую переадресацию, выглядит следующим образом:

```
RewriteEngine on
RewriteRule ^old\.html$ new.html
```

Несмотря на то, что это очень простой пример, тем не менее он может многих сбить с толку. Единственным сложным моментом в RewriteRule является структура old в URL. В ней используются три специальных символа.

- Символ ^ (перенос каретки) указывает на начало URL для замены, расположенного в подпапках директории, содержащей файл .htaccess. Если вы опустите символ ^, то исходный код воспримет также, например, файл с именем cold.html. Именно из-за такого рода неопределенности вам следует всегда использовать этот символ.
- Знак $ указывает на конец строки, которой необходимо организовать замену. Вам следует использовать его, чтобы избежать ситуации, когда к строке может добавиться первая часть более длинных URL.
- Символ ., размещенный перед расширением файла, — это специальный символ, используемый в регулярных выражениях в качестве служебного, если только мы не отменим это при помощи обратного слеша, сказав таким образом Apache, что это обычный символ.

Итак, благодаря указанному выше правилу сервер напрямую переадресует вас со страницы old.html на страницу new.html. Ваши посетители не будут знать об этом, к тому же все происходит почти мгновенно.

Вынужденное формирование новых запросов. Иногда вам бывает необходимо, чтобы пользователь знал о переадресации, и вы можете сделать это, создав новый HTTP-запрос для новой страницы. В результате браузер загрузит новую страницу, как если бы ее на самом деле и запрашивали, причем в строке адреса также будет отображаться URL новой страницы. Все, что вам надо сделать, — добавить к правилу параметр [R]:

```
RewriteRule ^old\.html$ new.html [R]
```

На рис. 6.25 показана переадресация в действии.

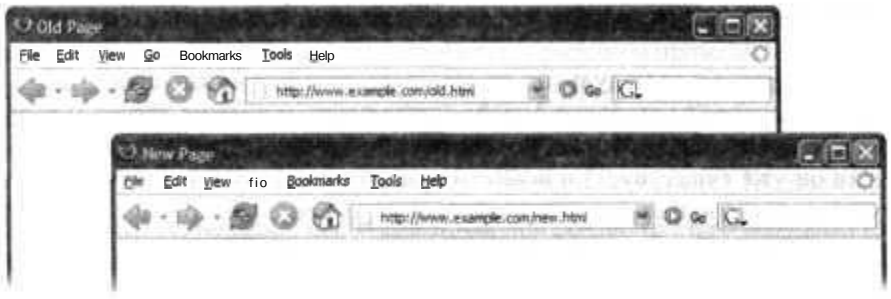


Рис. 6.25. Переадресация пользователя с old.html на new.html

Использование регулярных выражений

Теперь приступим к изучению действительно полезных возможностей. Мощь mod_rewrite прямо пропорциональна его сложности. Если вы сейчас в первый раз столкнулись с регулярными выражениями (см. трюк 87), то они могут показаться непростыми, но богатство предоставляемых ими возможностей стоит того, чтобы изучить их досконально и стать настоящим профи.

Благодаря регулярным выражениям вы можете создавать правила, позволяющие организовывать массовую переадресацию сразу нескольких URL. Это бывает очень полезно при создании больших сайтов с множеством страниц, генерируемых при помощи одного единственного PHP-файла. Например, вы разработали ваш сайт таким образом, чтобы структура URL выглядела так: http://www.example.com/articles/<artide_id>. Она довольно ясна и хороша. Тем не менее, если все ваши статьи генерируются при помощи одного сценария show_article.php, как это часто и бывает, вам захочется настроить переадресацию каждого из URL на его настоящее месторасположение. Воспользуйтесь следующим правилом:

```
RewriteRule ^articles/([0-9][0-9])/$ show_article.php?articleID=$1
```

В этом случае вы организуете переадресацию для всех URL, начинающихся с articles/ и у которых дальше следуют две цифры и слеш. Например, благодаря этому правилу будут созданы соответствующие адреса для таких URL, как articles/12/ или articles/99/, и они будут перенаправлены в PHP-сценарий (рис. 6.26).

Часть выражения, расположенная в квадратных скобках, называется *диапазоном*. В данном случае мы позволяем анализировать диапазон чисел символов 0-9,

Рис. 6.26. Переадресация с чистого URL на статью, расположенную в системе PHP

то есть любые цифры. Могут быть другие диапазоны, например [A-Z], включающий в себя все прописные буквы, [a-z] — для всех строчных букв, а также [A-Za-z], подразумевающий под собой все буквы в любом регистре.

В регулярное выражение помещена только часть URL, так как мы хотим дальше использовать полученные в результате поиска значения. В данном случае мы передаем это значение в PHP-сценарий в качестве аргумента. Когда мы получим какое-либо значение в квадратных скобках, то сможем использовать его для *обратных ссылок*. Каждой части, которую вы поместили в скобку, дается индекс начиная с 1. Таким образом, обратная ссылка имеет номер $1, третья — $3 и т. д. Иначе говоря, после переадресации загруженная в браузере страница будет иметь адрес, похожий на следующий: **show_artide.php?artideID=12.**

Добавление обратных слешей. Если посетитель вашего сайта ввел в строке браузера что-нибудь типа **articles/12,** то, несмотря на написанное выше правило, переадресации не будет, так как в конце URL отсутствует слеш. Чтобы учесть другие возможности написания URL, позаботьтесь об их переадресации на тот же URL, добавив слеш:

```
RewriteRule ^articles/([0-9][0-9])$ art1cles/$l/ [R]
```

Множественную переадресацию можно последовательно размещать в одном и том же файле .htaccess, чем мы и не преминули воспользоваться. Это правило устанавливается перед добавленным нами ранее следующим образом:

```
RewriteRule ^articles/([0-9][0-9])$ articles/$l/ [R]
RewriteRule ^articles/([0-9][0-9])/$ show_article.php?articleID=$1
```

В результате, если пользователь наберет в адресной строке URL вида articles/12, исходя из первого правила к нему будет добавлен обратный слеш и будет отправлен новый запрос по адресу **articles/12/.** Второе же правило соответственно перенаправит URL на **show_article.php?articleID=12.** Очень ловко, не правда ли?

Модификаторы соответствий. Вы можете расширить ваши шаблоны регулярных выражений, добавив некоторые символы, которые будут выступать в качестве модификаторов, что позволит вам организовывать соответствия для URL с неограниченным количеством символов. В более ранних примерах мы позволяли использовать только две цифры для номера ID каждой из статей. Это не самое

хорошее решение, так как, если количество опубликованных статей перевалит за 99, это приведет к тому, что такие URL, как show_article.php?articleID=100, наше правило не будет обрабатывать. Поэтому, вместо того чтобы строго забить количество символов, которые следует искать, мы немного расширим возможности правила, позволив обрабатывать адреса с любым количеством цифр. Новое правило делает следующее:

```
RewriteRule ^articles/([0-9]+)$ articles/$1/ [R]
```

Обратите внимание на закравшийся в нашу запись символ плюс (+). Этот модификатор относится к идущей прямо перед ним записи и означает «один или несколько символов из заключенного в квадратные скобки диапазона». В данном конкретном случае он означает, что правило будет организовывать соответствие для URL, которые начинаются с articles/ и заканчиваются хотя бы одной цифрой. Таким образом, оно подходит как для адреса articles/1, так и для articles/1000. Есть другие модификаторы, которые вы можете использовать, например звездочка (*), который означает «ноль или несколько символов из диапазона, заключенного в квадратные скобки», и знак вопроса (?), который означает «ноль или только один символ из диапазона, заключенного в квадратные скобки». Использование переназначения URL позволяет вашим пользователям реже сталкиваться с ошибками 404, да и вообще складывается впечатление, что ваш сайт работает более гладко.

Росс Шеннон (Ross Shannon)

1 ТРЮК №61 Создание переадресации для рекламы

Добавьте на ваш сайт возможность размещать на нем рекламу в случайном порядке между щелчками кнопкой мыши на ссылке.

Сайты с какими-либо материалами, например веб-сайт об играх IGN (http://www.ign.com/), также получают доходы, размещая рекламу. Если вы щелкнете кнопкой мыши на ссылке к статье, то либо можете увидеть ее содержимое, либо откроется страница с рекламой. На ней есть как ссылка на рекламу, так и ссылка на запрашиваемую вами статью (чтобы вы могли самостоятельно перейти к запрашиваемой вами странице). Страница с рекламой также автоматически может перенаправить вас к статье, если вы подождете несколько секунд.

ПРИМЕЧАНИЕ

Я должен признаться, что я дважды подумал, прежде чем написать и включить в книгу этот трюк, потому что мне совсем не нравится реклама такого рода. Но я пришел к выводу, что вы сами решите, использовать ли его. Это как в случае с «Поваренной книгой анархиста» — только факт того, что есть книга, в которой написано, как сделать бомбу, совершенно не означает, что вам обязательно нужно ее сделать.

На рис. 6.27 показано взаимодействие между страницами в системе переадресации для рекламы. Все ссылки на странице index.php ведут к странице redir.php,

предназначенной для переадресации. Исходя из полученного случайным образом значения страница redir.php решает, будете вы еще сидеть и смотреть рекламу или будете перенаправлены к запрашиваемой вами статье.

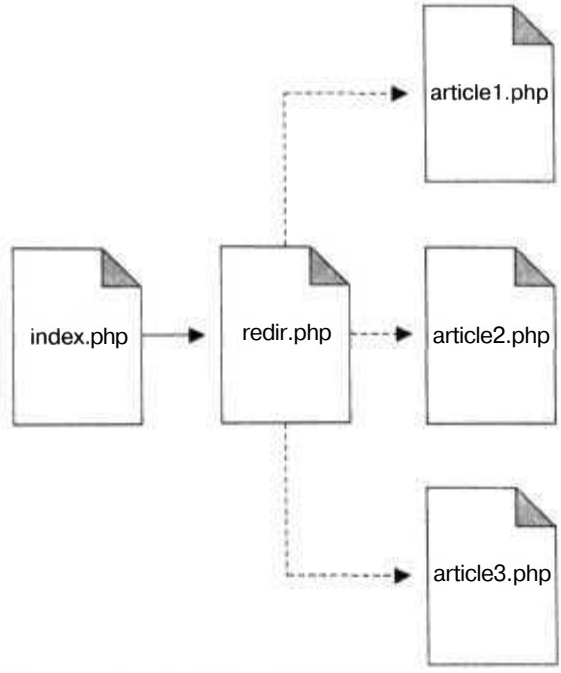


Рис. 6.27. Взаимодействие страниц в системе переадресации для рекламы

Код

Сохраните исходный код из примера 6.32 в файл с именем index.php.

Пример 6.32. Домашняя страница со ссылками на статьи

```
<?php
function redir_link( $url, $text )
{
  ?>
  <a href="redir.php?url=<?php echo( $url ); ?>"><?php echo( $text ); ?></a>
  <?php
}
?>
<html>
<body>
Here are my articles:<br/><br/>
<?php redir_link( 'article1.html', 'Article one' ); ?><br/>
<?php redir_link( 'article2.html', 'Article two' ); ?><br/>
<?php redir_link( 'article3.html', 'Article three' ); ?><br/>
</body>
</html>
```

Сохраните исходный код из примера 6.33 в файл с именем redir.php.

Пример 6.33. Переадресация на рекламу

```
<?php
srand(time( ));
if ( mt_rand(0.10) < 7 )
{
  header( "Location: ".$_GET['url'] );
  exit:
}
?>
<html>
<head>
<script language="Javascript">

function redir( )
{
  window.location='<?php echo( $_GET['url'] ): ?>';
}
function startTimer( )
{
  window.setTimeout( "redir( ):". 2000 );
}
</script>
</head>
<body onload="startTimer( )">
Here is my groovy ad. You can continue onto the article
<a href="<?php echo( $_GET['url'] ); ?>">here</a>. Or just
wait for a couple of seconds.
</body>
</html>
```

Сохраните исходный код из примера 6.34 в файл с именем article1.html.

Пример 6.34. Первая статья

```
<html>
<head><title>Article one</title></head>
<body>
  Article one
</body>
</html>
```

Сохраните исходный код из примера 6.35 в файл с именем article2.html.

Пример 6.35. Вторая статья

```
<html>
<head><title>Article two</title></head>
<body>
  Article two
</body>
</html>
```

Исходный код из примера 6.36 сохраните в файл с именем article3.html.

Пример 6.36. Третья статья

```
<html>
<head><title>Article three</title></head>
```

```
<body>
  Article three
</body>
</html>
```

Запуск трюка

Загрузите файлы на PHP-сервер и перейдите в браузере к странице index.php (рис.6.28).

Рис. 6.28. Домашняя страница со ссылками на статьи

Щелкните кнопкой мыши на ссылке Article two, которая на самом деле указывает на страницу redir.php. Иногда страница redir.php будет переадресовывать вас на страницу с рекламой, как это показано на рис. 6.29.

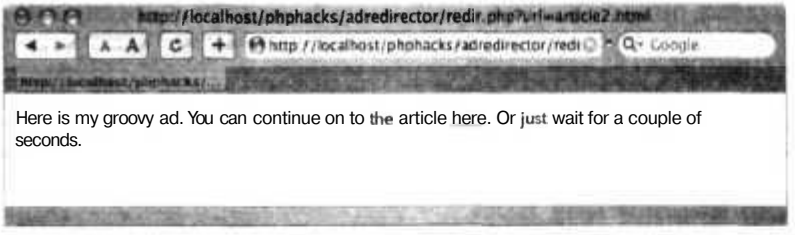


Рис. 6.29. Рекламная страница с таймером и ссылкой на статью

Если пользователи хотят пропустить рекламу, для этого есть прямая ссылка на статью. К тому же на странице есть таймер, который при окончании отсчета запускает переадресацию на страницу, запрошенную пользователем. Если реклама не появляется из-за сгенерированного случайного значения, то пользователь переходит сразу к нужной ему статье, как это показано на рис. 6.30. Есть несколько способов реализовать этот механизм переадресации рекламы. В рассмотренном нами варианте пользователь все еще может перейти напрямую к статье, получив на странице, предназначенной для переадресации, точный URL-адрес требуемой страницы. Если вы хотите, чтобы реклама появлялась даже при переходе по прямой ссылке, то можете внести соответствующие изменения, добавив проверку вверху каждой страницы в вашем приложении, которая будет в случайном порядке переадресовывать пользователей на рекламу, вместо того чтобы отображать страницу с требуемой статьей.

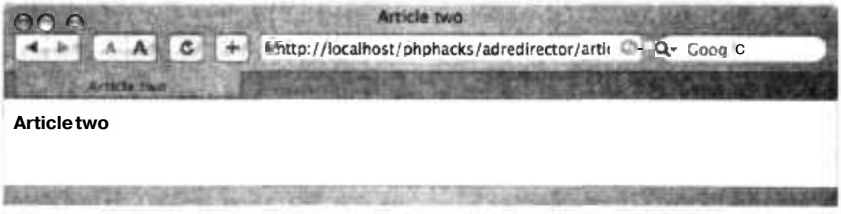


Рис. 6.30. Страница со статьей

ТРЮК №62

Добавляем кнопку Buy Now

Используйте платежную систему PayPal и добавьте для этого кнопку Buy Now в ваше интернет-приложение на PHP.

Кнопки PayPal Buy Now идеально подходят для развития вашей электронной торговли. Они просты в использовании, а также не требуют каких-либо длительных процедур проверки. Одно нажатие — и вы на защищенном сайте PayPal. Эти кнопки также очень просто устанавливать, однако именно из-за этого во многих интернет-приложениях отсутствует какой-либо дополнительный код, который бы следил за обеспечением защиты продаж. В этом трюке будет рассказано, как обеспечить безопасность покупок, сделанных при помощи кнопки Buy Now. Я покажу вам, как создавать кнопки Buy Now от начала и до конца, как их улучшить, чтобы при покупке в базе данных создавались соответствующие записи, а также расскажу вам, как обезопасить процесс покупок при помощи Instant Payment Notification (IPN).

ПРИМЕЧАНИЕ

Для этого трюка вам потребуется PHP 5 и MySQL версии 4.1.3 или выше, а также расширение mysqli.

Создание кнопки Buy Now

Вам не надо беспокоиться о том, что придется самолично создавать HTML-форму для кнопки Buy Now. Просто откройте вашу учетную запись на PayPal, выберите пункт Merchant Tools и щелкните кнопкой мыши на ссылке Buy Now Buttons. Через секунду вы получите HTML, готовый для размещения на вашем сайте. Вы можете узнать, как выглядит кнопка Buy Now, взглянув на рис. 6.31.

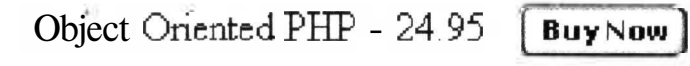


Рис. 6.31. Кнопка Buy Now

В таком виде у этой формы есть несколько недостатков. В ней отсутствует возможность отслеживания покупок при помощи базы данных, и, кроме того, как и любая другая форма в Интернете, она подвержена атакам взломщиков. Можно сделать локальную копию формы и изменить значения, чтобы нанести вред вашему сайту (и вашим клиентам). Более того, если не будет дополнительной провер-

ки значений из этой формы, можно запросто приобрести товар со скидкой. В файле buynow.html из примера 6.37 (показанном в сокращенном виде) содержится код на HTML для кнопки **Buy Now.** У него есть одно большое отличие от кода, сгенерированного системой PayPal: атрибут action тега `<form>` указывает не на сайт PayPal, а на другой сценарий — `presubmit.php` (показанный в примере 6.39). Этот промежуточный сценарий позволяет добавлять информацию в базу данных, прежде чем будут отправлены данные о покупке на PayPal.

Отслеживаем покупку

Если рассказывать в нескольких словах, то исходный код в сценарии `presubmit.php` добавляет запись в таблицу `tblorders` и в связанную с ней таблицу `tblorderitems`. Затем получается ID заказа, который добавляется к строке запроса, сформированной исходя из значений, переданных странице. И наконец, строка с запросом передается на сайт PayPal, как это показано на рис. 6.32.

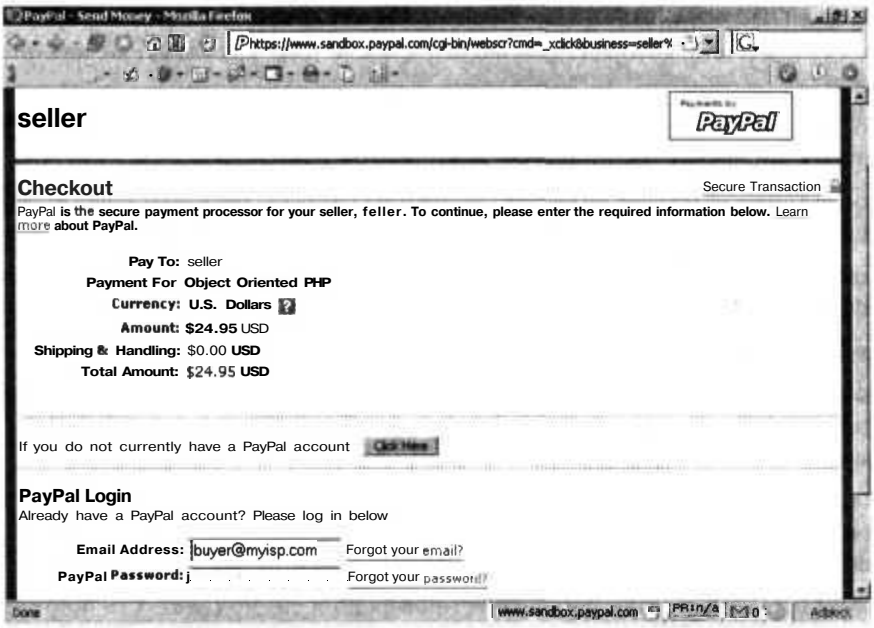


Рис. 6.32. Страница PayPal, предназначенная для проверки

Создание записи в базе данных и последующая передача номера заказа в систему PayPal поможет нам отследить покупку, а также обезопасить ее. Поскольку для доступа к базе данных используется относительно новый объектно-ориентированный интерфейс из расширения mysqli, то необходимо сделать небольшое отступление. Даже если у вас нет опыта в объектно-ориентированном программировании, то легко можно понять суть кода, добавляющего запись в таблицу заказов. Поскольку в таблице присутствует поле `auto_increment`, то после добавления в нее заказа необходимо получить ID, чтобы в дальнейшем при его помощи можно было идентифицировать покупку.

При добавлении же записи в таблицу tblorderitems мы уже не действуем так прямолинейно. Для создания записи в этой таблице код использует готовый оператор — в расширении mysqli поддерживается такая возможность MySQL 4.1. Готовые операторы обычно используются для добавления нескольких записей в базу данных и работают намного более эффективно, чем одиночные операторы SQL. Тем не менее в описанном здесь коде готовые операторы используются по причине того, что передаваемые в них данные не требуют предварительной обработки. Готовые операторы автоматически извлекают данные (отличная и очень удобная возможность, которой можно воспользоваться при случае).

Проверка продажи

Обеспечение безопасности при оплате подразумевает под собой необходимость удостовериться в том, что оплата была сделана с соответствующей учетной записи, что была перечислена нужная сумма и что это не повторная транзакция сделанного ранее заказа. IPN позволяет сделать все это программным способом, проанализировав URL, который получит сообщение об оплате. Рассмотрим последовательность действий, которая производится после того, как была сделана покупка на сайте.

1. После нажатия кнопки Buy Now запускается сценарий, создающий запись в базе данных и направляющий покупателя на сайт PayPal.
2. После того как PayPal получает платеж, в IPN URL передается скрытое сообщение. В нем содержится зашифрованный код и информация о платеже.
3. Чтобы удостовериться, что это сообщение не поддельное и что оно на самом деле было сформировано системой PayPal, необходимо вернуть его назад.
4. PayPal ответит, проверив сообщение и подтвердив, что оно пришло от него.

Сценарий verifypurchase.php (из примера 6.40) содержит код, который подтверждает источник сообщения и проверяет данные. Его работу можно описать следующим образом.

1. После того как от системы PayPal пришло сообщение, оно снова подготавливается к отсылке: к сообщению добавляется пара имя/значение cmd=_notify-validate, как этого требует PayPal.
2. Если переопределенное сообщение проверено и его подлинность подтверждена системой PayPal, то код подтверждает корректность дополнительных данных о покупке.
3. Стоимость какой-то определенной продукции получается из базы данных и сравнивается с принятым в сообщении значением.
4. Затем код производит проверку ID, чтобы удостовериться, что это не дублирующая транзакция сделанной ранее покупки, и, проверив адрес электронной почты пользователя, убеждается, что платеж был получен с правильной учетной записи.

В коде также содержится несколько дополнительных комментариев к менее понятным, на первый взгляд, деталям. Использование пакета CURL — не единствен-

ный возможный способ повторной передачи данных системе PayPal, но с его помощью это действительно можно сделать очень просто. При использовании IPN имена некоторых полученных значений отличаются от тех, которые были отправлены изначально. receiver_email — это синоним значению полю business для адреса электронной почты, получаемому из формы. В поле mc_gross хранится размер полученного платежа. В этом примере его значение должно совпадать со стоимостью покупки. С другой стороны, поля item_number и custom не претерпели никаких изменений. И снова доступ к базе данных производится при помощи объектно-ориентированного интерфейса из mysqli.

ПРИМЕЧАНИЕ

В зависимости от настроек сервера при создании объекта, предназначенного для соединения, могут либо понадобиться, либо не понадобиться настройки порта и сокета. Кроме того, стоит отметить, что вы можете задать базу данных при создании соединения. Поскольку этот код использует (и зависит от) PHP 5, то объекты автоматически передаются по ссылкам и нет никакой необходимости исправлять синтаксис, когда объекты выступают в роли параметров для функции.

Код

Сохраните исходный код из примера 6.37 в файл с именем buynow.html.

Пример 6.37. Пример страницы, использующей кнопку Buy Now

```
<html>
<head>
<title>Buy Now Button</title>
</head>
<body>
<!-- alter action of form-->
<form action="presubmit.php" method="post">
  <label>Purchase: Object-Oriented PHP for 24.95</label>
  <input type="hidden" name="cmd" value="_xclick" />
  <input type="hidden" name="business" value="seller@myisp.com" />
  <input type="hidden" name="item_name" value="Object-Oriented PHP" />
  <input type="hidden" name="item_number" value="673498" />
  <input type="hidden" name="amount" value="24.95" />
  <input type="hidden" name="no_note" value="1" />
  <input type="hidden" name="currency_code" value="USD" />
  <input type="image" src="https://www.paypal.com/en_US/i/btn/x-click-but23.gif"
  border="0" name="submit" alt-"" />
</form>
</body>
</html>
```

Сохраните исходный код из примера 6.38 в файл с именем connection.php.

Пример 6.38. Информация о соединении

```
<?php
  $hostname = "localhost";
  $databasename = "books";
  $username = "username";
  $password = "password";
?>
```

Исходный код из примера 6.39 сохраните в файл с именем presubmit.php.

Пример 6.39. Сценарий, сохраняющий заказ, прежде чем передать его системе PayPal

```
<?php
include "connection.php";
//Создаем соединение, используя mysqli
$con = new mysqli($hostname, $username, $password, $databasename,
3306, "/var/lib/mysql/mysql.sock");
//Создаем id заказа
$strsql = "INSERT INTO tblorders SET orderdate - CURDATE( )";
$con->query($strsql);
//Получаем insertid – свойство, а не метод
$id = $con->insert_id;
$item_number = $_POST['item_number'];
//Теперь добавляем заказ в базу данных
$strsql = "INSERT INTO tblorderitems SET orderid = ?, ".
"inventorynumber - ?";
//Используем команду, даже если добавляем только один элемент
$stmt - $con->stmt_init( );
$stmt->prepare($strsql);
//Присваиваем значения переменным типа integer и string
$stmt->bind_param('ii', $id, $item_number);
$stmt->execute( );
//Отправляем данные заново
$querystring = "?";
//Цикл для отправленных значений
foreach($_POST as $key -> lvalue)
{
 $value = urlencode(stripslashes($value));
  $querystring .- "$key=$value&";
}
//Обновляем id заказа в querystring
//Используем "custom", а не "on0" для значения id заказа
$querystring .= "custom=$id";
header('location:https://www.paypal.com/cgi-bin/webscr'.$querystring);
exit( );
?>
```

Сохраните исходный код из примера 6.40 в файл с именем verifypurchase.php.

Пример 6.40. Сценарий для проверки заказа

```
<?php
include "connection.php";
////////////////////////////////////////////////////////////////
function check_txnid($con, $txnid)
{
  $valid_txnid = false;
  //Получаем содержимое
  Sstrsql - "SELECT * FROM tblorders "
  " WHERE txnid = '$txnid'";
  $rs = $con->query($strsql);
  if($rs->num_rows == 0)
  {
    $valid_txnid = true;
  }
  return $valid_txnid;
```

```
}
////////////////////////////////////////////////////////////
function check_price($con, $price, $inventoryid)
{
  $valid_price - false:
  //Получаем содержимое
  $strsql - "SELECT listprice FROM tblbooks ".
  " WHERE inventorynumber - '$inventoryid'";
  $rs = $con->query($strsql);
  $row = $rs->fetch_array( );
  $num = (float)$row[0];
  if($num == $price)
  {
    $valid_price = true;
  }
  return $valid_price;
}
//////////////    //////////////    //////////IIIIIIII//////
function check_email($email)
{
  $valid_email = false;
  //Сравниваем с адресом электронной почты продавца в системе paypal
  if ($email == "seller@myisp.com" )
  {
    $valid_email = true;
  }
} return $valid_email;
////////////////////////////////////////////////////////////
function do_post($data)
{
  //Теперь отправляем назад на paypal
  $c - curl_init('https://www.paypal.com/cgi-bin/webscr');
  curl_setopt($c. CURLOPT_POST,1);
  curl_setopt($c. CURLOPT_POSTFIELDS, $data);
  curl_setopt($c, CURLOPT_SSL_VERIFYPEER,FALSE);
  curl_setopt($c. CURLOPT_RETURNTRANSFER, 1);
  $status = curl_exec($c);
  curl_close($c);
  return Sstatus;
}
////////////////////////////////////////////////////////////
//Цикл для отправленных значений
$data = "":
foreach($_POST as $key -> $value)
{
  lvalue = urlencode(stripslashes($value));
  $data .= "$key=$value&";
}
//Необходимо добавить это значение перед отправкой на paypal
Sdata .- "cmd=_notify-validate";
$status = do_post($data);
//Разбираем запрос
Sstatus = rtrim($status);
Spayment_status = $_POST['payment_status'];
```

```
//get transaction id
$txn_id = $_POST['txn_id'];
if ($status == "VERIFIED" && $payment_status == "Completed")
{
  //Эти переменные нам нужны
  $price = $_POST['mc_gross'];
  //Получаем номер заказа
  $orderid = $_POST['custom'];
  $inventoryid = $_POST['item_number'];
  //Адрес электронной почты продавца, то есть его учетная запись в системе paypal
  //Это то же самое, что и поле business в paynow.html
  $receiver_email = $_POST['receiver_email'];
  //Создаем новое соединение mysqli
  $con = new mysqli($hostname, $username, $password, $databasename, 3306,
  "/var/lib/mysql/mysql.sock");
  //Проверяем адрес электронной почты продавца, цену и txn id
  //Нет необходимости изменять синтаксис, чтобы передать объект по ссылке
  $valid_txnid = check_txnid($con, $txn_id);
  $valid_price = check_price($con, $price, $inventoryid);
  $valid_email = check_email($receiver_email);
  //Если все проверили, то пишем код
  if($valid_price && $valid_email && $valid_txnid)
  {
    //Обновляем базу данных с txn id
    $strsql = "UPDATE tblorders SET txnid = '$txn_id' ".
    "WHERE orderid = $orderid";
    $con->query($strsql);
    $message ="Successful, transaction id: $txn_id\n";
  }
  else
  {
    //Неудачная транзакция
    $message ="Unsuccessful, transaction id: $txn_id\n";
  }
}
else if($status == "INVALID")
{
  //Сообщение о подозрительной транзакции
  $message ="Suspicious IPN with transaction id: $txn_id";
}
else
{
  //Обрабатываем другие типы
  $message ="Incomplete purchase with transaction id: $txn_id";
}
mail ("notify@myisp.com", "PayPal", $message);
```

Запуск трюка

Для начала вам понадобится учетная запись в системе PayPal. Откройте домашнюю страницу PayPal и создайте себе торговую учетную запись. Затем вам понадобится изменить предоставленный в качестве примера сценарий и страницы, чтобы они могли работать с вашими приложениями. На странице `buynow.html`, конечно же, будет отражаться товар, который вы продаете. Вам также необходимо

будет изменить адрес электронной почты в файлах buynow.html и verifypurchase.php. Замените seller@myisp.com адресом, который вы указали при регистрации учетной записи в системе PayPal. Это очень важно, так как именно из него будет проводиться идентификация учетной записи, на которую будет поступать оплата. Измените также адрес **notify@myisp.com** — он будет предназначаться для подтверждения факта оплаты.

ПРИМЕЧАНИЕ

Возможно, вам и не понадобится получать подтверждение об оплате, или вы захотите заменить это место в программе кодом, который будет вести файл журнала, что будет особенно полезно в случае неудавшейся оплаты.

Измените сценарий connection.php так, чтобы он отображал переменные в соответствии с вашим MySQL-сервером. В сценарий presubmit.php не требуется вносить никаких изменений, если только вы не изменили структуру базы данных. Не сомневаюсь, что вы создадите свою собственную базу данных для ваших деловых нужд, но если вы хотите проверить работоспособность этого кода в таком виде, в каком он находится сейчас, то предлагаю команды на SQL, которые создадут для вас минимально необходимую для работы базу данных:

```
CREATE TABLE tblbooks (
  inventorynumber int(11) NOT NULL auto_increment,
  title varchar(150) NOT NULL default '',
  author varchar(100) NOT NULL default '',
  cost float(6,2) NOT NULL default '0.00',
  listprice float(7,2) NOT NULL default '0.00',
  publicationdate varchar(4) default NULL,
  publisher varchar(4) NOT NULL default '',
  PRIMARY KEY (inventorynumber),
  KEY authidx (author),
  KEY titleidx (title),
) ENGINE=MyISAM DEFAULT CHARSET=latin1

CREATE TABLE tblorders (
  orderid int(11) NOT NULL auto_increment,
  customerid int(11) default NULL,
  orderdate date default NULL,
  txnid varchar(17) default NULL,
  PRIMARY KEY (orderid)
) ENGINE=MyISAM DEFAULT CHARSET=latin1

CREATE TABLE tblorderitems (
  orderid int(11) NOT NULL default '0',
  inventorynumber int(11) NOT NULL default '0',
  PRIMARY KEY (orderid,inventorynumber)
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

Далее загрузите файлы на ваш сервер и удостоверьтесь, что файлы connection.php, buynow.html и presubmit.php расположены в одной и той же директории. В эту же директорию вы можете поместить и сценарий verifypurchase.php, но, возможно, было бы неплохо создать для него свою директорию. Если вы так и поступите, то удостоверьтесь, что изменили пути к подключаемым файлам в файле

connection.php. Перейдите к вашей учетной записи в системе PayPal, включите поддержку IPN и введите полный URL-адрес к сценарию verifypurchase.php. Чтобы сделать покупку, откройте в вашем браузере страницу buynow. php. Вы можете быть уверены, что все работает, если при нажатии кнопки Buy Now вы перейдете к сайту PayPal и после оплаты получите письмо по электронной почте, содержащее ID транзакции.

Улучшаем трюк

Нельзя предусмотреть все сразу. В нашем конкретном случае цена определенного приобретаемого товара совпадает со стоимостью всей покупки. Однако очень часто нужно учитывать и стоимость доставки, а также возможность оплаты, используя другую валюту. Вы можете легко внести эти дополнения, изменив функцию check_price().

Использование mysqli необязательно, но предоставляемая этим модулем возможность работать с готовыми операторами может быть очень полезна, особенно если вы решите создать возможность использовать корзину для нескольких покупок, а не ограничиваться возможностью сделать только одну покупку.

Кроме того, имеет смысл зарегистрироваться в системе PayPal в качестве разработчика, если вы частенько создаете приложения, использующие PayPal. В результате вы сможете воспользоваться изолированной программной средой PayPal для тестирования приложений, прежде чем запускать их в эксплуатацию. Особенно это важно при использовании приложений IPN, где взаимодействие происходит только между серверами и от пользователя не требуется вводить никакие данные. Отладка автоматизированного ответа от сервера также вызывает некоторые сложности, так как у вас нет под руками браузера, чтобы отобразить ошибку. Один из вариантов — отправлять значение переменной, содержащей данные для переотправки их системе PayPal по электронной почте. Сделав так, вы увидите в сообщении примерно следующее:

```
mc_gross=24.95&address_status=confirmed&payer_id=TYWM55XFZCN8S
&tax=0.00&address_street=36+Main+Street
&payment_date=16%3A00%3A32+Aug+11%2C+2005+PDT
&payment_status=Completed&charset=windows-1252&address_zip=12345
&first_name=Peter&mc_fee=0.82&address_country_code=US
&address_name=Peter+Buyer&notify_version=1.9&custom=20
&payer_status=unverified&business=seller%40isp.com
&address_country=United+States&address_city=Toledo&quantity=1
&verify_sign=ACUe-E7Hjxmeel8FjYAtjnx-yjHAAVhtx75Yq6UdimmRaeJhnewrOugZ
&payer_email=buyer%40myisp.com&txn_id=1E044782YK461110T
&payment_type=instant&last_name=Buyer&address_state=OH
&recei ver_email=seller%40isp.com&payment_fee=0.82
&receiver_id=JEFVKNSSDLTBL&txn_type=web_accept
&item_name=Object+Oriented+PHP&mc_currency=USD
&item_number=06734980548&test_ipn=1&payment_gross=24.95
&shipping=0.00&cmd=_notify-validate
```

Реализовав это, вы сможете получать подтверждения об отправке всех пар переменных имя/значение.

Питер Лэвин (Peter Lavin)

Смотрите также

- «Создание корзины» (см. трюк 66).

ТРЮК №63

Выясните, откуда пришли ваши посетители

Используйте модуль PEAR Net-Geo, чтобы выяснить, в какой точке мира находятся пользователи вашего сайта.

Вы когда-нибудь интересовались, где живут пользователи, посещающие ваш сайт? Оказывается, выяснить это не так уж и сложно. Во-первых, после обработки запроса в Apache из него вам передается IP-адрес этого запроса. Во-вторых, при помощи простой доступной библиотеки PEAR можно сопоставить этот IP с физическим месторасположением. В этом трюке будет рассказано, как это все работает.

Код

Сохраните исходный код из примера 6.41 в файл с именем geo.php.

Пример 6.41. Преобразователь IP-адреса в географические координаты

```
<html>
<body>
<?php
require_once( "cache/lite.php" ):
require_once( "net/geo.php" );
Sip = $_SERVER['REMOTE_ADDR'];
Sip = "64.246.30.37";
Sgeo - new Net_Geo( );
?>
<table>
<tr><td>IP</td><td><?php echo( Sip ); ?></td></tr>
<tr><td>City</td><td><?php echo( $res['CITY'] ); ?></td></tr>
<tr><td>State</td><td><?php echo( $res['STATE'] ): ?></td></tr>
<tr><td>Country</td><td><?php echo( $res['COUNTRY'] ); ?></td></tr>
<tr><td>Latitude</td><td><?php echo( $res['LAT'] ); ?></td></tr>
<tr><td>Longitude</td><td><?php echo( $res['LONG'] ); ?></td></tr>
</table>
</body>
</html>
```

Как вы видите, все очень просто. Необходимы только определенные модули PHP и PEAR, а все, что вам надо, сделает функция Net_Geo(). Из объекта, который эта функция вернет вам в результате работы, вы можете просто, указав соответствующие параметры, извлечь город, штат, страну, широту и долготу.

Запуск трюка

Для начала вам понадобится модуль PEAR Net-Geo (см. трюк 2). Установите его и откройте в браузере файл geo.php (рис. 6.33). Согласитесь, круто? Теперь я знаю

город, штат и страну, откуда пришел запрос, а также широту и долготу. Имея такую информацию, я могу нанести все запросы на карту и выяснить, откуда они приходят с географической точки зрения.

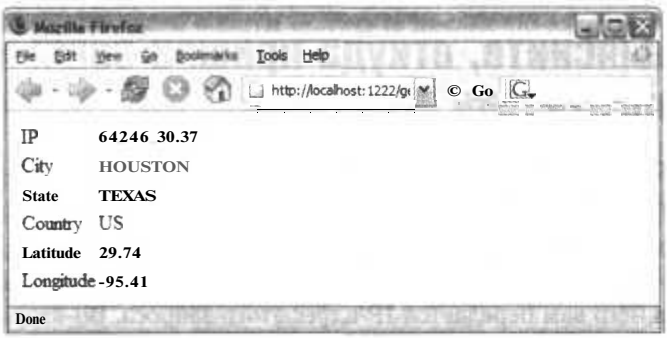


Рис. 6.33. Географическое местоположение запроса

Смотрите также

- «Создание простых карт Google» (см. трюк 95).

Импорт данных из vCard

Научите ваше приложение на PHP считывать данные в формате vCard, предназначенные для хранения контактной информации.

Очень часто мы просим пользователей наших интернет-приложений записать данные в файл, который у них уже существует, в то время как приложения с гибкими настройками само могло бы напрямую считать эти данные из (уже существующего) файла пользователя. Примером этому может послужить файл vCard — универсальный стандарт, предназначенный для хранения контактной информации, в частности адреса электронной почты и физического адреса. Каждая нормальная почтовая программа либо программа для работы с адресными книгами может импортировать и экспортировать файлы в формате vCard (VCF). В этом трюке будет показано, как считывать данные из файлов данного формата, используя модуль PEAR `Contact_Vcard_Parse` (см. трюк 2).

Код

Сохраните исходный код из примера 6.42 в файл с именем `index.php`.

Пример 6.42. Простая страница для загрузки файла

```
<html>
<body>
<form enctype="multipart/form-data" action="read.php" method="post">
```

```
VCF file: <input type="hidden" name="MAX_FILE_SIZE" value="2000000" />
  <input type="file" name="file" /><br/>
  <input type="submit" value="Upload" />
</form>
</body>
</html>
```

Сохраните исходный код из примера 6.43 в файл с именем read.php.

Пример 6.43. Код, предназначенный для чтения данных в формате vCard

```
<html>
<body>
<?php
require_once( 'Contact_Vcard_Parse.php' );
if ( $_FILES['file']['tmp_name'] )
{
  $parse = new Contact_Vcard_Parse( );
  $cardinfo = $parse->fromFile( $_FILES['file']['tmp_name'] );
  foreach( $cardinfo as $card )
  {
    $first = $card['N'][0]['value'][0][0];
    $last = $card['N'][0]['value'][1][0];
    $email = $card['EMAIL'][0]['value'][0][0];
    ?>
    <a href="mailto:<?php echo( $email ); ?>">
    <?php echo( $first ); ?> <?php echo( $last ); ?>
    </a><br/>
    <?php
  }
}
?>
</body>
</html>
```

Вся основная работа в этом трюке происходит в сценарии read.php: он считывает временно загруженный файл, используя метод fromFile() объекта типа Contact_Vcard_Parse. Затем сценарий заносит в память каждый адрес электронной почты, который будет найден при помощи стандартной техники поиска в PHP с использованием текстовых шаблонов.

Запуск трюка

Создайте файл vCard путем экспорта контактов из вашей программы для работы с электронной почтой или из адресной книги. Затем загрузите код на сервер и откройте в браузере страницу index.php (рис. 6.34).

Откройте VCF-файл, нажав кнопку Browse. Затем нажмите кнопку Upload, чтобы передать файл на сервер (рис. 6.35).

На этой странице отображаются только имя и фамилия из vCard, размещенные в скобках в якорном теге в виде ссылки на адрес электронной почты пользователя. В vCards может содержаться несколько записей, и если это так, то пользователь может видеть их все в виде ссылок для каждой из них.

Рис. 6.34. Страница, принимающая для импорта файлы vCard

Рис. 6.35. Простая страница для импорта, отображающая имя из списка с адресами электронной почты

Смотрите также

- «Формирование файлов в формате vCard, используя данные из вашего приложения» (см. трюк 65).

ТРЮК №65

Формирование файлов в формате vCard с использованием данных из вашего приложения

Используйте простой шаблон vCard для динамического создания файлов типа vCards с данными из ваших приложений.

Если на вашем сайте в Интернете есть база данных с контактами, было бы неплохо создать для каждого из них ссылку на адрес электронной почты из этого контакта, но согласитесь, что было бы удобно, чтобы вся информация такого рода хранилась в одном удобном формате? Отлично, для решения этой задачи идеально подходит формат vCard (VCF). К тому же вы можете создавать vCards автоматически, используя для этого PHP (как и в трюке, в котором мы считывали данные из файлов в формате VCF (см. трюк 64)).

Код

Сохраните исходный код из примера 6.44 в файл с именем vcard.php.

Пример 6.44. Код для создания файлов в формате vCard

```
<?php
header( "Content-type:text/x-vCard" );
$first = "Howard";
$last - "Dean";
$email = "dean@dnc.org";
?>
BEGIN:VCARD
VERSION:2.1
N:<?php echo($last); ?>;<?php echo($first); ?>
FN:<?php echo($first); ?> <?php echo($last); ?>
EMAIL:PREF; INTERNET:<?php echo($email); ?>
REV:20050626T024452Z
END:VCARD
```

Запуск трюка

Загрузите файл с исходным кодом на сервер и откройте его в браузере. Вы должны увидеть окно, предназначенное для загрузки файлов с сервера (рис. 6.36).

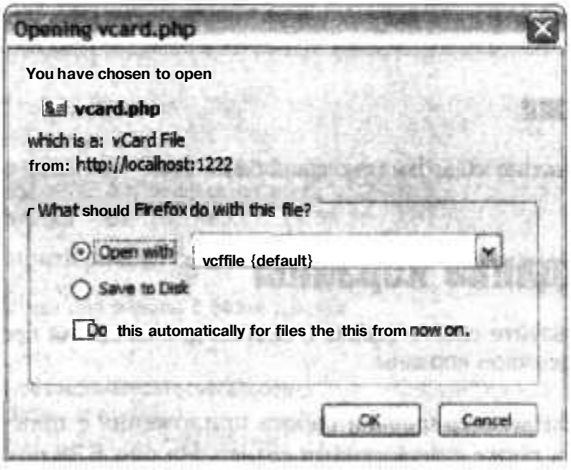


Рис. 6.36. Окно для загрузки файлов, появляющееся при открытии файла vcard.php

По умолчанию (в Windows) запустится Outlook Express и будет создан новый файл с контактом, что очень удобно. Вы можете добавить этого человека в ваш список контактов, выбрав пункт меню Save and Close (Сохранить и закрыть), как это показано на рис. 6.37. Весь секрет заключается в заголовке Content-type, который и сообщает при загрузке файла пользователем, что это файл в формате vCard, а не просто текст. В этом случае операционная система запускает приложение, которое по умолчанию настроено на обработку данных в формате vCards.

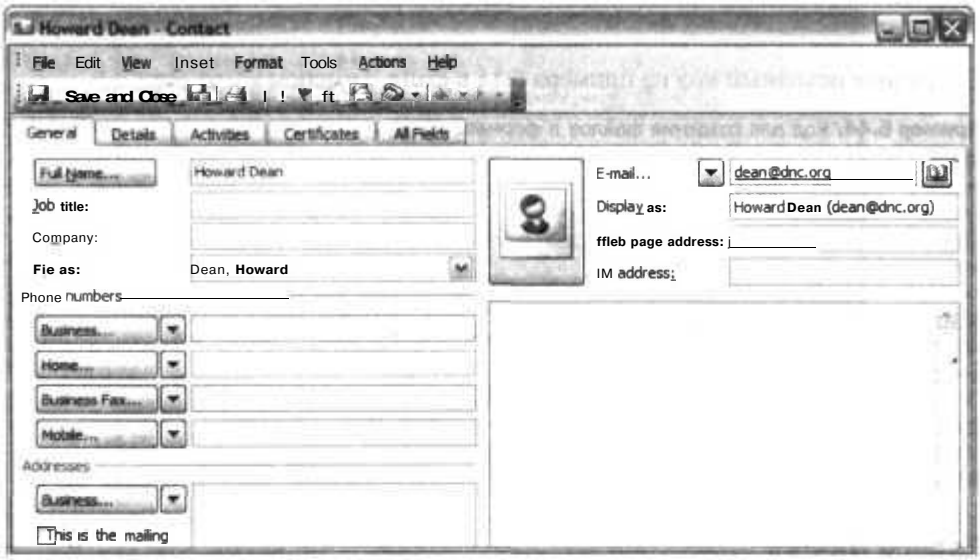


Рис. 6.37. Импорт данных в Outlook в формате vCard

ПРИМЕЧАНИЕ

Маловероятно, но может случиться так, что у вас не установлено приложение, которое может работать с данными vCards, и вы сможете только сохранить где-нибудь на диске предлагаемый вам файл.

Смотрите также

- «Импорт данных из vCards» (см. трюк 64).

ТРЮК №66

Создание корзины

Используйте cookie-файлы и сеансы для создания простых приложений с поддержкой корзины.

В этом трюке продемонстрирована работа приложения с простой корзиной, созданной на PHP, а также с переменами сеанса. На рис. 6.38 показано взаимодействие страниц в приложении, использующем корзину. Для начала пользователь открывает страницу `index.php`, откуда он может всегда перейти к странице `checkout.php` (которая покажет содержимое его корзины). На странице `index.php` он может добавить покупки в корзину, нажав кнопку Add, которая передает информацию об этом на страницу `add.php` и вновь возвращает пользователя на страницу `index.php`. На странице `checkout.php` пользователь может удалить товар, причем будет использоваться схожая с добавлением покупки в корзину логика работы, но для удаления нам понадобится страница `delete.php`, которая также вернет пользователя на страницу `checkout.php`.

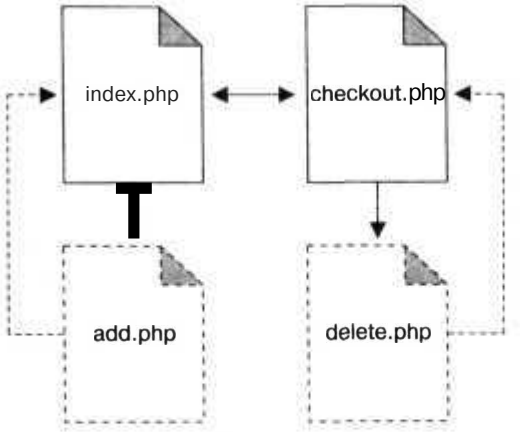


Рис. 6.38. Взаимодействие между страницами при использовании корзины

Код

Сохраните исходный код из примера 6.45 в файл с именем shopcart.sql.

Пример 6.45. Схема базы данных для корзины

```
DROP TABLE IF EXISTS product;
CREATE TABLE product (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  name TEXT,
  price FLOAT,
  PRIMARY KEY( id )
);

INSERT INTO product VALUES ( 0, "Code Generation in Action", 49.99 );
INSERT INTO product VALUES ( 0, "Podcasting Hacks", 29.99 );
INSERT INTO product VALUES ( 0, "PHP Hacks", 29.99 );
```

Исходный код из примера 6.46 сохраните в файл с именем dblib.php.

Пример 6.46. Библиотека для работы с базой данных

```
<?php
require_once( "DB.php" );
$dsn = 'mysql://root:password@localhost/shopcart';
$db =& DB::Connect( $dsn, array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )); }
function get_db( ) { global $db; return $db; }
function get_products( )
{
  global $db;
        $res = $db->query( "SELECT * FROM product", array( ) );
        $out = array( );
  if ( $res != null )
    while( $res->fetchInto( $row, DB_FETCHMODE_ASSOC ) ) { $out []= $row; }
    return $out;
}

function product_info( $id )
```

```
{
  global $db;
        $res = $db->query( "SELECT * FROM product WHERE id-?",
  array( Sid ) );
  if C $res !- null )
  {
    $res->fetchInto( $row, DB_FETCHMODE_ASSOC );
    return $row;
  }
  return null;
}
?>
```

Сохраните исходный код из примера 6.47 в файл с именем index.php.

Пример 6.47. Страница с товарами

```
<?php
require_once( "dblib.php" ):
session_start( ):
$products - get_products( ):
?>
<html>
<head>
<title>My Products</title>
</head>
<style type-"text/css">
h1 { border-bottom: 1px solid black; font-size: medium; margin-bottom: 10px; }
</style>
<script>
function buy( prod_id )
{
  document.getElementById( 'prod_id' ).value = prod_id;
  document.getElementById( 'buyform' ).submit( );
  return null;
}
</script>
<body>
<form id="buyform" action="add.php" method-"post">
  <input type="hidden" name-"prod_id" id="prod_id" value-"" />
</form>
<table width="600" border-"0" cellspacing="0" cellpadding="5">
<tr>
<td width="70%" valign="top">
<h1>Products</h1>
<table width="100%">
<?php foreach( $products as $product ) { ?>
<tr>
<td width="70%"><?php echo( $product['name'] ); ?></td>
<td width="15%" align="right">$<?php echo( $product['price'] ); ?></td>
<td width="15%" align="center"><a href="javascript:buy( <?php echo(
$product['id'] ): ?> );">buy</a>
</tr>
<?php } ?>
</table>
</td>
```

```
<td width="30%" valign-"top">
<h1>Shopping cart</h1>
<?php
if( isset( $_SESSION['cart'] ) ) {
?>
<!-- CART : <?php echo( join( ".", array_keys( $_SESSION['cart'] ) ) ); ?> -->
<table width="100%" cellspacing-"!)" cellpadding="5">
<?php
foreach( array_keys( $_SESSION['cart'] ) as $product ) {
  $info - product_info( $product ):
  ?>
  <tr><td>
  <?php echo( $info['name' ] ); ?>
  </td></tr>
<?php } ?>
<tr><td align="center">
<a href="checkout.php">Checkout</a>
</td></tr>
</table>
<?php
}
?>
</td>
</tr>
</table>
</body>
</html>
```

Сохраните исходный код из примера 6.48 в файл с именем add.php.

Пример 6.48. Сценарий, добавляющий товар в корзину

```
<?php
session_start( );
if ( !isset( $_SESSION['cart'] ) )
$_SESSION['cart'] - array( ):
$_SESSION['cart'][$_POST['prod_id']] - 1:
header( "location: index.php" ):
?>
```

Исходный код из примера 6.49 сохраните в файл с именем checkout.php.

Пример 6.49. Страница, предназначенная для проверки

```
<?pnp
require_once( "dblib.php" ):
session_start( ):
?>
<html>
<head>
<title>Your Cart</title>
<script language="Javascript">
function submit_delete( )
{
  document.getElementById( "delform" ).submit( ):
  return null;
}
```

```
</script>
<style type="text/css">
h1 { border-bottom: 1px solid black; font-size: medium; margin-bottom: 10px; }
</style>
</head>
<body>
<?php
if ( isset( $_SESSION['cart'] ) && count( $_SESSION['cart'] ) > 0 ) {
?>
<!-- CART : <?php echo( join( ".", array_keys( $_SESSION['cart'] ) ) ): ?> -->
<form id="delform" action="delete.php" method="post">
<table width="600">
<tr>
<th width="3%"></th>
<th width="77%">Product</th>
<th width="20%">Price</th>
</tr>
<?php
foreach( array_keys( $_SESSION['cart'] ) as $product )
{
$prod - product_info( $product ):
?>
<tr>
<td><input type="checkbox" name="ids[]" value="<?php echo( $product ): ?>" />
</td>
<td><?php echo( $prod['name'] ); ?></td>
<td align="right"><?php echo( $prod['price'] ): ?></td>
</tr>
<?php } ?>
<tr>
<td></td>
<td align="center"><a href="javascript:submit_delete( )">Delete checked items</a>
       
<a href="index.php">Return to store</a>
       
<a href="buy.php">Buy these items</a>
</tr>
</table>
</form>
<?php } else { ?>
You should go and <a href="index.php">buy some stuff</a>.
<?php } ?>
</body>
</html>
```

Сохраните исходный код из примера 6.50 в файл с именем `delete.php`.

Пример 6.50. Сценарий, удаляющий товар из корзины

```
<?php
session_start( ):
foreach( $_POST['ids'] as $did )
unset( $_SESSION['cart'][$did] );
header( "location: checkout.php" );
?>
```

Работа с корзиной начинается со страницы index.php, которая показывает расположенные в базе данных товары. Щелкнув кнопкой мыши на любом из них, вы передадите данные странице add.php, которая начнет сеанс и добавит товар в корзину. Затем сценарий add.php возвращает пользователя назад к странице с товарами, на которой также будет показано содержимое корзины. Когда в корзину добавлен хотя бы один товар, пользователю становится доступна ссылка на страницу checkout.php. На этой странице можно видеть содержимое корзины и, кроме того, можно удалить из нее товары при помощи определенной ссылки. Эта ссылка указывает на сценарий delete.php, который удаляет товары из корзины и переводит пользователя назад к странице checkout.php.

Запуск трюка

Для запуска этого трюка требуется сперва создать базу данных и загрузить в нее схему:

```
% mysqladmin --user=root --password=password createshopcart
% mysql --user=root --password=password shopcart < shopcart.sql
```

Теперь загрузите PHP-сценарии на сервер и откройте страницу index.php. Вы увидите домашнюю страницу магазина, как это показано на рис. 6.39. Слева на странице расположен список товаров со ссылками. Пользователь может купить эти товары, щелкнув кнопкой мыши на определенных ссылках. При щелчке кнопкой мыши на ссылке в сценарий add.php передается ID продукта, и он добавляет товар в корзину, которая хранится в пользовательской переменной сеанса.

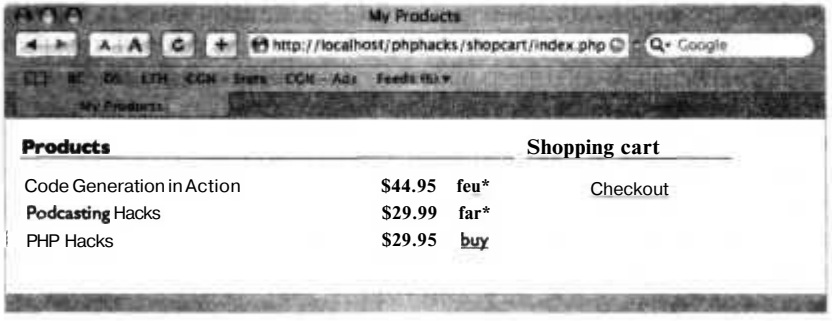


Рис. 6.39. Домашняя страница магазина index.php

Сценарий add.php отправляет пользователя назад к странице index.php, на которой справа показано обновленное содержимое корзины (рис. 6.40).

После того как вы щелкнули кнопкой мыши на ссылке buy для всех книг, необходимо произвести проверку. Щелкните кнопкой мыши на ссылке Checkout, и вы перейдете к странице checkout.php, как это показано на рис. 6.41. В данном случае мы видим, что в корзину были добавлены три книги.

Теперь пользователь может выбрать любой товар и удалить его из корзины или продолжить делать покупки через корзину. Для удаления книг из корзины необходимо

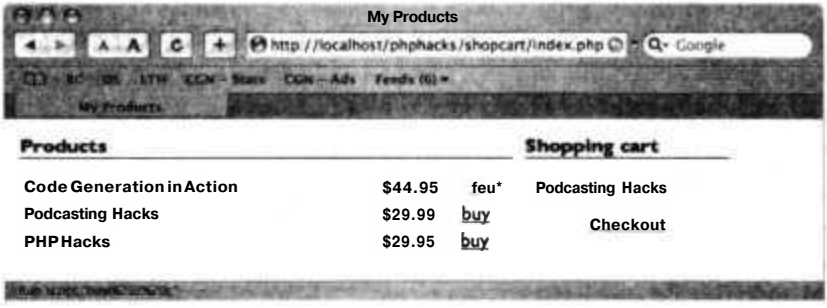


Рис. 6.40. Внешний вид страницы после щелчка кнопкой мыши на ссылке buy возле товара Podcasting Hacks

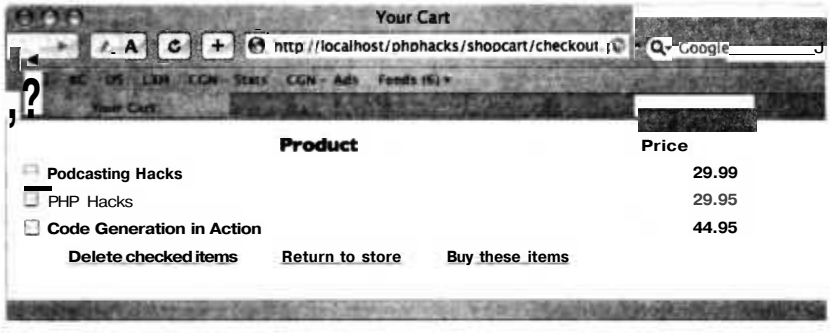


Рис. 6.41. Страница, предназначенная для проверки

сперва выбрать их, как это показано на рис. 6.42. Далее пользователь должен щелкнуть кнопкой мыши на ссылке Delete checked items, чтобы удалить товары из корзины. При этом сценарий delete.php просто удаляет выбранные значения из массива, используемого корзиной. Затем сценарий отправляет пользователя назад к странице, на которой проводилась проверка, как это показано на рис. 6.43. Чтобы завершить работу с корзиной, вам необходимо подтвердить покупку товаров, щелкнув кнопкой мыши на ссылке Buy these items. В трюке «Добавляем кнопку Buy Now» (см. трюк 62) более подробно рассказывается о реализации возможностей делать покупки через Интернет.

Рис. 6.42. Выбираем несколько книг, чтобы удалить их из корзины

Рис. 6.43. Книги были удалены из корзины

Смотрите также

J «Добавляем кнопку Buy Now» (см. трюк 62).

ГЛАВА 7

Шаблоны

Трюки 67-78

В 1994 году Эрих Гамма, Ричард Хелм, Ральф Джонсон и Джон Влиссидес издали книгу «Дизайнерские шаблоны» (Эдисон Уэсли). Книга в кратчайшие сроки стала классикой программирования из-за очень полезного материала, размещенного в ней, причем она также положила начало новому метаязыку, распространенному в технических и архитектурных сообществах. Основополагающий принцип, заложенный в книге, подразумевающий создание комплекта шаблонов для структуры объекта, а также структуры его внутренних объектов, был заимствован из строительной архитектуры и плавно адаптирован для программирования.

В книге были размещены 40 шаблонов, отобранных с опорой на накопленный за многие годы опыт. Каждый шаблон был представлен в не зависимой от языка форме, и большинство из них можно было спокойно использовать в любой среде разработки.

ПРИМЕЧАНИЕ

Некоторые из шаблонов, такие как шаблон Итератор (Iterator), были специально разработаны для того, чтобы компенсировать определенные недостатки языков программирования (в случае с шаблоном Итератор это был язык C++). PHP использует шаблон Итератор наследственно, при помощи метода foreach.

Как известно, PHP не поддерживал шаблоны. Однако все изменилось с появлением PHP 5, который стал серьезно использоваться в качестве языка и среды для разработки корпоративных приложений. Теперь, получив в свое распоряжение мощную объектную модель, прочные IDE, а также новые обширные возможности для разработчиков, корпоративное сообщество обратило внимание на PHP.

В последнее время вышло несколько книг по PHP, в которых было рассказано об использовании в этом языке программирования шаблонов, и, я думаю, было бы неплохо рассказать о шаблонах и здесь. Я решил в этой книге ссылаться на оригинальный вариант книги «Дизайнерские шаблоны». Я отобрал ряд шаблонов из нее, чтобы рассмотреть их реализацию. Как сами шаблоны, так и их исполнение являются цельной строительной базой, а также источником вдохновения при написании вашего собственного кода.

ТРЮК №67

Отслеживание ваших объектов

Используйте шаблон Наблюдатель (Observer) для слабой связи ваших объектов.

Слабая связь очень важна во всех крупномасштабных проектах, но мало кто толком понимает, что значит этот термин. Вы когда-нибудь сталкивались с такой ситуацией, когда вы вносите минимальные изменения в ваш проект и, как следствие, оказывается, что вам придется изменить его практически целиком? Такая ситуация очень часто возникает из-за сильной связи между модулями в приложении. Каждый модуль зависит от конкретного состояния или функции из других модулей. Когда возникает ошибка в одном из них, все остальные также выходят из строя. Когда вносятся изменения в один из этих модулей, необходимо также изменить и все другие.

Шаблон Наблюдатель служит для ослабления связей между объектами, упрощая модель взаимодействия между ними. Объект сам позволяет отслеживать его, предоставляя для этого механизм, в котором объекты могут регистрироваться. Когда в отслеживаемый объект вносятся какие-то изменения, он, используя объект для уведомлений, сообщает об этом отслеживающим его объектам. Для отслеживаемого объекта не важно, как или почему за ним наблюдают, знает ли он тип объекта, который это делает. Кроме того, наблюдатели обычно не интересуются, почему или как в объект были внесены изменения. Все, что они делают, — это отслеживают их. Классическим примером может послужить код, в котором диалог следит за состоянием поля с флажком. Этому полю не важно, наблюдает за ним один объект или тысяча. Оно просто отсылает сообщение, когда его состояние изменяется. Диалог также абсолютно не интересуется тем, как было реализовано это поле с флажком, он просто наблюдает за состоянием этого поля и отслеживает сообщения о произошедших в нем изменениях.

В этом трюке я продемонстрирую вам работу с шаблоном Наблюдатель, создав отслеживаемый список посетителей. В этом объекте отображается содержимое таблицы клиентов из базы данных. Объект `CustomerList` отправит уведомление, когда будет добавлен новый клиент. Для реализации возможности такого рода слежки он будет использовать объект `SubscriptionList`. Объекты же, выступающие в роли наблюдателей, будут использовать экземпляры класса Subscri pti onLi st, предназначенного для того, чтобы другие объекты могли зарегистрировать себя в `CustomerList`. Эти наблюдатели воспользуются методом add(), чтобы добавить самих себя в список, а `CustomerList` будет использовать метод `invoke()` для отправки сообщений наблюдателям. Неважно, есть у нас тысяча отслеживающих объектов или нет ни одного. Вся прелесть в том, что объекты, выступающие в роли наблюдателей, не зависят от CustomerList и не взаимодействуют с ним напрямую, они изолированы от клиентов при помощи класса Subscri pti onList. В этом примере у нас будет один наблюдатель — объект Log, который выводит любые полученные от `CustomerList` сообщения на экран. Взаимодействие между объектами показано на рис. 7.1.

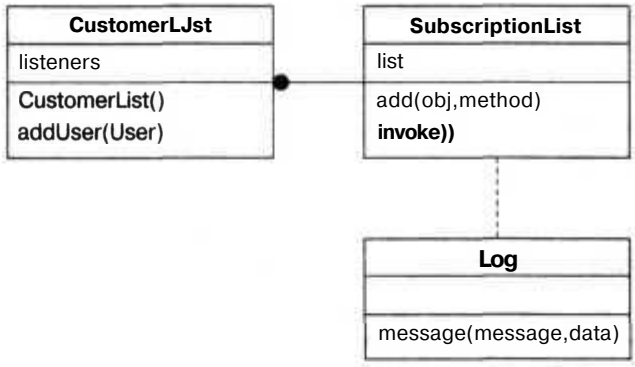


Рис. 7.1. Объекты CustomerList, SubscriptionList и присоединенный к ним Log

Код

Сохраните исходный код из примера 7.1 в файл с именем observer.php.

Пример 7.1. Пример работы с шаблоном Наблюдатель

```
<?php
class Log
{
  public function message) $sender. $messageType, $data )
  {
    print $messageType." - ".$data."\n";
  }
}

class SubscriptionList
{
  var $list = array) );
  public function add( $obj, $method )
  {
    $this->list []= array) $obj. $method ):
  }
  public function invoke) )
  {
    $args - func_get_args( );
    foreach( $this->list as $1 ) { call_user_func_array( $1, $args ); }
  }
}

class CustomerList
{
   public $listeners:
  public function CustomerList) )
  {
    $this->listeners = new SubscriptionList) );
  }
  public function addUser( $user )
  {
    $this->listeners->invoke( $this. "add", "$user" );
```

```
    }
}

$1 = new Log( );
$cl = new CustomerList( );
$cl->listeners->add( $1, 'message' );
$cl->addUser( "starbuck" );
?>
```

Запуск трюка

Запустите этот исходный код из командной строки:

```
% php observer.php
add - starbuck
```

Первым делом код создает журнал и список клиентов. Затем журнал прописывается в этом списке при помощи метода `add()`. И наконец, надо добавить пользователя в список клиентов. При добавлении наблюдателям будет отправлено сообщение, в нашем случае — журналу, который выдаст на экран сигнал о том, что был добавлен новый клиент. Можно легко расширить возможности этого кода, чтобы он производил, например, регистрацию этого пользователя или отправлял сообщение по электронной почте о новом клиенте — и все это без внесения изменений в код `CustomerList`. Это и есть слабая связь, и теперь становится понятно, почему так важен шаблон Наблюдатель. Количество вариантов его использования просто огромно.

Системы, работающие с окнами, используют шаблоны Наблюдатель, называя их *событиями*. У таких компаний, как Tibco, вся их бизнес-модель построена на базе шаблона Наблюдатель, объединяющего такие огромные бизнес-системы, как Human Resources и Payroll. Системы работы с базами данных используют шаблон Наблюдатель и отправляют на выполнение код, который следит за событиями *триггеров*. Они активируются, когда в записи определенного типа из базы данных вносятся какие-то изменения. Использование шаблона Наблюдатель также бывает очень полезно во всех тех случаях, когда вы думаете, что в дальнейшем может быть использовано изменение состояния, но вы еще не определились, для кого это будет иметь значение. Вы можете реализовать отслеживающие объекты позже и не привязывать их к объектам, за которыми будет производиться наблюдение.

Одним из фокусов, который может получиться при помощи шаблона Наблюдатель, является возникновение бесконечного цикла. Такая ситуация может возникнуть, когда элементы, отслеживающие систему, могут также вносить в нее изменения. Например, раскрывающийся список позволяет изменить выбранное значение и сообщает структуре данных об этом. Эта же структура данных затем сообщает полю о том, что было изменено значение, после чего оно также изменяет значение, чтобы то соответствовало вновь полученному, и все это для того, чтобы вновь отправить сообщение в структуру данных о том, что произошли изменения и т. д. Самый простой способ решить эту проблему — правильно реализовать работу поля с раскрывающимся списком, чтобы избегать подобной рекурсии. Оно

просто должно игнорировать сообщение от структуры данных, если в данный момент оно само сообщает этой структуре об изменении значения.

Смотрите также

- «Создание очереди сообщений» (см. трюк 50).
- «Преобразование любого объекта в массив» (см. трюк 53).

ТРЮК №68

Создание объектов при помощи Абстрактной фабрики

Используйте шаблон Абстрактная фабрика (Abstract Factory), чтобы контролировать тип создаваемого объекта.

Абстрактная фабрика — это дизайнерский шаблон, выступающий в роли распределителя. Вы просто говорите ему, что вы хотите, и он возвращает вам объект исходя из указанных критериев. Самое главное, что вы можете выбрать типы создаваемых объектов, просто изменив тип фабрики. В этом примере простейшая фабрика создает объект Record для записей, каждая из которых будет иметь ID, имя и фамилию. Взаимодействие между всеми классами показано на рис. 7.2.

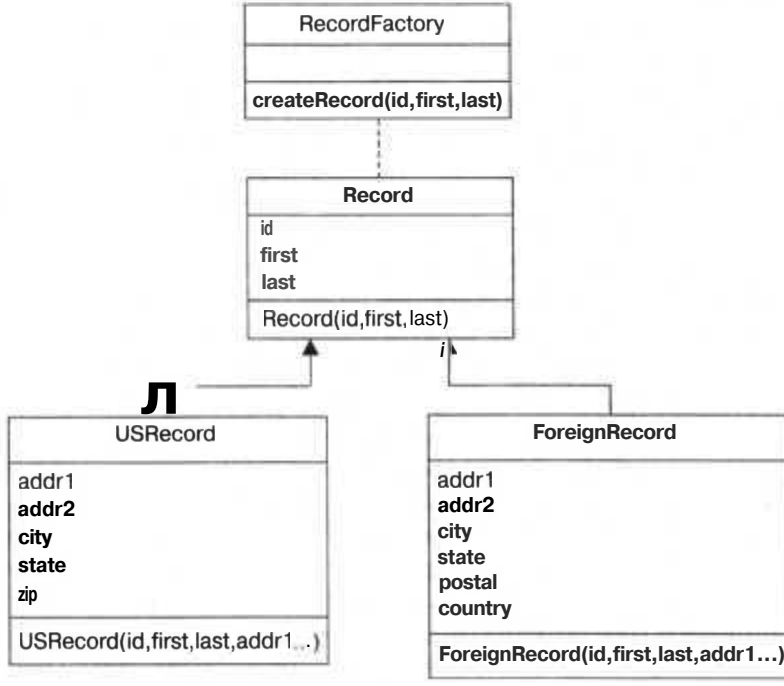


Рис. 7.2. Классы Record и RecordFactory

ПРИМЕЧАНИЕ

Фабричные объекты обычно создают несколько типов объектов. Но, чтобы этот пример получился попроще, я ограничил фабрику, и она создает объект только одного типа.

В PHP нет способа строго установить, что объекты какого-то определенного типа могут создаваться только фабрикой. Но если вы будете использовать ее как можно чаще, разработчики, постоянно копируя и вставляя ваш код, в итоге перейдут к использованию фабрики, и очень скоро она де-факто станет средством для создания объектов различных типов.

Код

Сохраните исходный код из примера 7.2 в файл с именем `abs_factory.php`.

Пример 7.2. Пример работы с шаблоном Абстрактная фабрика

```
<?php
class Record
{
  public $id = null;
  public $first - null;
  public $last - null;
  public function__construct( $id. $first. $last )
  {
    $this->id = $id:
    $this->first = $first:
    $this->last = $last;
  }
}

class USRecord extends Record
{
  public $addr1 = null;
  public $addr2 = null;
  public $city = null:
  public $state - null:
  public $zip = null;
  public function__construct( $id. $first. $last,
    $addr1. $addr2, $city. $state. $zip )
  {
    parent::__construct( $id. $first. $last ):
    $this->addr1 - $addr1:
    $this->addr2 - $addr2:
    $this->city = $city:
    $this->state = $state:
    $this->zip = $zip;
  }
}

class ForeignRecord extends Record
{
  public $addr1 = null;
  public $addr2 = null;
  public $city = null;
  public $state = null;
```

```
  public $postal - null;
  public Scountry - null;
  public function___construct( Sid, $first. Slast,
    Saddrl. $addr2. Scity. Sstate. Spostal. $country )
  {
    parent::_ _construct( lid. Sfirst, $last ):
    $this->addrl - $addrl:
    $this->addr2 - $addr2:
    $this->city - $city;
    $this->state - Sstate;
    $this->postal - Spostal;
    $this->country = Scountry;
  }
}

class RecordFactory
{
  public static function createRecord( $id. $first, Slast.
  $addrl. $addr2. Scity. Sstate. $postal, $country )
  {
    if ( strlen( Scountry ) > 0 && Scountry !- "USA" )
      return new ForeignRecord( Sid. $first, $last.
      $addrl, Saddr2. $city, Sstate. Spostal. Scountry );
    else
      return new USRecord( Sid. Sfirst. Slast.
      Saddrl. Saddr2. Scity. Sstate. Spostal ):
  }
}

function readRecords( )
{
  Srecords - array( );
  Srecords []- RecordFactory::createRecord(
    1. "Jack". "Herrington". "4250 San Jaquin Dr.". "".
    "Los Angeles", "CA". "90210". ""
  ):
  Srecords []- RecordFactory::createRecord(
    1, "Megan". "Cunningham". "2220 Toorak Rd.". "".
    "Toorak". "VIC". "3121". "Australia"
  );
  return Srecords;
}
Srecords = readRecords( );
foreach( Srecords as $r )
{
  Sclass = new ReflectionClass( $r ):
} print $class->getName( )." - ".Sr->id." - ".$r->first." - ".$r->last."\n"
?>
```

В первой части кода реализован базовый класс Record, а также производные классы USRecord и ForeignRecord. Они на самом деле являются довольно простыми оберточными структурами данных. После этого фабричный класс в зависимости от передаваемых в него данных может создавать объекты типа USRecord или ForeignRecord. Проверочный код в конце сценария добавляет несколько записей, а затем выводит на экран их типы, а также хранимые в них данные.

Запуск трюка

Этот трюк следует запускать при помощи PHP-интерпретатора для командной строки:

```
% php abs_factory.php
USRecord - 1 - Jack - Herrington
ForeignRecord - 1 - Megan - Cunningham
```

Есть несколько вариантов того, как вы можете использовать шаблон Абстрактная фабрика в приложениях на PHP, работающих с базами данных.

- **Создавать объекты баз данных.** Фабрика может предоставить в ваше распоряжение объект любого типа, взаимодействующий с любыми таблицами в базе данных.
- **Создавать мобильные объекты.** Фабрика может предоставить в ваше распоряжение множество различных объектов в зависимости как от типа операционной системы, так и от базы данных, с которой связано приложение.
- **Создавать по стандарту.** Приложение может поддерживать файлы различных форматов и использовать фабрику для создания объектов, соответствующих заданному типу файла. Программы для чтения файлов могут зарегистрироваться на фабрике, чтобы добавить поддержку определенных файлов и не вносить никаких изменений ниже по коду.

После того как вы немного поработаете с шаблонами, вы сами научитесь понимать, в каких случаях есть смысл использовать какой-то определенный шаблон. Это те ситуации, когда вам необходимо разработать множество объектов различных типов. Вы поймете, что вам понадобится изменять и типы полученных объектов, и то, как они были созданы. Кроме того, вам придется внести огромное количество изменений в код. Если же вы воспользуетесь фабрикой, то вам придется изменить только ту часть кода, которая отвечает за создание объектов.

Смотрите также

J «Создание гибких объектов при помощи Фабричных методов» (см. трюк 69).

Создание гибких объектов при помощи Фабричных методов

Используйте шаблон Фабричный метод (Factory Method) для создания объектов, позволяющих изменять тип производных классов.

Шаблон Фабричный метод имеет очень близкое отношение к шаблону Абстрактная фабрика. Принципы их работы очень схожи. Если у вас есть класс, который создает множество объектов, то вы можете воспользоваться защищенным методом, чтобы инкапсулировать создание объекта. В этом случае производный класс

может просто перегрузить защищенные методы в фабрике, чтобы создать объект другого типа.

В трюке, вместо того чтобы использовать класс Factory, класс RecordReader для создания нового объекта Record использует метод под названием newRecord(). В этом случае производный от RecordReader класс может изменить тип созданных при помощи перегруженного метода newRecord() объектов Record. Этот процесс графически изображен на рис. 7.3.

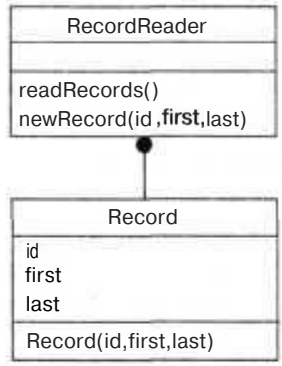


Рис. 7.3. Взаимодействие между RecordReader и Record

Код

Сохраните исходный код из примера 7.3 в файл с именем factory_method.php.

Пример 7.3. Пример использования фабричных методов в классе

```
<?php
class Record
{
  public $id - null;
  public $first = null;
  public $last = null;
  public function Record( $id. Sfirst. $last )
  {
    $this->id = Sid:
    $this->first = Sfirst;
    $this->last - $last;
  }
}

class RecordReader
{
  function readRecords( )
  {
    Srecords - array( ):
    Srecords []= $this->newRecord( 1, "Jack", "Herrington" ):
    Srecords []= $this->newRecord( 2, "Lori". "Herrington" );
    Srecords []= $this->newRecord( 3. "Megan", "Herrington" );
    return Srecords:
  }
```

```
  protected function newRecord( $id. $first, $last )
  {
    return new Record( $id, Sfirst, $last ):
  }
}
$rr = new RecordReader( );
$records = $rr->readRecords( );
foreach( Srecords as $r )
{
  print $r->id." - ".$r->first." - ".$r->last."\n":
}
?>
```

Запуск трюка

Запустите этот трюк из командной строки, используя PHP-интерпретатор:

```
% php factory_method.php
1 - Jack - Herrington
2 - Lori - Herrington
3 - Megan - Herrington
```

В этом коде объект RecordReader подвергается обработке, а также вызывается его метод readRecords(). Он, в свою очередь, вызывает метод newRecord() для создания всех объектов Record. Затем созданные объекты выводятся на экран в цикле foreach.

Более наглядный пример шаблона Фабричный метод приведен в спецификации XML DOM API, устанавливаемой при инсталляции PHP 5. Объект DOMDocument, который является корнем определенного DOM-дерева, содержит несколько фабричных методов: createElement(), createAttribute(), createTextNode() и т. д. Любой вызов из объекта DOMDocument может заместить эти методы, чтобы изменить объекты, созданные при загрузке XML-дерева с диска или строк.

Как и при использовании шаблона Абстрактная фабрика, основным показателем того, что вы должны применять шаблон Фабричный метод, является необходимость написания большого и сложного кода для создания объекта. Использование шаблона Абстрактная фабрика или Фабричный метод гарантирует, что, когда вы изменяете код, создавая типы объектов или используя уже созданные объекты, воздействие на ваш код будет минимальным.

Смотрите также

- «Создание объектов при помощи Абстрактной фабрики» (см. трюк 68).

ТРЮК №70

Выделение кода создания структур при помощи шаблона Строитель

Используйте шаблон Строитель (Builder) для выделения кода, реализующего многократное создание структур, формирующих HTML или текст для сообщения по электронной почте.

Я всегда считал, что создающий что-либо код — самый элегантный код в системе. Я пришел к этому выводу после того, как целый год писал книгу, полностью посвященную коду, формирующему структуры.

ПРИМЕЧАНИЕ

Кстати, книга называется «Генерирование кода в действии». Она все еще доступна и может стать отличным подарком для друзей или членов семьи.

Примером кода, формирующего структуры, может послужить код, считывающий XML-документ с диска и создающий его представление в памяти. Другой пример — код, создающий сообщения для отправки по электронной почте, в котором сообщается, что у пользователя закончилась внесенная предоплата.

Именно на создании сообщений такого рода я хочу остановиться, но собираюсь немного схитрить. Я хочу воспользоваться шаблоном Строитель, чтобы при помощи одного и того же кода создавать HTML, XHTML и текстовые версии сообщений. Я собираюсь написать код, который будет создавать сообщения о просроченных платежах и использовать для этого шаблон Строитель вместо того, чтобы напрямую формировать строку. У этого объекта-строителя будет несколько методов, как вы можете это видеть на рис. 7.4. Методы `startBody()` и `endBody()` — это оберточные функции для создания сообщений. Метод addText () добавляет текст, а метод `addBreak()` вставляет перенос каретки.

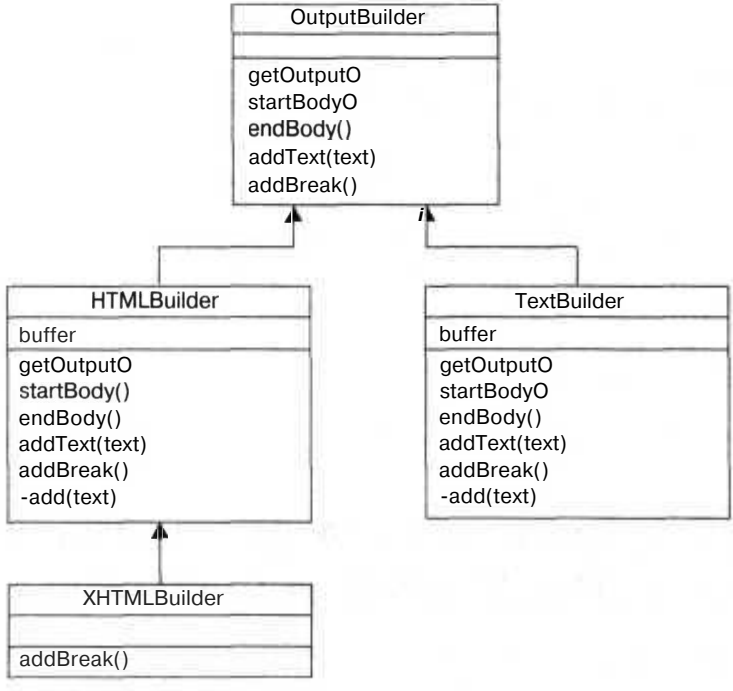


Рис. 7.4. Иерархия выходных данных строителя

У абстрактного класса OutputBuilder есть несколько конкретных реализаций. Одна из них — класс HTMLBuilder, который формирует HTML. От него наследуется класс XHTMLBuilder, исправленный так, чтобы научиться формировать XHTML. И, наконец, есть класс TextBuilder, который создает текстовый вариант сообщения.

Код

Сохраните исходный код из примера 7.4 в файл builder.php.

Пример 7.4. Несколько примеров классов-строителей и небольшой тестовый код

```
<?php
abstract class OutputBuilder
{
  abstract function getOutput( ):
  abstract function startBody( ):
  abstract function endBody( ):
  abstract function addText( $text ):
  abstract function addBreak( ):
}
class HTMLBuilder extends OutputBuilder
{
  private $buffer = "";
  public function getOutput( )
  {
    return "<html>\n".$this->buffer."\n</html>\n";
  }
  public function startBody( ) { $this->add( "<body>" ); }
  public function endBody( ) { $this->add( "</body>" ); }
  public function addText( $text ) { $this->add( $text ); }
  public function addBreak( ) { $this->add( "<br>\n" ); }
  protected function add( $text ) { $this->buffer .= $text; }
}
class XHTMLBuilder extends HTMLBuilder
{
  public function addBreak( ) { $this->add( "<br />\n" ); }
}
class TextBuilder extends OutputBuilder
{
  private $buffer = "";
  public function getOutput( )
  {
    return $this->buffer."\n";
  }
  public function startBody( ) { }
  public function endBody( ) { }
  public function addText( $text ) { $this->add( $text ); }
  public function addBreak( ) { $this->add( "\n" ); }
  protected function add( $text ) { $this->buffer .= $text; }
}
function buildDocument( $builder )
{
  $builder->startBody( ):
```

```
    $builder->addText( 'Jack,' );
    $builder->addBreak( );
    $builder->addText( 'You owe us $10.000. Have a NICE day.' );
    $builder->endBody( ):
}
print "HTML:\n\n":
$html = new HTMLBuilder( );
buildDocument( $html ):
echo( $html->getOutput( ) );
print "\nXHTML:\n\n";
$xhtml = new XHTMLBuilder( );
buildDocument( $xhtml );
echo( $xhtml->getOutput( ) ):
print "\nText:\n\n":
$text = new TextBuilder( ):
buildDocument( $text ):
echo( $text->getOutput( ) ):
?>
```

Запуск трюка

Запустите этот трюк из командной строки, используя PHP-интерпретатор:

```
% php builder.php
HTML:
<html>
<body>Jack,<br>
You owe us $10.000. Have a NICE day.</body>
</html>
XHTML:
<html>
<body>Jack,<br />
You owe us $10.000. Have a NICE day.</body>
</html>
Text:
Jack.
You owe us $10.000. Have a NICE day.
```

Здесь продемонстрированы результаты работы трех различных строителей. Первый используется для создания сообщения с правильно расставленными HTML-элементами и тегом
. В версии для создания XHTML-сообщения тег
 изменяется на
. Текстовая же версия содержит простой текст, а также переносы каретки. Взглянув на код, вы увидите определение абстрактного класса OutputBui l der в начале файла, а затем следуют различные реализации для конкретных вариантов выходных данных. Функция buildDocument() использует строителя для создания сообщения. Код в конце файла предназначен для проверки работоспособности функции buildDocument() с различными версиями строителей.

В PHP-приложениях шаблон Строитель вы можете использовать в следующих случаях.

- **Для чтения файлов.** Каждый раз, когда вы анализируете содержимое файла, вам следует использовать шаблон Строитель, чтобы разделить обработку файла от создания размещаемых в памяти структур данных, содержащих информацию из файла.

- Для **записи в файл.** Как я показал в этом трюке, вы можете использовать шаблон Строитель для создания выходных данных в нескольких форматах, используя для этого один и тот же документ.
- **Для автоматического формирования кода.** Вы можете использовать строителей для формирования кода для нескольких языков программирования, и все это — в одной системе.

.NET Framework использует шаблон Строитель при создании HTML-кода для страницы, чтобы позволить одному и тому же элементу управления формировать несколько вариантов HTML исходя из типа браузера, запрашивающего страницу.

Отделение «что» от «как» при помощи Стратегий

Используйте шаблон Стратегия (Strategy), чтобы отделять код, передающий структуры данных, от кода, работающего с этими структурами.

Шаблон Стратегия используется для отделения процесса работы с объектом от способа его размещения. В этом трюке в качестве примера я воспользуюсь системой для подбора автомобилей. Код будет советовать вам автомобиль исходя из заданного вами критерия поиска. В данном случае я буду задавать характеристики идеального в моем понимании автомобиля, а код будет подбирать наиболее подходящий этим требованиям автомобиль. Вся ценность шаблона Стратегия в том, что с его помощью я могу отделить код сравнения автомобилей от кода, подбирающего автомобиль. Схему UML для этого трюка вы можете видеть на рис. 7.5. Объект CarChooser использует объект `CarWeighter` для сравнения каждого объекта `Car` с идеальной моделью. В результате клиенту возвращается наиболее подходящий объект типа `Car`.

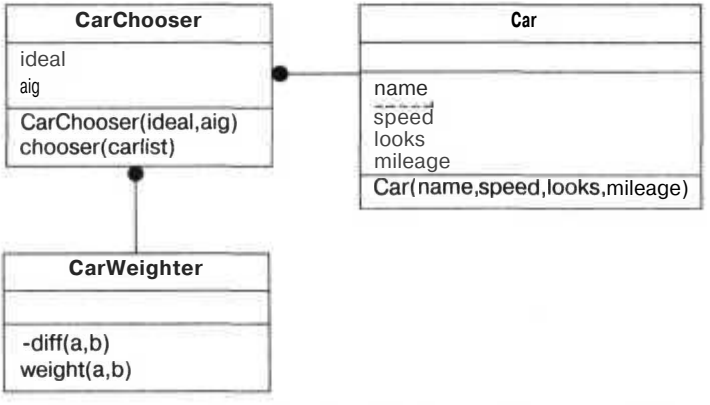


Рис. 7.5. Взаимодействие объектов CarChooser, CarWeighter и Car

Код

Сохраните исходный код из примера 7.5 в файл strategy.php.

Пример 7.5. Пример использования шаблона Стратегия

```
<?php
class Car
{
  public $name;
  public $speed;
  public $looks;
  public $mileage;
  public function Car( $name, $speed, $looks, $mileage )
  {
    $this->name = $name;
    $this->speed = $speed;
    $this->looks = $looks;
    $this->mileage = $mileage;
  }
}

class CarWeighter
{
  private function diff( $a, $b )
  {
    return abs( $a - $b );
  }
  public function weight( $a, $b )
  {
    $d=0;
    $d += $this->diff( $a->speed, $b->speed );
    $d += $this->diff( $a->looks, $b->looks );
    $d += $this->diff( $a->mileage, $b->mileage )
    return ( 0 - $d );
  }
}
class CarChooser
{
  private $ideal;
  private $alg;
  function CarChooser( $ideal, $alg )
  {
    $this->ideal = $ideal;
    $this->alg - $alg;
  }
  public function choose( $carlist )
  {
    $minrank = null;
    $found = null;
    $alg = $this->alg;
    foreach( $carlist as $car )
    {
      $rank = $alg->weight( $this->ideal, $car );
      if ( !isset( $minrank ) ) $minrank - $rank;
      if ( $rank >= $minrank )
```

```
        {
            $minrank = $rank;
            $found - $car;
        }
    }
    return $found;
  }
}

function pickCar( $car )
{
  $carlist = array( );
  $carlist [ ] - new Car( "zippy", 90. 30, 10 );
  $carlist []= new Car( "mom'n'pop", 45, 30, 55 );
  $carlist []= new Car( "beauty", 40, 90. 10 );
  $carlist [ ] - new Car( "enviro". 40. 40. 90 ):
  $cw = new CarWeighter( );
  $cc - new CarChooser( $car. $cw ):
  $found - $cc->choose( $carlist ):
  echo( $found->name."\n" );
}
pickCar( new Car( "ideal". 80. 40. 10 ) );
pickCar( new Car( "ideal". 40. 90. 10 ) );
```

В самом начале файла я определяю класс `Car`, в котором хранится название автомобиля и его характеристики, такие как скорость, внешний вид и пробег. Эти параметры могут принимать значения от 0 до 100 (это сделано для простоты математических вычислений). Затем следует класс `CarWeighter`, который изучает два автомобиля и возвращает сравнительные характеристики. И, наконец, есть класс `CarChooser`, который использует объект типа CarWeighter для выбора лучшего автомобиля исходя из заданных критериев. Функция `pickCar()` заносит в список несколько автомобилей, а затем использует объект типа CarChooser для выбора из списка наиболее подходящего автомобиля (и возвращает его в виде другого объекта типа `Car`). Код в конце файла, предназначенный для проверки, запрашивает два автомобиля: один с упором на скорость, а другой с упором на внешний вид.

Запуск трюка

Запустите этот трюк в командной строке при помощи PHP-интерпретатора:

```
% php strategy.php
zippy
beauty
```

Взглянув на выведенную на экран информацию, вы увидите, что в качестве быстрой машины мне рекомендуют автомобиль под названием «zippy», что можно назвать неплохим результатом. Если же я хочу что-то более сексуальное, то мне рекомендуют остановить свой выбор на автомобиле под названием «beauty». Отлично! Код, который определяет, подходит данный автомобиль или нет, полностью абстрагирован от кода, помещающего автомобили в список и извлекающего их оттуда. Вы можете изменить алгоритм, оценивающий конкретный

автомобиль независимо от алгоритма, выбирающего наиболее подходящее авто из списка. Например, вы можете учитывать автомобили, которыми вы интересовались или которые принадлежали вам раньше, в алгоритм, рассчитывающий характеристики автомобиля. Вы также можете изменить алгоритм кода, отбирающего автомобили из списка, так, чтобы он выдавал вам три наиболее подходящих автомобиля.

Организация связей между двумя модулями при помощи переходника

Используйте класс-переходник (adapter class) для передачи данных между двумя модулями, когда вы не хотите изменять API ни один из этих модулей.

Иногда вам надо получить данные из двух модулей, каждый из которых использует свой собственный формат для их передачи. Изменять один или сразу оба модуля — не лучший выход, так как вам придется вносить изменения и в весь остальной код. Одним из решений этой проблемы может быть использование класса-переходника. Переходник — это класс, который понимает передаваемые в него от обоих объектов данные и служит для того, чтобы один объект мог общаться с другим.

Продемонстрированный в этом трюке переходник преобразовывает данные из объекта, имитирующего базу данных, так, чтобы они подходили для процессора, работающего с текстом и диаграммами. На рис. 7.6 изображен объект RecordGraphAdapter, расположенный между TextGraph, который находится слева от него, и `RecordList`, находящимся справа от него. В объекте TextGraph очень хорошо описан формат данных, которые он ожидает для приема от класса под названием TextGraphDataSource. Класс же `RecordList` используется в качестве хранилища и содержит список Records, а в каждом объекте типа Record содержится имя, возраст и зарплата.

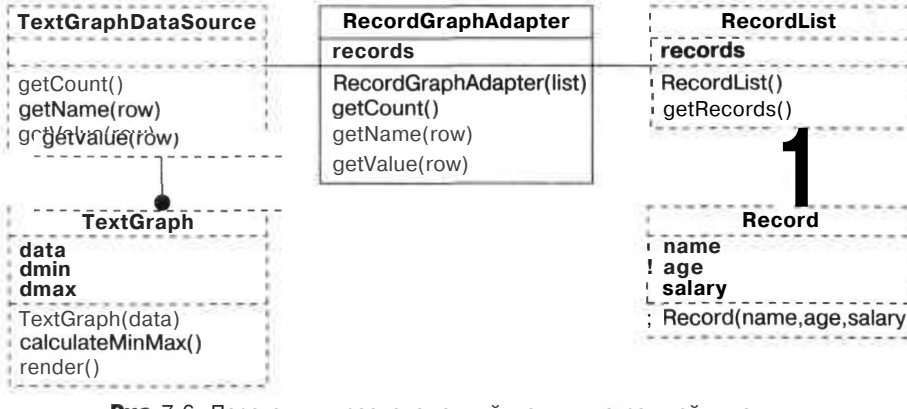


Рис. 7.6. Переходник, расположенный между диаграммой и данными

Для этого примера нужна диаграмма с зарплатами. Работа переходника заключается в извлечении данных из RecordList и преобразовании их к виду, понятному TextGraph. Для этого их нужно передать в объект TextGraphDataSource.

Код

Сохраните исходный код из примера 7.6 в файл с именем adapter.php.

Пример 7.6. Переходник, использующий диаграммы с текстом

```
<?php
abstract class TextGraphDataSource
{
  abstract function getCount( ):
  abstract function getName( $row ):
  abstract function getValue( $row );
}
class TextGraph
{
  private $data:
  private $dmin;
  private $dmax;
  public function TextGraph( $data )
  {
    $this->data = $data;
  }
  protected function calculateMinMax( )
  {
    $this->dmin = 100000;
    $this->dmax = -100000;
    for( $r = 0; $r < $this->data->getCount( ): $r++ )
    {
      $v - $this->data->getValue( $r );
      if ( $v < $this->dmin ) { $this->dmin - $v; }
      if ( $v > $this->dmax ) { $this->dmax = $v; }
    }
  }
  public function render( )
  {
    $this->calculateMinMax( );
    $ratio = 40 / ( $this->dmax - $this->dmin );
    for( $r = 0; $r < $this->data->getCount( ); $r++ )
    {
      $n = $this->data->getName( $r ):
      $v = $this->data->getValue( $r );
      $s = ( $v - $this->dmin ) * $ratio;
      echo( sprintf( "%10s : ". $n ) );
    } for( $st = 0; $st < $s; $st++ ) { echo("*"): }
    echo( "\n" );
}

class Record
{
```

```
  public $name;
  public Sage;
  public Ssalary;
  public function Record( $name. Sage. Ssalary )
  {
    $this->name = Sname;
    $this->age = $age;
    $this->salary = Ssalary;
  }
}

class RecordList
{
  private $records - array( ):
  public function RecordList( )
  {
    $this->records []= new Record( "Jimmy". 23. 26000 );
    $this->records []= new Record( "Betty". 24. 29000 ):
    $this->records []= new Record( "Sally". 28. 42000 ):
    $this->records [] - new Record( "Jerry". 28. 120000 );
    $this->records [] - new Record( "George". 43. 204000 );
  }
  public function getRecords( )
  {
    return $this->records;
  }
}

class RecordGraphAdapter extends TextGraphDataSource
{
  private Srecords:
  public function RecordGraphAdapter( $rl )
  {
    $this->records = $rl->getRecords( );
  }
  public function getCount( )
  {
    return count( $this->records ):
  }
  public function getName( $row )
  {
     return $this->records[ $row ]->name;
  }
  public function getValue( $row )
  {
    return $this->records[ Srow ]->salary;
  }
}
$rl = new RecordList( ):
Sga = new RecordGraphAdapter( $rl );
$tg = new TextGraph( Sga ):
Stg->render( );
?>
```

Верхняя часть файла предназначена для диаграммы. Там определяется абстрактный класс TextGraphDataSource, а также класс TextGraph. Он использует класс TextGraphDataSource для передачи данных. В середине файла определены классы

Record и RecordList (содержащие данные, которые требуется вывести в виде диаграммы). В оставшейся части файла определяется класс RecordGraphAdapter, преобразовывающий RecordList в формат, который уже можно использовать в диаграмме.

Исходный код, расположенный в самом конце файла, создает объект типа RecordList, затем — переходник, а также объект типа TextGraph, связанный с переходником. После того как данные считываются из адаптера, они заносятся в диаграмму.

Запуск трюка

Для запуска этого трюка воспользуйтесь PHP-интерпретатором для командной строки:

```
% php adapter.php
Jimmy :
Betty : *
Sally : ****
Jerry : **********
George : ****************************************
```

Наименьшая шкала в диаграмме у Jimmy, а лидирует George. Диаграмма расширяется автоматически, поэтому у Jimmy вообще нет звездочек (минимальное значение), а у George 40 звездочек (максимальное значение). Так держать, George! Но что действительно важно — этот код справился с преобразованием данных без каких либо проблем, и для этого ему не понадобилось «копаться» в классе Record. Всегда используйте переходник, когда у вас есть два API, которые должны работать вместе, и когда нет смысла изменять ни один из них.

ТРЮК №73

Создание переносного кода при помощи шаблона Мост

Используйте шаблон Мост (Bridge), чтобы скрывать реализацию объекта или изменять ее в зависимости от используемой среды.

Однажды, когда я работал на одну компанию, создававшую очень большое кроссплатформенное приложение на C++, мы решили воспользоваться шаблоном Мост. Шаблон Мост позволяет вам скрывать в другом классе часть кода, содержащую реализацию класса, в тех случаях, когда вы хотите скрыть его от других конструкторов или когда эта реализация кода зависит от платформы. В этом трюке мы рассмотрим второй вариант, и я расскажу вам обо всех преимуществах использования этого моста. На рис. 7.7 показано взаимодействие между классами TableCreator и TableCreatorImp. Роль первого заключается в создании таблиц в указанной базе данных. Класс для реализации TableCreatorImp определяется в другом файле, который подключается из указанной в базе данных директории. Такой гибкий подход позволяет иметь одну версию исходного кода, предназначенного для работы с Oracle и одну для работы с MySQL (или какой-либо другой базой данных).

Это очень удобно, так как в каждой из этих баз данных для создания таблиц используется различный синтаксис.

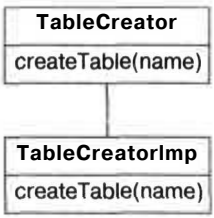


Рис. 7.7. Класс TableCreator и класс для реализации

Код

Сохраните исходный код из примера 7.7 в файл с именем `bridge.php`.

Пример 7.7. Базовый класс для шаблона Мост

```
<?php
require( "sql.php" ):
class TableCreator
{
  static function createTable( $name )
  {
    TableCreatorImp::createTable( $name ):
  }
}
TableCreator::createTable( "customer" ):
?>
```

Сохраните исходный код из примера 7.8 в файл `mysql/sql.php`.

Пример 7.8. Пример реализации класса для MySQL

```
<?php
class TableCreatorImp
{
  static public function createTable( $name )
  {
    echo( "MySQL version of createTable for $name\n" ):
  }
}
?>
```

Сохраните исходный код из примера 7.9 в файл oracle/sql .php.

Пример 7.9. Пример реализации для Oracle

```
<?php
class TableCreatorImp
{
  static public function createTable( $name )
  {
    echo( "Orac'e version of createTable for $name\n" );
  }
}
?>
```

Запуск трюка

Для запуска этого кода в командной строке нужно использовать некоторые дополнительные параметры, указывающие PHP-интерпретатору на то, какую нужно использовать директорию: mysql или oracle в качестве пути для подключения файла (и для этого мы воспользуемся указанным в базе данных мостом). Рассмотрим вариант, в котором будет использоваться версия кода для MySQL:

```
% php -d include_path='.:/usr/local/php5/lib/php:mysql' bridge.php
MySQL version of createTable for customer
```

В следующем варианте используется код для Oracle:

```
% php -d include_path='.:/usr/local/php5/lib/php:oracle' bridge.php
Oracle version of createTable for customer
```

Это не так сложно, как ракетостроение, поэтому вы должны все очень быстро понять. На самом деле есть несколько версий реализации класса `TableCreator`, расположенных в директориях, указывающих на используемую платформу. Очевидно, что написанный здесь код не создает таблицы. Это всего лишь каркас, в который следует добавить код, чтобы он на самом деле заработал. Но специфика создания таблиц в базах данных не имеет отношения к шаблону Мост (представьте, что я сказал вам: «Вы можете изучить этот вопрос самостоятельно»). Один из больших недостатков мостов заключается в том, что вы не можете расширить возможности классов, зависящих от реализации. В данном случае этого не требуется, так как все методы в реализации класса статические. Но в случае с регулярными объектами с нестатическими методами класс для реализации является производным от класса с отсутствующей реализацией. Например, класс `CButtonImp` является производным от `CButton`. Чтобы добавить в него какие-либо дополнительные возможности, коду потребуется создать класс, производный от класса `CbuttonImp` со скрытой реализацией. Тем не менее, можно поспорить, что эта проблема актуальна только для языков, использующих компиляторы, например для C++.

ТРЮК №74

Реализация расширяемой обработки при помощи Цепочек обязанностей

Используйте шаблон Цепочка обязанностей (Chain of Responsibility) при создании каркаса для обработки данных в режиме Plug and Play (PnP).

Смотреть футбол с программистами очень увлекательно. Даже когда в четвертом тайме счет 33:7, а до конца матча остается полторы минуты, вы все еще рискуете получить довольно большой диапазон ответов на вопрос о том, кто же победит. Это все потому, что программисты привыкли брать в расчет любую потенциально возможную ситуацию, вне зависимости от того, насколько она маловероятна (и, как правило, даже абсурдна). Я также заметил, что большинство программистов,

включая меня самого, не любят закрывать глаза на какие-либо возникшие вопросы. Всегда намного проще написать код, в котором будет учитываться 100 возможных вариантов развития событий, даже если ваш менеджер клянется, что возможен будет только один из них.

Вот почему я считаю, что использовать шаблон Цепочка обязанностей в таких ситуациях очень удобно. Представьте, что вы зашли в заполненную людьми комнату и у вас есть коробка с пончиками. Вы приоткрываете упаковку и достаете из нее пончик с желе. И тут вы начинаете спрашивать присутствующих людей одного за другим: «Хотите пончик с повидлом?», пока не находите того, кто действительно хочет. И так вы поступаете со всеми пончиками с разной начинкой, пока они не закончатся. Это и есть шаблон Цепочка обязанностей: каждый из присутствующих в комнате заранее регистрируется у поставщика пончиков. Затем, по мере того как доставляются сами пончики, вы можете узнать, кто какой пончик хочет, просмотрев ваш список зарегистрировавшихся в нем потребителей. Плюс в том, что вас, как поставщика, не волнует, как много людей хотят ваши пончики, вам даже абсолютно безразлично, что они с ними делают. Все, что вам нужно делать, — это регистрировать их и поставлять пончики.

В этом трюке я написал код, который представляет URL-адреса в виде таких пончиков. На самом деле код просто передает эти URL группе обработчиков, которые, возможно, преобразуют его. Если URL-адрес не принят на обработку, то он будет просто удален и проигнорирован.

На рис. 7.8 показано, как это все будет работать. `URLMapper` выступает в роли поставщика пончиков. У него есть наполненная URL-адресами коробка, которые он готов передать в любой объект, зарегистрированный в интерфейсе `URLHandler`. В нашем случае `ImageURLHandler` обрабатывает URL-запросы с картинками при помощи работающего с изображениями сценария. Таким же образом `DocumentURLHandler` будет перенаправлять любой запрос к документу на соответствующую PHP-страницу. Такой подход позволяет нам рассылать URL, не создавая какого-либо специального кода для обработки, но тем не менее они все-таки будут преобразованы так, как это и требуется в вашем приложении.

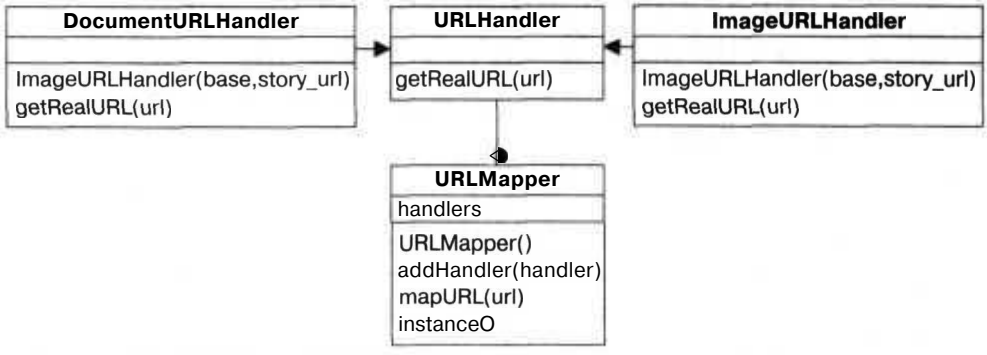


Рис. 7.8. Интерфейс URLHandler, преобразователь и два обработчика

Код

Сохраните исходный код из примера 7.10 в файл с именем chain.php.

Пример 7.10. Пример использования в PHP шаблона Цепочка обязанностей

```
<?php
abstract class URLHandler
{
  abstract function getRealURU $url );
}

class URLMapper
{
  private $handlers - array( );
  private function URLMapper( )
  {
  }
  public function addHandler( $handler )
  {
    $this->handlers []= $handler;
  }
  public function mapURL( $url )
  {
    foreach( $this->handlers as $h )
    {
      $mapped = $h->getRealURL( $url );
      if ( isset( $mapped ) ) return Smapped:
    }
    return $url;
  }
  public static function instance( )
  {
    static $inst = null;
  } if( !isset( $inst ) ) { $inst - new URLMapper( ); }
}   return $inst:

class ImageURLHandler extends URLHandler
{
  private $base;
  private $imgurl;
  public function ImageURLHandler( $base. Simgurl )
  {
    $this->base - $base;
    $this->imgurl = Simgurl;
  }
  public function getRealURU $url )
  {
    if ( preg_match( "|^".$this->base."(.*?)$|", $url, $matches ) )
    {
      return $this->imgurl.$matches[1];
    }
    return null;
  }
```

```
}

class StoryURLHandler extends URLHandler
{
  private $base;
  private $story_url;
  public function StoryURLHandler( $base. $story_url )
  {
    $this->base - $base:
    $this->story_url = $story_url;
  }
  public function getRealURL( $url )
  {
    if ( preg_match( "|^".$this->base."(.*?)/(.*?)/(.*?)$|", $url,
      $matches ) )
    {
      return $this->story_url.$matches[1].$matches[2].$matches[3];
    }
    return null;
  }
}
$ih - new ImageURLHandler( "http://mysite.com/images/"
"http://mysite.com/image.php?img=" ):
URLMapper::instance( )->addHandler( $ih ):
$ih = new StoryURLHandler( "http://mysite.com/story/",
"http://mysite.com/story.php?id=" ):
URLMapper::instance( )->addHandler( $ih );
$testurls = array( ):
$testurls [ ]- "http://mysite.com/index.html";
$testurls [ ]- "http://mysite.com/images/dog";
$testurls [ ]- "http://mysite.com/story/11/05/05";
$testurls  []= "http://mysite.com/images/cat";
$testurls [ ]- "http://mysite.com/image.php?img=lizard";
foreach( $testurls as $in )
{
  $out - URLMapper::instance( )->mapURL( $in );
  print "$in\n --> $out\n\n";
}
?>
```

Запуск трюка

Запустите сценарий chain.php при помощи PHP-интерпретатора для командной строки:

```
% php chain.php
http://mysite.com/index.html
--> http://mysite.com/index.html
http://mysite.com/images/dog
-->http://mysite.com/image.php?img=dog
http://mysite.com/story/11/05/05
--> http://mysite.com/story.php?id=110505
http://mysite.com/images/cat
--> http://mysite.com/image.php?img=cat
http://mysite.com/image.php?img=lizard
--> http://mysite.com/image.php?img=lizard
%
```

Каждый добавленный URL посылается в URLMapper, который возвращает преобразованный адрес. В первом случае URL не обрабатывается, поэтому он просто удаляется. Во втором случае `ImageURLHandler` видит, что это URL-адрес картинки, поэтому он преобразует его при помощи сценария `image.php`. Третий URL распознается обработчиком документов, который преобразует его при помощи сценария `story.php`.

Огромная ценность шаблона Цепочка обязанностей заключается в том, что вы можете расширять его возможности, не внося изменения в код основной программы. По мере того как поставщик объектов обзаведется солидным API для зарегистрированных объектов, он сможет обработать практически любую ситуацию.

Возможно, одним из наиболее ярких примеров использования шаблона Цепочка обязанностей является веб-сервер Apache. Все его функции похожи на одного большого поставщика пончиков, передающего различные запросы зарегистрированным в нем обработчикам.

ВНИМАНИЕ

Использовать этот шаблон не всегда легко. На самом деле работа с этим шаблоном требует некоторой серьезной подготовки! Он довольно сложен в отладке, и не всегда предельно ясно, как правильно им воспользоваться. Есть также два варианта этого шаблона: в первом запрос останавливается, если найден его обработчик, а во втором запрос продолжает свою работу вне зависимости от того, был ли найден для него соответствующий обработчик. Не всегда понятно, какую из этих версий следует использовать. К тому же, при использовании второго варианта, в котором все обработчики вызываются во всех ситуациях, отладка может быть крайне затруднительной (даже больше чем обычно в шаблонах). Просто всегда помните, что богатство возможностей компьютерных программ всегда идет в ущерб их сложности и производительности.

Разбиение больших классов на части при помощи Компоновщика

Используйте шаблон Компоновщик (Composite) для разбиения очень больших классов на более мелкие и простые в управлении.

Мне бывает очень любопытно, когда я слышу в новостях о гигантских базах данных с практически любой информацией о каждом человеке, которая только может понадобиться людям. В то время как большинство людей озабочено вмешательством в их частную жизнь, меня больше угнетает то, что, скорее всего, все эти базы данных очень плохо спроектированы. Я практически уверен, что в них используется ужасный мегаобъект, называемый Person, у которого около 4000 полей и 8000 методов.

Откуда же я узнал об этом? Раньше я видел подобные объекты и работал с ними! Именно для классов такого рода и необходим шаблон Компоновщик. Шаблон Компоновщик оставит класс Person, но разобьет на группы все его 4000 полей

и разместит их в дочерних объектах. На самом деле объект Person будет состоять из 100 или около того объектов, каждый из которых будет включать в себя другие, более мелкие объекты (которые также будут состоять из еще более мелких и т. д.).

Я не хочу сказать, что я встречался с настолько плохими классами, реализованными на PHP, но я встречался с классами, в которых содержалось более чем 100 полей, и все это просто из-за того, что они предназначены для отображения таблиц, состоящих из большого количества полей, имеющих отношение к одной единственной записи. В этом трюке показано, как разбить класс Customer (в котором даже намного больше полей) на несколько более мелких классов, но чтобы в итоге он так и остался одним цельным классом Customer.

ПРИМЕЧАНИЕ

Му, на самом деле в этом примере я работаю с классом, в котором восемь **или** около того полей — я позволю вам самим расширить класс до нескольких сотен полей — **именно** с такими объектами я и встречался раньше.

Структура Customer показана на рис. 7.9. Класс Customer содержит один объект типа CustomerName и один типа CustomerAddress.

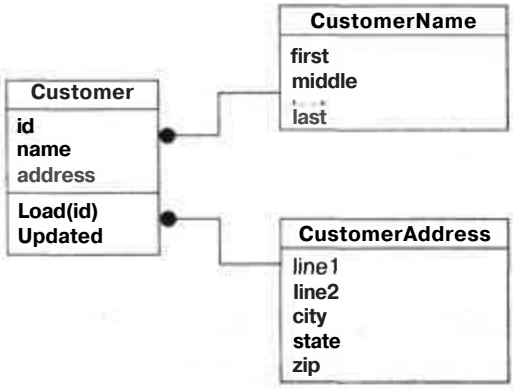


Рис. 7.9. Составной класс Customer и входящие в него дочерние классы

Код

Сохраните исходный код из примера 7.11 в файл composite.php.

Пример 7.11. Составной объект, включающий в себя несколько более мелких объектов

```
<?php
class CustomerName
{
  public $first = "";
  public $middle = "";
  public $last = "";
}

class CustomerAddress
{
```

```
  public $line1 = "";
  public $line2 - "";
  public $city » "";
  public $state - "":
  public $zip = "";
}

class Customer
{
  public $id - null;
  public $name - null;
  public $address = null;
  public function Customer( )
  {
    $this->name = new CustomerName( ):
    $this->address = new CustomerAddress( );
  }
  public function Load( $id )
  {
    $this->id » Sid:
    $this->name->first = "George";
    $this->name->middle - "W";
    $this->name->last = "Bush";
    $this->address->line1 = "1600 Pennsylvania Ave.";
    $this->address->line2 - "";
    $this->address->city = "Washington";
    $this->address->state = "DC";
    $this->address->zip - "20006";
  }
  public function Update( )
  {
    //Обновляем запись в базе данных
    //или вставляем запись, если нет такого id
  }
  public function___toString( )
  {
    return $this->name->first." ".$this->name->last;
}
$cust = new Customer( );
$cust->l oad( 1 );
print( $cust );
print( "\n" );
?>
```

Запуск трюка

Запустите этот сценарий из командной строки при помощи PHP-интерпретатора:

```
% php composite.php
George Bush
```

Код создает нового клиента и загружает запись под номером 1. У меня в коде строго установлено имя George W. Bush. Затем код выводит на экран содержимое объекта Customer. Не так-то и много, но чем идея проще, тем она эффективнее.

Вам не надо создавать метакласс с сотней полей. Лучше создать группу классов поменьше, таких как CustomerName и CustomerAddress, которые в дальнейшем вы сможете объединить во что-то большее, — в нашем случае это класс Customer. Это даже лучше, так как подкласс CustomerAddress можно использовать в дальнейшем, например при создании классов, в которых требуются поля с почтовым адресом.

Хорошей идеей также будет при использовании шаблона Компоновщик хранить данные для объекта раздельно в нескольких таблицах базы данных. Каждой таблице должен соответствовать объект или структура данных.

Использование шаблона Компоновщик также позволяет оптимизировать чтение данных из нее. Поскольку для загрузки каждого дочернего объекта, например адреса, требуется отдельный запрос, можно не спешить с загрузкой всех данных сразу. Другими словами, вы можете отложить загрузку какого-либо определенного объекта до тех пор, пока он вам не понадобится. Это поможет избежать вам таких ситуаций, когда приходится загружать содержимое сотен полей из нескольких таблиц, когда вам на самом деле всего лишь нужно, например, только имя и фамилия человека.

ТРЮК №76

Упрощение API при помощи Фасада

Используйте шаблон Фасад (Façade), чтобы упростить API, который вы предлагаете использовать другим программистам.

Очень хотелось бы, чтобы именно этот шаблон программисты использовали как можно чаще, и не только из-за забавной загогулины под буквой «с» в слове Façade. Просто дело в том что созданный фасад означает, что другие программисты подумали о вас и они хотят удостовериться, что вы получили всю нужную информацию (и вы ничего не сломаете). Взять хотя бы простой API для создания журналов, как в примере, показанном на рис. 7.10.

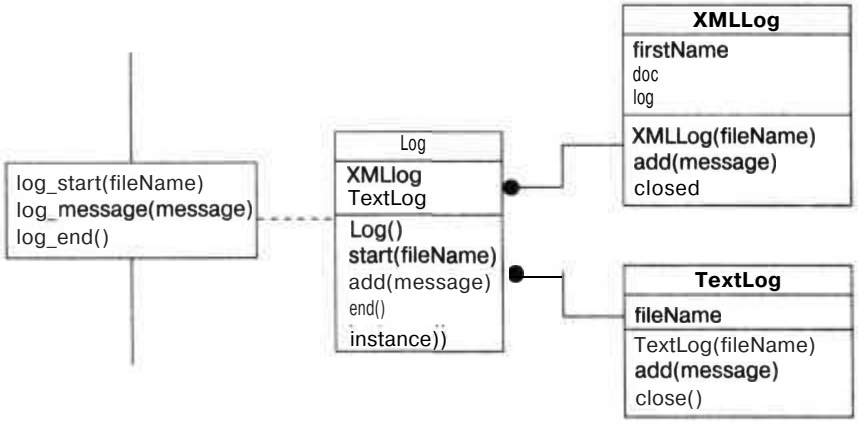


Рис. 7.10. API для создания журналов с простым фасадом

При помощи этого API можно вести журналы в XML-виде, в текстовом виде или и в том, и в другом одновременно. Ух ты! Будучи программистом, я просто восхищен мастерством, с которым был создан код. Кажется, там есть методы для всего: для начала сообщения, для добавления текста, для удаления... Даже для простой обработки текста и XML.

Но все, что мне на самом деле хотелось бы знать, — это то, какой метод и когда следует использовать. Это как раз то, чем и занимается фасад: он следит за тем, чтобы я правильно использовал API. В этом примере фасад — это список из трех функций, принимающих в качестве параметра и возвращающих в качестве результата текстовую строку. Эта строка является чем-то вроде теоретической закладки, которая говорит: «Я все сделаю правильно, просто вызови нужный метод, и я позабочусь обо всем остальном».

Фасад не только упрощает API, он также скрывает от клиента детали реализации, и на самом деле она может измениться, а пользователь даже не будет ничего знать об этом. Это важно настолько же, насколько и то, как фасад упрощает работу. Помните, что слабое связывание подразумевает под собой мощную и надежную систему.

Код

Сохраните исходный код из примера 7.12 в файл с именем test.php.

Пример 7.12. Пример кода для создания журналов

```
<?php
require( "log.php" );
log_start( "mylog" );
log_message( "Opening application" );
log_message( "Logging a message" );
log_message( "Shutting down" );
log_end( );
?>
```

Сохраните исходный код из примера 7.13 в файл log.php.

Пример 7.13. Фасад для журналов

```
<?php
require( "log_impl.php" );
function log_start( $fileName )
{
  Log::instance( )->start( $fileName );
}

function log_message( $message )
{
  Log::instance( )->add( $message );
}
function log_end( )
{
  Log::instance( )->end( );
}
?>
```

Исходный код из примера 7.14 сохраните в файл log_impl.php.

Пример 7.14. Библиотека для работы с журналами, скрытая за фасадом

```
<?php
class XMLLog
{
  private $fileName;
  private $doc;
  private $log;
  public function XMLLog( $fileName )
  {
    $this->fileName = $fileName;
    $this->doc » new DOMDocument( );
    $this->doc->formatOutput = true;
    $this->log = $this->doc->createElement( "log" );
    $this->doc->appendChild( $this->log );
  }
  public function add( $message )
  {
    $mess_obj - $this->doc->createElement( "message" );
    $text = $this->doc->createTextNode( $message );
    $mess_obj->appendChild( $text );
    $this->log->appendChild( $mess_obj );
  }
  public function close( )
  {
    $this->doc->save( $this->fileName );
  }
}

class TextLog
{
  private $fh;

  public function TextLog( $fileName )
  {
    $this->fh = fopen( $fileName, "w" );
  }

  public function add( $message )
  {
    fprintf( $this->fh, $message."\n" );
  }

  public function close( )
  {
    fclose( $this->fh );
  }
}

class Log
{
  private $xmlLog = null;
  private $textLog = null;
  public function Log( )
  {
```

```
  }
  public function start( $fileName )
  {
    $this->xmlLog = new XMLLog( $fileName.".xml" );
    $this->textLog = new TextLog( $fileName.".txt" );
  }
  public function add( $message )
  {
    $this->xmlLog->add( $message );
    $this->textLog->add( $message );
  }
  public function end( )
  {
    $this->xmlLog->close( );
    $this->textLog->close( );
  }
  public static function instance( )
  {
    static $inst = null;
    if ( !isset( $inst ) ) $inst = new Log( );
    return $inst;
  }
}
?>
```

Запуск трюка

Этот трюк следует запускать при помощи PHP-интерпретатора для командной строки:

```
% php test.php
% cat mylog.txt
Opening application
Logging a message
Shutting down
% cat mylog.xml
<?xml version="1.0"?>
<log>
<message>Opening application</message>
<message>Logging a message</message>
<message>Shutting down</message>
</log>
```

Не так-то и много, но это как раз тот код (а не результат его работы), который нас и интересует. В сценарии `test.php` создается журнал, отправляются несколько сообщений, и затем журнал закрывается. Для этого используются только три функции из сценария с фасадом `log.php`. На самом деле в идеале сторонних программистов и должен интересовать только файл `log.php`. В нем для создания двух журналов используется объект, созданный при помощи шаблона Одиночка (см. трюк 77) в сценарии `log_impl.php`. Затем он отправляет каждое сообщение в каждый из журналов, и в итоге создается соответствующий текстовый или XML-файл.

ТРЮК №77

Создание константных объектов при помощи шаблона Одиночка

Используйте шаблон Одиночка (Singleton) для создания обособленных в системе объектов.

Из всех шаблонов, описанных в созданной Бандой Четырех книге «Дизайнерские Шаблоны», ни один не используется так часто, как шаблон Одиночка, отчасти, возможно, из-за того, что это очень просто.

Одиночка — это объект, экземпляр которого может быть только один в системе. Отличным примером потенциального одиночки является управление базой данных. Для каждого экземпляра PHP-интерпретатора должен быть только один идентификатор базы данных. В этом трюке реализован как раз такой подход — одиночная версия идентификатора. На рис. 7.11 показана модель UML для одиночки, работающего с базой данных (я же говорил вам, что это просто!).

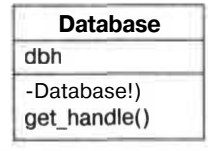


Рис. 7.11. Одиночка для базы данных

Действительно, не так-то и много. Объект может работать с базой данных, и у него есть два метода. Первый из них — конструктор, объявленный как `private`, чтобы удостовериться, что никто за пределами класса не может создать объект. Второй — статический метод под названием `get_handle`, возвращающий идентификатор базы данных.

Код

Сохраните исходный код из примера 7.15 в файл с именем `singleton1.php`.

Пример 7.15. Оберточный класс-одиночка для работы с базой данных

```
<?php
require( 'DB.php' ):
class Database
{
  private $dbh;
  private function Database( )
  {
    $dsn = 'mysql://root:password@localhost/test';
    $this->dbh =& DB::Connect( $dsn. array( ) ):
    if (PEAR::isError($this->dbh)) { die($this->dbh->getMessage( )): }
  }
  public static function get_handle( )
```

```
    static $db - null;
    if ( !isset($db) ) $db = new Database( );
    return $db->dbh:
  }
}
echo( Database::get_handle( )."\n" );
echo( Database::get_handle( )."\n" );
echo( Database::get_handle( )."\n" );
?>
```

У этого простого одиночки есть конструктор, который предназначен для авторизации в базе данных, а также один статический метод, который создает объект, если он еще не был создан, и возвращает идентификатор базы данных из этого объекта. Если вы будете использовать этот метод для получения доступа к базе данных, то можете быть спокойны — вы сможете соединиться с базой данных только один раз за загрузку страницы.

Запуск трюка

Запустите этот сценарий при помощи PHP-интерпретатора для командной строки:

```
% php singleton1.php
Object id #2
Object id #2
Object id #2
```

Здесь показано, что при нескольких вызовах статического метода `get_handle()` в результате возвращается один и тот же объект. Это означает, что при каждом вызове используется один и тот же объект Database и один и тот же идентификатор базы данных.

Улучшаем трюк

Возможность доступа к базе данных — это одно, а как же насчет чего-то более сложного? Попробуем создать открытый для общего доступа список штатов, как это показано в примере 7.16.

Пример 7.16. Массив-одиночка для штатов

```
<?pnp
class StateList
{
  private $states - array( ):
  private function StateList( )
  {
  }
  public function addState( $state )
  {
    $this->states []= $state:
  }
  public function getStates( )
  {
```

```
    return $this->states;
  }
  public static function instance( )
  {
    static $states = null;
    if ( !isset($states) ) $states = new StateList( );
    return $states;
  }
}
StateList::instance( )->addState( "Florida" );
var_dump( StateList::instance( )->getStates( ) );
StateList::instance( )->addState( "Kentucky" );
var_dump( StateList::instance( )->getStates( ) );
?>
```

Этот код создает класс-одиночку StateList, в котором содержится список штатов. Вы можете добавлять штаты в список, а также получать полный список всех находящихся в нем штатов. Для доступа к одному открытому для общего пользования экземпляру этого объекта вам придется воспользоваться статическим методом i nstance() (вместо того чтобы напрямую создать экземпляр). Чтобы запустить этот сценарий, воспользуйтесь интерпретатором для работы в командной строке:

```
% php singleton2.php
array(1) {
  [0]=>
  string(7) "Florida"
}
array(2) {
  [0]=>
  string(7) "Florida"
  [1]=>
  string(8) "Kentucky"
}
```

При первом запуске мы видим, что в списке есть только один элемент Florida. Затем добавляется элемент Kentucky.

Я немного колебался, прежде чем порекомендовать вам повсеместно использовать шаблон Одиночка. Исходя из моего опыта могу сказать, что его часто используют там, где это не требуется. На самом деле я чаще встречался с кодом, в котором этот шаблон притягивали за уши, чем с кодом, в котором этого не было, что говорит о том, что шаблон Одиночка используется некорректно. Если вы вынуждены предпринимать какие-то дополнительные меры, чтобы воспользоваться одиночкой, то может оказаться, что это неправильное использование шаблона.

ТРЮК №78

Упрощенная работа с данными при помощи Посетителей

Используйте шаблон Посетитель (Visitor) для разделения процесса просмотра данных и получения доступа к ним.

В начале моей карьеры программиста я провел много научной работы, связанной с системами сбора данных. Это были системы, которые записывали тестовые данные через заданные интервалы в 3 микросекунды, или 333 333 раза в секунду. Выходило около 38 Мбайт информации в минуту! За довольно небольшой промежуток времени размер файла уже исчислялся в гигабайтах. Нет необходимости говорить, что у нас возникали проблемы, связанные с записью и хранением таких объемов информации.

Другая проблема заключалась в попытке обработать эти данные. Как вы сможете обработать файл размером в несколько гигабайт, когда на вашей машине всего 128 Мбайт оперативной памяти? Необходимо разбивать данные на составные части. подразумевает под собой считывание фрагментов файла, а также сохранение нужных вам данных и удаление ненужных.

Стоит отметить, что эти научные алгоритмы, предназначенные для обмена данными, были достаточно трудоемкими, когда речь шла о работе с очень большими объемами информации. Чтобы решить эту проблему — и сделать это красиво — мы использовали . Один из объектов занимался извлечением данных из памяти, а другой обрабатывал уже загруженные в память данные. На рис. 7.12 показан объект `RecordList`, в котором содержится список Records. У него есть функция `iterate()`, которая при изменении используемой функции передает в нее каждую из записей. Благодаря этому функция для обработки данных при передаче ее в `iterate()` не должна беспокоиться о том, как эти записи расположены в памяти. Все, что ей надо делать, — работать с данными надлежащим образом.

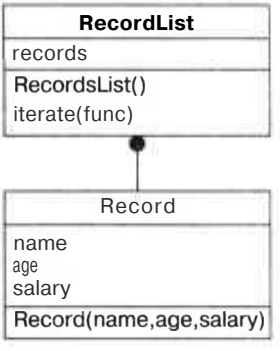


Рис. 7.12. Класс RecordList и метод iterate

Код

Сохраните исходный код из примера 7.17 в файл с именем `visitor1.php`.

Пример 7.17. Шаблон Посетитель, перемещающийся по записям из базы данных

```
<?php
class Record
{
  public $name;
  public Sage;
```

```
  public $salary;
  public function RecorcK $name, Sage, $salary )
  {
    $this->name - $name;
    $this->age = $age;
    $this->salary = Ssalary:
  }
}

class RecordList
{
  private $records - array( );
  public function RecordList( )
  {
    $this->records []= new Recordt "Larry". 22, 35000 );
    $this->records []= new Recordt "Harry". 25, 37000 ):
    $this->records []= new Recordt "Mary". 42. 65000 ):
    $this->records [] - new Recordt "Sally". 45. 80000 );
  }
  public function iterate( $func )
  {
    foreach( $this->records as $r )
    {
      call_user_func( $func. $r );
    }
  }
}
$min - 100000;
function find_min_salary( $rec )
{
  global $min;
  if( $rec->salary < $min ) { $min = $rec->salary; }
}
$rl = new RecordList( ):
$rl->iterate( "find_min_salary". $min ):
echo( $min."\n" );
?>
```

Запуск трюка

Запустите этот трюк из командной строки:

```
% php visitor1.php
35000
```

Алгоритм находит наименьшую зарплату из всех просмотренных записей. Код в этом сценарии очень простой. Класс Record получает данные из каждой записи. Затем класс RecordList предварительно загружает какую-то фиктивную информацию (в настоящих системах она может быть считана из базы данных или из файла). В методе `iterate()` при помощи цикла `foreach` перебираются все записи из списка. Функция `call_user_func()` вызывает переданную в нее функцию для обработки данных из каждой записи. В этом примере в качестве такой функции выступает функция `find_min_salary()`, которая просматривает каждую запись и находит минимальную зарплату.

Улучшаем трюк

Я считаю, что версия этого шаблона, работающая с функциями, немного неуклюжая. Я бы лучше создал объект для посетителя, который получал бы доступ к каждой записи. В этом случае минимальное значение можно хранить в объекте и иметь возможность получать к нему доступ в дальнейшем. На рис. 7.13 показан вариант, в котором метод `iterate()` принимает в себя объект (типа `RecordVisitor`), а не функцию. Обновленный код вы можете видеть в примере 7.18.

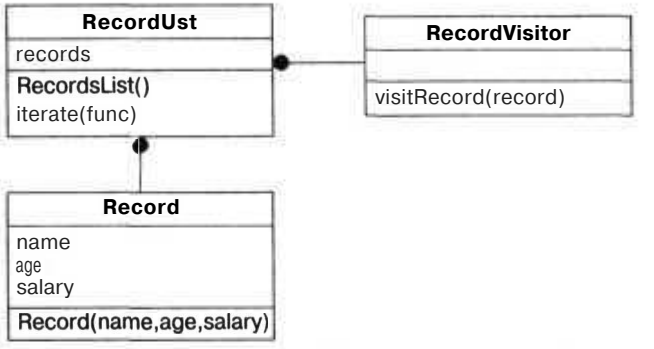


Рис. 7.13. Вариант, в котором Посетитель является объектом

Пример 7.18. Обновленная версия Посетителя

```
<?php
class Record
{
  public $name;
  public $age;
  public $salary;
  public function Record( $name. Sage. $salary )
  {
    $this->name - $name:
    $this->age - $age:
    $this->salary = $salary;
  }
}

abstract class RecordVisitor
{
  abstract function visitRecord( $rec ):
}

class RecordList
{
  private $records = array( );
  public function Recordl_ist( )
  {
    $this->records []= new Record( "Larry". 22. 35000 );
    $this->records [] - new Record( "Harry". 25. 37000 ):
    $this->records []= new Record( "Mary". 42. 65000 ):
    $this->records [] - new Record( "Sally", 45. 80000 ):
```

```
  }
  public function iterate( $vis )
  {
    foreach( $this->records as $r )
    {
      call_user_func( array( $vis, "visitRecord" ), $r );
    }
  }
}

class MinSalaryFinder extends RecordVisitor
{
  public $min = 1000000;
  public function visitRecord( $rec )
  {
    if( $rec->salary < $this->min ) { $this->min = $rec->salary; }
  }
}
$rl = new RecordList( );
$msl = new MinSalaryFinder( );
$rl->iterate( $msl );
echo( $msl->min."\n" );
?>
```

Я добавил абстрактный класс RecordVisitor и реализовал его в классе MinSalaryFinder, где и хранится минимальное значение. Теперь проверочный код сначала создает объект RecordList, затем объект MinSalaryFinder и передает его в качестве параметра в метод iterate(). И наконец, выводит на экран минимальное значение.

Прежде чем мы закончим с этим трюком, хочу высказать вам еще пару мыслей. Во-первых, PHP не силен в динамическом вызове функций. Передача функции по имени — не лучшая идея, к тому же потенциально ошибочная. В языках Python, Perl, Ruby, Java и C# (и практически во всех других языках) есть возможность создать переменную-указатель на функцию. При помощи этого указателя на функцию будет проще вызвать метод (в зависимости от языка можно использовать ссылки). Я люблю PHP, как и все остальные, но считаю, что эта оплошность должна быть исправлена в его следующих версиях.

ГЛАВА 8

Тестирование

Трюки 79–85

В последнее время очень активно обсуждается тестирование приложений, а также включение его в и без того очень напряженный процесс разработки. Эту главу я начну с рассмотрения возможности тестирования программы при помощи специальных компонентов для проверки работоспособности отдельных функций и классов приложения. Затем мы перейдем к автоматическому формированию компонентов для тестирования, а также к периодически изменяющимся средствам тестирования приложений. Вы создадите роботов, которые будут формировать HTTP-запросы к серверу, и тесты, которые при помощи средств автоматизации Internet Explorer проверяют веб-интерфейсы на ошибки.

Проверка кода при помощи компонентов для тестирования

Используйте PHPUnit и PHPUnit2 для непрерывного тестирования вашего кода.

PHP постепенно становится мощным средством для разработки корпоративных приложений. По мере того как сложность приложений возрастает, возрастают и потребности в тестировании, цель которого — убедиться, что приложение работает стабильно. Очевиден тот факт, что надежные приложения имеют многоярусную структуру, а каждый его уровень ранее был отдельно подвергнут пристальному тестированию. Например, вы можете иметь отдельный замкнутый уровень для работы с базой данных (прошедший комплексное тестирование), поверх которого расположена остальная бизнес-логика приложения (также прошедшая через тестирование), поверх которой, в свою очередь, расположен слой, содержащий пользовательский интерфейс (который также подвергся тестированию). Начинаете понимать, в чем идея?

Для проверки каждого из уровней обычной практикой считается использование компонентов для тестирования. Это тесты, которые запускаются после внесения каких-либо изменений в код и перед тем, как эти изменения были внесены в библиотеку (вы ведь используете управление версиями в ваших корпоративных проектах, не так ли?). На самом деле это нормально — настаивать на том, что код

должен пройти тестирование при помощи компонентов, прежде чем он даже просто сможет претендовать на внесение в централизованную базу данных проекта. В дальнейшем, когда будут выявлены новые ошибки, будут созданы новые компоненты для тестирования. Они будут выполнять ошибочный код и сообщать о возникшей ошибке, которая и будет исправлена, а затем для исправленного кода вновь запускаются созданные тесты, чтобы удостовериться, что проблемы больше нет. Благодаря такому подходу все будут знать о возникновении каких-то сложностей (из списка о выявленных ошибках).

В этом трюке будет показано, как использовать интегрированную среду PHPUnit2, доступную через сеть распределения модулей (см. трюк 2), для создания в PHP 5 компонентов, предназначенных для тестирования.

ПРИМЕЧАНИЕ

В случае с PHP 4 вы, наверное, захотите воспользоваться исходной версией PHPUnit. Этот модуль также доступен в PEAR.

Код

Сохраните исходный код из примера 8.1 в файл с именем Add.php.

Пример 8.1. Простая функция для сложения

```
<?pnp
function add( $a. $b ) { return $a + $b; }
?>
```

Исходный код из примера 8.2 нужно сохранить в файл TestAdd.php. В этом простом сценарии содержится два компонента для тестирования: testlO и test2(), оба предназначенные для проверки метода add().

Пример 8.2. Компоненты для проверки функции, выполняющей операцию сложения

```
<?pnp
require_once 'Add.php';
require_once 'PHPUnit2/Framework/TestCase.php';
class TestAdd extends PHPUnit2_Framework_TestCase
{
  function testK ) { $this->assertTrue( add( 1. 2 ) – 3 ); }
  function test2( ) { $this->assertFalse( add( 1. 1 ) == 3 ); }
}
?>
```

Это вполне нормально, что для проверки такой простой функции как, add(), потребовалось использование (как минимум) двух компонентов для тестирования. Представьте же, как много их потребуется для всестороннего тестирования обычного приложения!

Запуск трюка

Для исполнения этого кода воспользуйтесь приложением phpunit, предназначенным для запуска тестов:

```
% phpunit TestAdd.php
PHPUnit 2.2.1 by Sebastian Bergmann.
..
Time: 0.0028750896453857
OK (2 tests)
%
```

Здесь показан запуск компонентов, предназначенных для тестирования (отделенный двумя точками), а также его результат, который в нашем случае положительный, так как оба теста были пройдены успешно.

ПРИМЕЧАНИЕ

Удостоверьтесь, что в качестве параметра в phpunit вы передали имя сценария, содержащего компоненты для тестирования, а не имя проверяемого вами сценария.

Смотрите также

- «Формирование компонентов для тестирования» (см. трюк 80).

ТРЮК №80

Формирование компонентов для тестирования

Используйте PHP для создания компонентов, предназначенных для тестирования, на основе комментариев к коду.

Компоненты для тестирования (см. трюк 79) очень важны при разработке надежных приложений, поэтому стоит потратить некоторое время на их создание. Однако есть что-то среднее между написанием компонентов для тестирования полностью вручную (включая также те ситуации, когда эти тесты одинаковые) и автоматизированным созданием тестов. В этом трюке показано, как использовать сценарий, формирующий компоненты, предназначенные для тестирования, из комментариев к вашему коду на PHP. Эти комментарии имеют специально предназначенную для тестирования форму, но их использование поможет вам избежать создания избыточного кода.

Код

Сохраните исходный код из примера 8.3 в файл с именем Add. php. Имейте в виду, что при формировании некоторых тестов используются размещенные в комментариях к PHP-сценарию операторы == и !=.

Пример 8.3. Код, на базе которого формируются компоненты для тестирования

```
<?php
// UNIT_TEST_START
//(1.2) - 3
// (1.-1) == 0
```

```
// ( 1. 1 ) !- 3
// ( 1. -1 ) !- 1
// UNIT_TEST_END
function add( $a. $b ) { return $a + $b; }
// UNIT_TEST_START
// ( 1. 2 ) — -1
// ( 1. -1 ) — 2
// ( 1. 1 ) !- 1
// ( 1. -1 ) !- 1
// UNIT_TEST_END
function minus( $a, $b ) { return $a - $b: }
?>
```

Сохраните исходный код из примера 8.4 в файл с именем GenUnit. php. Этот сценарий отвечает за формирование тестов исходя из комментариев, размещенных в другом сценарии.

Пример 8.4. Генератор компонентов для тестирования

```
<?php
if ( count( $argv ) < 2 )
{
  print "GenUnit.php usage:\n":
  print " php GenUnit.php <PHP Script>\n";
  exit:
}
$infile - $argv[l];
define( 'STATE_NORMAL', 0 );
define( 'STATE_IN_UNIT_DEF', 1 );
define( 'STATE_WAITING_FOR_FUNC', 2 ):
$state - STATE_NORMAL;
$fh = fopen( $infile. "r" );
$tests - arrayt ):
$funcs - arrayt ):
while( $str = fgets( $fh ) )
{
  if ( $state -- STATE_NORMAL )
  {
    if ( preg_match( "|UNIT_TEST_START|", $str ) )
      $state - STATE_IN_UNIT_DEF:
  }
  else if ( $state == STATE_IN_UNIT_DEF )
  {
    if ( preg_match( "|UNIT_TEST_END|", $str ) )
      $state - STATE_WAITING_FOR_FUNC;
    else
    {
      $str = preg_replace( "|^//\s*|", "". $str );
      $str = preg_replace( "|\s*$|", "". $str ):
      $tests []= $str;
    }
  }
  else if ( $state — STATE_WAITING_FOR_FUNC )
  {
    if ( preg_match( "|function\s+(.*?)\(|". $str. $out ) )
    {
```

```
      $funcs []= array(
        'function' -> $out[l].
        'tests' => $tests
      );
      $State - STATE_NORMAL;
      $tests = array( ):
    }
  }
}
fclose( $fh );
ob_start( );
$outfile = "Test".$infile;
$classname = preg_replace( "|[.]php$|i". "". $outfile );
echo( "<?php\n" );
?>
//Этот код написан к файлу GenUnit.php
//
//Не изменяйте код вручную. ваши изиенения
//будут потеряны в следующий раз при выполнении GenUnit.php.
require_once '<?php echo( $infile ); ?>';
require_once 'PHPUnit2/Framework/TestCase.php';

class <?php echo( $classname ): ?> extends PHPUnit2_Framework_TestCase
{
  <?php
  $id = 1;
  foreach( $funcs as $func )
  {
    foreach( $func['tests'] as $test ) {
      ?>
      function test<?php echo($id); ?>( ) { $this->assertTrue( <?php echo(
        $func['function'].$test ) ?> ): }
      <?php
      $id+=1;
    }
  }
?>
}
<?php
echo( "?>\n" );
$test_php = ob_get_clean( );
print ($id-1)." tests created in $outfile\n";
$fh = fopen( $outfile, "w" );
fwrite( $fh, $test_php ):
fclose( $fh );
?>
```

Сценарий GenUnit.php для нас очень интересен. Его работа заключается в чтении PHP-сценариев. Специальная функция для этого расположена в начале файла. Затем сценарий использует реализованный в нем механизм для просмотра каждой строки из файла и поиска начала и конца специальных комментариев, предназначенных для тестирования: каждый из таких комментариев сценарий обрабатывает, после чего сохраняет в массиве $funcs сформированные им в соответствии с этими комментариями тесты.

ПРИМЕЧАНИЕ

Вы можете запросто заменить в файле **GenUnit.php** разделители **UNIT_TEST_START** и **UNIT_TEST_END** вашими собственными.

Во второй части сценария формируется исходный код тестов на PHP, для чего используется содержимое массива `$funcs`. Первым делом сценарий размещает в буфере PHP полученные от PHP выходные данные и создает тестовые классы и функции. Затем он прекращает буферизацию и сохраняет полученный код тестов на PHP в файл.

Запуск трюка

Для запуска сценария GenUnit.php воспользуйтесь PHP-интерпретатором для командной строки:

```
% php GenUnit.php Add.php
8 tests created in TestAdd.php
```

Сценарий считывает файл `Add.php` и создает `TestAdd.php`. Этот файл (после того как была произведена работа с файлом `Add.php`) выглядит следующим образом:

```
<?php
//Этот код написан к файлу GenUnit.php.
//
//Не изменяйте код вручную, ваши изменения
//будут потеряны в следующий раз при выполнении GenUnit.php.
require_once 'Add.php';
require_once 'PHPUnit2/Framework/TestCase.php';

class TestAdd extends PHPUnit2_Framework_TestCase
{
  function test1( ) { $this->assertTrue( add( 1. 2 ) - 3 ); }
  function test2( ) { $this->assertTrue( add( 1. -1 ) - 0 ): }
  function test3( ) { $this->assertTrue( add( 1. 1 ) != 3 ); }
  function test4( ) { $this->assertTrue( add( 1. -1 ) !- 1 ); }
  function test5( ) { $this->assertTrue( minus( 1. 2 ) - -1 ); }
  function test6( ) { $this->assertTrue( minust 1. -1 ) - 2 ): }
  function test7( ) { $this->assertTrue( minust 1. 1 ) !- 1 ): }
  function test8( ) { $this->assertTrue( minus( 1. -1 ) != 1 ); }
}
?>
```

Теперь вы можете видеть, что код, который был размещен в комментариях, используется в тестовых функциях. Кроме того, обратите внимание на комментарий в начале файла, который сообщает потенциальным разработчикам о том, что этот файл был сформирован автоматически (а не был написан вручную). Это позволяет донести до пользователей тот факт, что если они внесут в этот файл какие-либо изменения, то те будут потеряны после следующего запуска генератора.

Вы можете запустить сформированный генератором код, как и всякий другой, написанный на PHP:

```
% phpunit TestAdd.php
PHPUnit 2.2.1 by Sebastian Bergmann.
```

```
Time: 0.010335922241211
OK (8 tests)
```

Здесь сообщается о том, что были запущены восемь тестов и все они отработали правильно.

Смотрите также

- «Проверка кода при помощи компонентов для тестирования» (см. трюк 79).

ТРЮК №81

Проверка на наличие битых ссылок

Используйте буферизацию выходных данных для анализа текущей страницы и CURL для проверки ссылок в ней, чтобы удостовериться, что они указывают на существующие страницы.

Битые ссылки — это головная боль интернет-администраторов. Самое ужасное заключается в том, что ссылка, которая сегодня работает, на следующей неделе может быть уже битой из-за постоянно меняющейся природы Интернета. Чтобы помочь вам избавиться от этой досадной проблемы, сценарий в этом трюке получает в себя блоки HTML-кода и проверяет находящиеся в нем ссылки на работоспособность. Затем он предоставляет подробный отчет, сообщая о битых ссылках, что позволяет вам с легкостью отыскать их и избавиться от этой проблемы.

Код

Сохраните исходный код из примера 8.5 в файл с именем index.php. Эта страница предоставляет вам одну рабочую и одну нерабочую ссылку.

Пример 8.5. Страница-источник для программы, проверяющей ссылки

```
<?php
require_once( "checklinks.php" ); ?>
<html>
<body>
<?php checklinks_start( ) ?>
<div style="width: 800px" />
<a href="http://www.cnn.com">CNN</a><br/>
<a href-"http://badlink">Bad link</a><br/>
<?php checklinks_end( ) ?>
</div>
</body>
</html>
```

Сохраните сценарий из примера 8.6 в файл checklinks.php.

Пример 8.6. Код для проверки работоспособности ссылок

```
<?php
function checklinks_start( )
```

```
{
  ob_start( );
}

function checklinks_end( )
{
  $doc = ob_get_clean( );
  preg_match_all( "/\<a.*?href=[\"|\']([.*?)[\"|\']\>/", $doc, $found );
  print( $doc ):
  $badlinks = array( );
  foreach( $found[1] as Slink )
  {
    $ch - curl_init( Slink );
    ob_start( ):
    curl_exec( $ch ):
    Sout - ob_get_clean( ):
    if ( curl_errno( Sch ) !- 0 )
      $badlinks [ ] - Slink;
    curl_close( Sch ):
  }
  if ( count( Sbadlinks ) > 0 ) {
  ?>
<br/>
<table style="background: red:" cellspacing="2" cellpadding="2" width="100%">
<tr><td style="white; color: white: text-align:center;">Bad links</td></tr>
<tr><td style="background: white:">
<?php foreach( Sbadlinks as Slink ) { echo( $link."<br/>" ): } ?>
</td></tr>
</table>
<?php } } ?>
```

Запуск трюка

Загрузите оба сценария на сервер и откройте в браузере страницу index.php. Если у вас есть доступ в Интернет, то в окне браузера вы увидите страницу, похожую на изображенную на рис. 8.1.

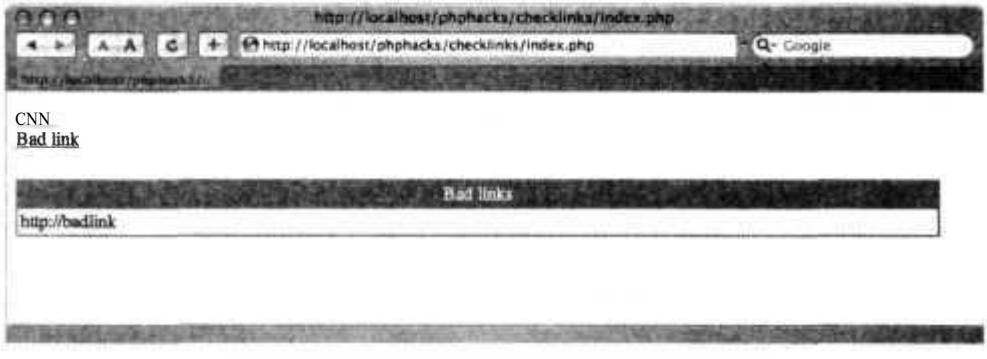


Рис. 8.1. Страница для проверки ссылок с отчетом

Та часть HTML, которую следует проверить, размещена между вызовами функций checklinks_start() и checklinks_end(). Эти методы, в свою очередь, используют функцию ob_start() и ob_get_clean(), чтобы разместить полученные от PHP данные в буфере. Затем функция checklinks_end() использует функции CURL для запроса страниц, размещенных в якорных тегах (<a>). Пока компьютер подключен к Интернету, ссылка на домашнюю страницу CNN должна быть в порядке, но ссылка http://badlink всегда будет нерабочей. В результате функция checklinks_end() выводит на экран отчет, сообщающий о битых ссылках.

Смотрите также

- «Проверка приложения при помощи смоделированных пользователей» (смотрите трюк 82).
- «Проверка приложения при помощи роботов» (смотрите трюк 83).
- «Следите за вашим сайтом» (смотрите трюк 84).

ТРЮК №82

Проверка приложения при помощи смоделированных пользователей

Используйте интерфейс автоматизации в Internet Explorer для тестирования вашего приложения через UI.

Вы можете проверить внутренний интерфейс вашего приложения при помощи компонентов для тестирования, в частности кода, работающего с базами данных, и кода, реализующего бизнес-логику. Вы также можете слегка проверить внешний интерфейс вашего приложения при помощи роботов (см. трюк 83), которые будут запрашивать страницы с сервера и передавать в них данные. Но как же вы можете проверить то, что на самом деле будет делать пользователь? Как вы можете имитировать нажатие кнопок и добавление данных в поля ввода, как это делает обычный пользователь?

В случае с особенно большими приложениями, использующими JavaScript, вам понадобится что-то, что будет нажимать кнопки, чтобы удостовериться, что ваше приложение делает то, что и должно делать. В Windows вы можете воспользоваться объектами Explorer COM, чтобы заставить браузер перейти на ваш сайт и даже нажать соответствующие кнопки. На рис. 8.2 проиллюстрировано взаимодействие между различными PHP-файлами, используемыми в этом трюке. test.php и print.php — обычные веб-страницы, написанные на PHP. Сценарий testagent.php запускается из командной строки и указывает браузеру, что ему следует открыть страницы, заполнить формы и проверить результат.

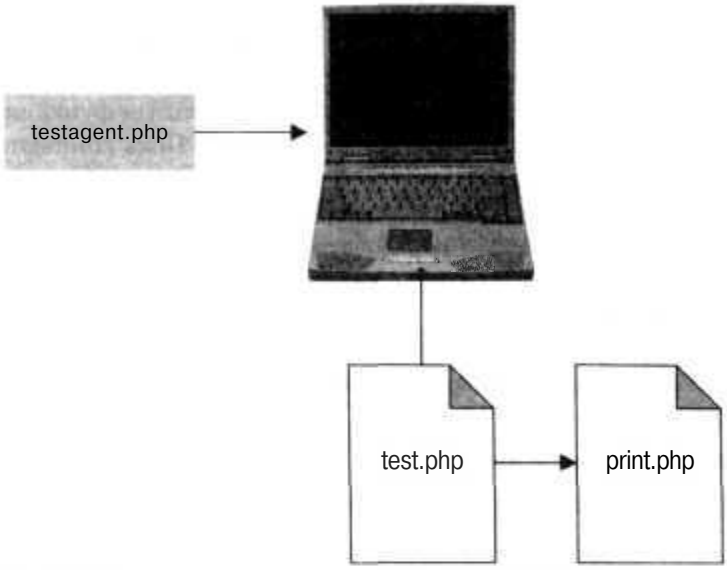


Рис. 8.2. Программа-агент для тестирования, управляющая браузером

Код

Сохраните исходный код из примера 8.7 в файл с именем `test.php`. Эта страница предназначена для проверки программы-агента

Пример 8.7. Первая страница с кодом для тестирования

```
<html>
<head>
<title>Automated test agent test page</title>
<head>
<body>
<form id="inp_form" action="print.php">
<table cellpadding="2"><tr>
<td>First:</td>
<td><input id="first_name" name="first" type="text"></td>
</tr><tr>
<td>Last:</td>
<td><input id="last_name" name="last" type="text"></td>
</tr>
<tr>
<td colspan="2" align="center">
<input type="submit" />
</td>
</tr>
</table>
</form>
</body>
</html>
```

Сохраните код из примера 8.8 в файл с именем `print.php`. Она служит в качестве формы для приема данных из примера 8.7.

Пример 8.8. Вторая страница с кодом для проверки

```
<html>
<head>
<title>Automated test agent print page</title>
</head>
<body>
You entered:<br/>
First:
<span id="res_first"><?php echo( $_GET['first'] ); ?></span>
<br/>
Last:
<span id="res_last"><?php echo( $_GET['last'] ); ?></span>
<br/>
</body>
</html>
```

Исходный код из примера 8.9 нужно сохранить в файл с именем `testagent.php`.

Пример 8.9. Сценарий, использующий средства автоматизации Internet Explorer

```
<?php
function delay( $ie. $amount )
{
  fort $c - 0: $c < ( $amount / 100 ); $c++ )
    com_message_pump( 100 ):
}

function test_page( $ie. $page. Sfirst. $last )
{
  $ie->Navigate( $page ):
  delay( $ie. 2000 ):
  $fn - $ie->Document->getElementById( "first_name" ):
  $fn->Value - Sfirst:
  $ln = $ie->Document->getElementById( "last_name" ):
  $ln->Value = $last:
  $inf - $ie->Document->getElementById( "inp_form" ):
  $inf->submit( ):
  delay( $ie. 2000 ):
  $rfn = $ie->Document->getElementById( "res_first" ):
  $rfn = $rfn->innerHTML;
  $rln = $ie->Document->getElementById( "res_last" ):
  $rln = $rln->innerHTML;
  if( strcmp( $rfn. Sfirst ) == 0 &&
  strcmp( $rln. $last ) — 0 )
  {
    print "Test passed.\n":
    return 0;
  }
  else
  {
    print "Test failed.\n";
    return -1:
  }
}
$ie - new COM("InternetExplorer.Application");
$ie->Visible - true:
$result - test_page( $ie.
"http://localhost:1222/com/test.php".
```

```
"Charles".
"Herrington" );
$ie->Quit( ):
exit( $result );
?>
```

Этот сценарий напрямую использует интерфейс Internet Explorer COM при помощи встроенных оберточных классов COM. Обработка начинается внизу, где код создает COM-объект типа `InternetExplorer.Application`. Затем сценарий выполняет функцию test_page(), чтобы использовать Internet Explorer для тестирования вашей страницы. Функция testpageO открывает в браузере URL-адрес, затем вводит данные в форму и передает их дальше. После этого функция немного ждет, чтобы позволить загрузиться полученной в результате странице. Далее она проверяет содержимое тегов в возвращенной странице, чтобы удостовериться, что там содержится нужная информация. После этого сценарий выходит из Internet Explorer и присваивает коду выхода значение 0 или -1 в зависимости от результата теста. И, наконец, выводится небольшое сообщение о том, как прошло тестирование.

ПРИМЕЧАНИЕ

Очевидно, что эти тесты очень индивидуальны и подходят только к определенному сайту и даже только к конкретной странице. Вы не сможете автоматически формировать их, но, тем не менее, тесты такого рода, имитирующие работу пользователя, очень важны.

Запуск трюка

Загрузите файлы `test.php` и `print.php` на ваш веб-сервер. Исправьте файл `testagent.php` так, чтобы он использовал конкретный URL-адрес вашей страницы. Затем воспользуйтесь следующей командой:

```
c:\testagent\> php testagent.php
Test passed.
```

Во время работы теста сначала вы увидите страницу для авторизации (рис. 8.3).

Рис. 8.3. Тестовая страница для ввода данных

Затем программа-агент, имитирующая пользователя, заполнит форму и нажмет кнопку Submit Query. Таким образом, оно перейдет к следующей странице (рис. 8.4), которая просто повторно отображает введенные данные.

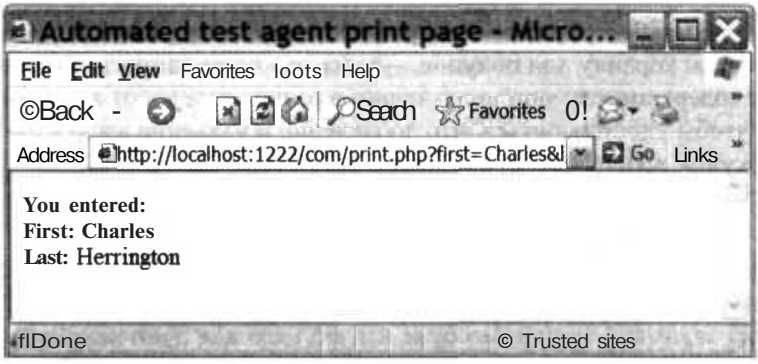


Рис. 8.4. Результат работы тестовой страницы

В файле `print.php` расположен тег , содержащий введенные ранее имя и фамилию. Программа-агент, имитирующая работу пользователя, использует свойство `innerHTML` этого тега , чтобы удостовериться, что значения, которые страница вернула в ответ, совпадают с введенными в форму.

ПРИМЕЧАНИЕ

Это как раз то, чем и занимаются такие коммерческие продукты, как SilkTest (http://segue.com/) и **TestComplete** (http://automatedqa.com/), предназначенные для тестирования веб-страниц. Эти продукты стоят больших денег, в то время как предложенный код бесплатный.

Смотрите также

- «Проверка приложения при помощи роботов» (см. трюк 83).
- «Следите за вашим сайтом» (см. трюк 84).

ТРЮК №83

Проверка приложения при помощи роботов

Используйте модуль PEAR HTTP_Client для проверки вашего PHP-приложения через Интернет.

Откуда вы знаете, как работает ваше приложение? Это как в случае со светом в вашем холодильнике: если вы не видите лампочку, откуда вы знаете, что она выключается? Один из способов постоянно следить за вашим приложением — создать робота, который будет тестировать ваш сайт. Вы можете периодически

запускать этого робота, чтобы удостоверится, что сервер корректно отвечает на запросы (и получать сообщения, когда это не так).

В этом трюке показано, как воспользоваться модулем PEAR HTTP_Client (см. трюк 2) для проверки приложения, использующего корзину для покупок (см. трюк 66). На рис. 8.5 проиллюстрирован сценарий robot.php, работающий с приложением, использующим корзину для покупок, — и все это через запросы к серверу. Робот проверяет содержимое каждой полученной в результате работы страницы в приложении, чтобы удостовериться, что добавление и удаление элементов из корзины для покупок происходит правильно.

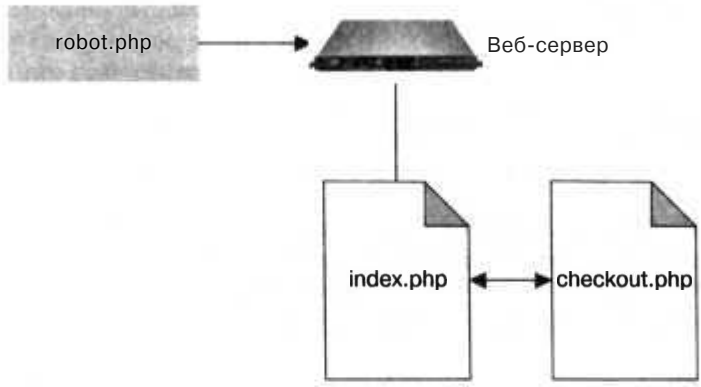


Рис. 8.5. Робот, проверяющий работу корзины для покупок через запросы к веб-серверу

Код

Сохраните исходный код из примера 8.10 в файл robot.php.

Пример 8.10. Робот для тестирования

```
<?php
require_once 'HTTP/Client.php';
function check_html( $testname, $client, $values )
{
  $resp = $client->currentResponse( );
  $body = $resp['body'];
  preg_match( "/\<\!\-\- CART \: (.*?) \-\-\>/", $body, $found );
  print "$testname: ";
  print ( $found[1] == join(",", $values ) ) ? "passed" : "failed";
  print "\n";
}
$client = new HTTP_Client( );
$client->get( "http://localhost/phphacks/shopcart/index.php" );
$client->post( "http://localhost/phphacks/shopcart/add.php", array( 'prod_id' =>
1 ) );
$client->get( "http://localhost/phphacks/shopcart/index.php" );
check_html( "Add one", $client, array( 1 ) );
$client->post( "http://localhost/phphacks/shopcart/add.php", array( 'prod_id' =>
```

```
2 ) );
$client->get( "http://localhost/phphacks/shopcart/index.php" );
check_html( "Add two", $client, array( 1, 2 ) );
$client->post( "http://localhost/phphacks/shopcart/add.php", array( 'prod_id' =>
3 ) );
$client->get( "http://localhost/phphacks/shopcart/index.php" );
check_html( "Add three", $client, array( 1, 2, 3 ) );
$client->get( "http://localhost/phphacks/shopcart/checkout.php" );
check_html( "Checkout", $client, array( 1, 2, 3 ) );
$client->post( "http://localhost/phphacks/shopcart/delete.php", array( 'ids[]' ->
2 ) );
$client->get( "http://localhost/phphacks/shopcart/checkout.php" );
check_html( "Remove two", $client, array( 1, 3 ) );
?>
```

Для начала код создает новый объект типа HTTP_Client PEAR. Затем он выполняет несколько запросов GET и POST, чтобы имитировать отправку транзакций от пользователя в систему, как если бы эти запросы шли от браузера. Но на самом деле его нет. Функция checkjitml используется для проверки возвращаемых страниц, чтобы изучить содержимое корзины для покупок в данный момент. Содержимое корзины находится в отдельном HTML-коде, ограниченном специальными комментариями, причем этот код находится и в странице index.php, и в странице checkout.php. В этих комментариях содержится список ID элементов, находящихся в корзине и разделенных запятыми. После отсылки ID продукта в страницу add.php при помощи запроса POST робот ожидает, что этот же ID вернется в страницу index.php и т. д. Если робот не получает корзину с требуемыми ему товарами, то он сообщит об ошибке при прохождении теста.

Запуск трюка

Для запуска этого трюка сперва необходимо установить на сервер и запустить приложение, использующее корзину для покупок (см. трюк 66). После этого вы можете проверить ее работу при помощи робота. Запустите его, используя PHP-интерпретатор для командной строки:

```
% php robot.php
Add one: passed
Add two: passed
Add three: passed
Checkout: passed
Remove two: passed
```

Здесь показано, что после каждого предпринятого роботом действия он получил тот ответ, который и ожидал. Кроме того, стоит отметить, что страницы, реализующие работу корзины для покупок, написаны таким образом, чтобы робот мог их проверить: в каждую страницу встроен комментарий, невидимый для конечного пользователя, но используемый роботом. Он считывает эти комментарии, содержащие текущую информацию о состоянии корзины, и сравнивает ее с запрограммированными значениями.

Улучшаем трюк

Роботы могут быть использованы и в качестве компонентов для тестирования (см. трюк 79). Для начала установите предназначенный для тестирования модуль PHPUnit2, доступный через PEAR (см. трюк 2). Затем воспользуйтесь кодом из примера 8.11 для запуска тестирования при помощи компонентов.

Пример 8.11. Использующая PHPUnit версия кода, реализующего робота

```
<?php
require_once 'HTTP/Client.php';
require_once 'PHPUnit2/Framework/TestCase.php';
class RobotUnit extends PHPUnit2_Framework_TestCase
{
  var $client = null;
  private function check_html( $testname, $values )
  {
    $resp = $this->client->currentResponse( );
    $body = $resp['body'];
    preg_match( "Л<\!\-\- CART \: (.*?) \-\-\>/", $body, $found );
    return ( $found[l] == join(".", $values ) );
  }
  function test1( )
  {
    $this->client = new HTTP_Client( );
    $this->client->get( "http://localhost/phphacks/shopcart/index.php" );
    $this->client->post( "http://localhost/phphacks/shopcart/add.php",
    array( 'prod_id' => 1 ) );
    $this->client->get( "http://localhost/phphacks/shopcart/index.php" );
    $this->assertTrue( $this->check_html( "Add one", array( 1 ) ) );
  }
  function test2( )
  {
    $this->client = new HTTP_Client( ):
    $this->client->get( "http://localhost/phphacks/shopcart/index.php" );
    $this->client->post( "http://localhost/phphacks/shopcart/add.php",
    array( 'prodjd' => 1 ) ):
    $this->client->get( "http://localhost/phphacks/shopcart/index.php" );
    $this->assertTrueC $this->check_html( "Add one", array( 1 ) ) );
    $this->client->post( "http://localhost/phphacks/shopcart/add.php",
    array( 'prodjd' => 2 ) );
    $this->client->get( "http://localhost/phphacks/shopcart/index.php" );
    $this->assertTrue( $this->check_html( "Add two", array( 1, 2 ) ) );
  }
  function test3( )
  {
    $this->client - new HTTP_Client( );
    $this->client->get( "http://localhost/phphacks/shopcart/index.php" ):
    $this->client->post( "http://localhost/phphacks/shopcart/add.php",
    array( 'prodjd' => 1 ) ):
    $this->client->get( "http://localhost/phphacks/shopcart/index.php" );
    $this->assertTrue( $this->check_html( "Add one", array( 1 ) ) ):
    $this->client->post( "http://localhost/phphacks/shopcart/add.php",
    array( 'prodjd' => 2 ) ):
    $this->client->get( "http://localhost/phphacks/shopcart/index.php" );
    $this->assertTrue( $this->check_html( "Add two", arrayC 1, 2 ) ) ):
```

```
        $this->client->post( "http://localhost/phphacks/shopcart/add.php",
        array( 'prod_id' -> 3 ) );
        $this->client->get( "http://localhost/phphacks/shopcart/index.php" );
        $this->assertTrue( $this->check_html( "Add three", array( 1, 2, 3 ) ) );
        $this->client->get( "http://localhost/phphacks/shopcart/checkout.php" );
        $this->assertTrue( $this->check_html( "Checkout", array( 1 , 2 , 3 ) ) );
        $this->client->post( "http://localhost/phphacks/shopcart/delete.php",
        array( 'ids[]' -> 2 ) );
        $this->client->get( "http://localhost/phphacks/shopcart/checkout.php" );
        $this->assertTrue( $this->check_html( "Remove two", array( 1, 3 ) ) );
    }
}
?>
```

В новой версии кода тестирование при помощи роботов реализовано в виде методов класса PHPUnit. Тесты работают так же, как и раньше, причем с каждым разом запускаются новые, пока при помощи третьего метода тест не будет запущен полностью. Это повторение необходимо для проверки каждой рабочей секции в отдельности. Например, если первый и второй тесты будут пройдены, а на третьем возникнет ошибка, то вы будете знать, что необходимо что-то исправить в файле checkout.php, так как именно он проверялся во время третьего теста.

Запустите созданные тесты при помощи специально предназначенного для этого приложения phpunit:

```
% phpunit RobotUnit.php
PHPUnit 2.2.1 by Sebastian Bergmann.

...
Time: 1.5706360340118
OK (3 tests)
```

В отчете сказано, что было запущено три теста и они все отработали нормально. Каждый из этих тестов (test1, test2 и test3) запускает робота, увеличивая количество отсылаемых запросов. Тестирование при помощи компонентов такого рода идеально подходит для комплексной проверки работы системы (а не небольшой части приложения). Когда система запущена, то есть загружены базы данных и страницы, вы можете запустить этот тест, чтобы удостовериться, что все страницы правильно отвечают на запросы, прежде чем вводить систему в строй.

Смотрите также

- «Создание корзины» (см. трюк 66).

ТРЮК №84

Следите за вашим сайтом

Используйте модуль PEAR HTTP_Client для создания программы слежения, которая будет посещать все страницы на вашем сайте.

В этом трюке показано, как создавать на PHP программы слежения для проверки страниц на вашем сайте. Она идеально походит для использования в тестовых

целях, и с ее помощью можно легко удостовериться, что после обновления все ваши PHP и HTML-страницы по-прежнему правильно отвечают на запросы.

Код

Сохраните исходный код из примера 8.12 в файл spider.php.

Пример 8.12. Простая программа слежения

```
<?php
require_once 'HTTP/Client.php';
require_once 'HTTP/Request/Listener.php';
$baseurl = "http://localhost/phphacks/spider/test/index.html";
$pages - array( ):
add_urls( $baseurl );
while( ( $page = next_page( ) ) !- null )
{
  add_urls( $page );
}
function next_page( )
{
  global Spages;
  foreach( array_keys( Spages ) as $page )
  {
    if ( Spages[ Spage ] == null )
    return Spage:
  }
  return null;
}

function add_urls( Spage )
{
  global Spages:
         Sstart - microtime( ):
         $urls - get_urls( Spage );
         $resptime - microtime( ) - Sstart;
  print "Spage.. An";
  $pages[ Spage ] - array( 'resptime' -> floor( Sresptime * 1000 ). 'url' =>
  Spage ):
  foreach( Surls as $url )
  {
    if ( !array_key_exists( Surl. Spages ) )
      Spages[ Surl ] - null;
  }
}

function get_urls( Spage )
{
  Sbase = preg_replace( "/\/([^\/]*?)$/", "/". Spage );
  $client = new HTTP_Client( );
  $client->get( Spage );
  $resp = $client->currentResponse( ):
  Sbody = Sresp['body'];
  $out - array( ):
  preg_match_all( "/(\<a.*?\>)/is". Sbody. $matches );
```

```
  foreach( $matches[0] as $match )
  {
    preg_match( "/href=(.*?)[\s|\>]/i", $match, $href ):
    if ( $href !- null )
    {
      $href - $href[1]:
      $href = preg_replace( "/^\"/", "", $href ):
      $href = preg_replace( "/\"$/", "", $href );
      if ( preg_match( "/^mailto:/", $href ) )
      {
      }
      elseif ( preg_match( "/^http:\/\//", $href ) )
      {
        if ( preg_match( '/^$base/', $href ) )
        $out []- $href:
      }
      else
      {
        $out []= $base.$href;
      }
    }
  }
  return $out;
}
ob_start( );
?>
<html>
<head>
<title>Spider report</title>
</head>
<body>
<table width="600">
<tr>
<th>URL</th>
<th>Response Time (ms)</th>
</tr>
<?php foreach( array_values( $pages ) as $page ) { ?>
<tr>
<td><?php echo( $page['url' ] ); ?></td>
<td><?php echo( $page['resptime' ] ); ?></td>
</tr>
<?php } ?>
</table>
</body>
</html>
<?php
$html = ob_get_clean( );
$fh = fopen( "report.html", "w" ):
fwrite( $fh, $html );
fclose( $fh );
?>
```

Код программы слежения начинает работу с одного URL-адреса и вызывает функцию addurl (), передавая этот URL в нее. Затем функция add_url () получает запрошенную страницу и извлекает из нее все ссылки. Они добавляются в глобальный

массив $pages. Далее сценарий перебирает все страницы из этого массива при помощи функции next_page(). После того как они все были отслежены, вторая часть сценария выводит на экран результат запроса к каждой странице сайта. Остальные примеры в этом трюке служат в качестве тестовых страниц для программы слежения. Сохраните первую из них, расположенную в примере 8.13, в файл с именем index.html.

Пример 8.13. Пример начальной страницы

```
<html><body>
  <a href="test1.html">Test 1</a><br/>
  <a href="test2.html">Test 2</a><br/>
  <a href="test3.html">Test 3</a><br/>
</body></html>
```

Исходный код из примера 8.14 нужно сохранить в файл test1.html.

Пример 8.14. Пример второй страницы

```
<html><body>
  <a href="http://www.cnn.com">CNN</a>
</body></html>
```

Сохраните исходный код из примера 8.15 в файл test2.html.

Пример 8.15. Пример третьей страницы

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

Исходный код из примера 8.16 сохраните в файл test3.html.

Пример 8.16. Пример четвертой страницы

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

Запуск трюка

Сохраните тестовые файлы (index.html, test1.html, test2.html и test3.html) в поддиректории для тестов. Затем запустите программу-шпиона при помощи PHP-интерпретатора для командной строки:

```
% php spider.php
http://localhost/phphacks/spider/test/index.html...
http://localhost/phphacks/spider/test/test1.html...
http://localhost/phphacks/spider/test/test2.html...
http://localhost/phphacks/spider/test/test3.html...
```

На экран будут выведены результаты работы программы с каждым из URL, который был просмотрен во время отслеживания сайта. В дополнение ко всему она также создаст отчет в HTML-формате, в котором будет сообщаться, какие страницы были отслежены (рис. 8.6).

В этом отчете указывается время в миллисекундах, затраченное на получение каждой страницы. Сюда входит время, необходимое на пересылку по сети, и время, потраченное сервером на формирование страницы. Поскольку в нашем случае HTML-страницы статичные, веб-сервер и программа-шпион запускаются локаль-

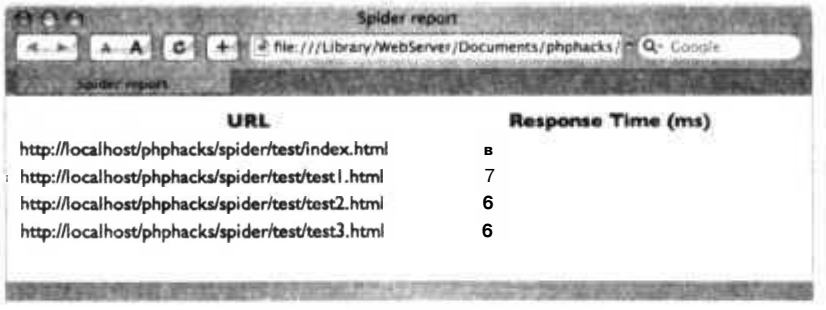


Рис. 8.6. Отчет программы-шпиона

но, а время, потраченное на получение каждой страницы, минимальное, то считается, что любой промежуток времени менее 200 — это довольно быстрый ответ.

ПРИМЕЧАНИЕ

Данный сценарий не обнаружит обособленные страницы, поэтому вам следует следить за этим. Если пользователь сделает закладки на отдельные страницы, которые не связаны с вашим сайтом, это может привести к появлению нерабочих страниц, которые этот сценарий не увидит и не внесет в отчет.

Смотрите также

- «Проверка приложения при помощи смоделированных пользователей» (смотрите трюк 82).
- «Проверка приложения при помощи роботов» (смотрите трюк 83).

ТРЮК №85

Автоматическое создание документации

Используйте комментарии PHPDoc, чтобы документировать ваш код, а также phpDocumentor для создания документации на базе комментариев к коду.

JavaDoc — это стандарт, используемый при создании комментариев к Java, он используется для автоматического формирования документации к классам, написанным на Java. Такого рода идея — формировать документацию на базе комментариев — была настолько популярна, что теперь практически у каждого языка программирования есть свой стандарт размещения комментариев, которые могут быть использованы для автоматического формирования документации. В PHP это PHPDoc; он позволяет упростить процесс написания документации к классам. phpDocumentor (http://www.phpdoc.org/) — это утилита с открытым кодом, которая анализирует код на PHP, извлекает из него документацию в формате PHPDoc и генерирует HTML на базе исходного кода, причем может применять различные стили. На рис. 8.7 показано, как PHPDoc берет PHP-файлы в качестве входной информации, в нашем случае это файл `Author.php`, и создает несколько HTML-файлов для пакета с документацией.

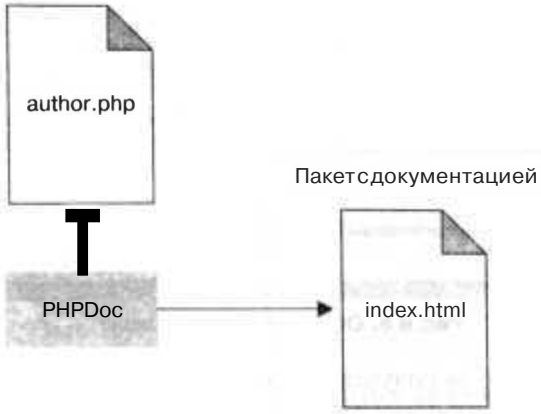


Рис. 8.7. Принцип работы PHPDoc

Код

Сохраните исходный код из примера 8.17 в файл Author.php.

Пример 8.17. Класс Author с документацией в формате PHPDoc

```
<?php
/**
* An author class
*/
class Author
{
  /**
  * Gets the name of the author
  */
  function getName( ) { }
  /**
  * Sets the name of the author
  * @param string $name The name of the author
  */
  function setName($name) { }
}
?>
```

Запуск трюка

В данном случае следует выполнить команду phpDocumentor применительно к PHP-файлам в вашем проекте. Файл Author.php — это пример класса с разметкой комментариев в формате PHPDoc: они все начинаются с символов /** и заканчиваются символами */. Текст, размещенный в этих специальных комментариях, затем вносится в документацию PHPDoc. Пометка @param сообщает генератору документации, что после нее следует тип параметра, затем имя параметра, а потом опи-

сание параметра. Чтобы получить полный список такого рода пометок, вам следует изучить документацию к phpDocumentor (http://www.phpdoc.org/). Здесь я перечислю только наиболее важные из них.

- @author — указывает имя автора.
- @copyright — определяет авторские права на исходный код.
- @license — текст лицензии к исходному коду.
- @see — перекрестные ссылки между этим классом или методом класса и другим классом или его методом. Следующий за этой пометкой текст — это текстовый идентификатор класса или метода, на который дается ссылка.
- @param — указывает на параметры функции.
- @return — определяет возвращаемые функцией значения.
- @todo — указывает на то, что осталось сделать в этом участке кода.

Запустите phpDocumentor следующим образом:

```
phpdoc -t doc -f *.php
```

ПРИМЕЧАНИЕ

Сгенерированные HTML-файлы будут размещены в директории doc.

Выполните эту команду, и вы увидите, как будет генерироваться документация. Щелкните дважды кнопкой мыши на странице docs/index.html в вашем браузере (рис. 8.8).

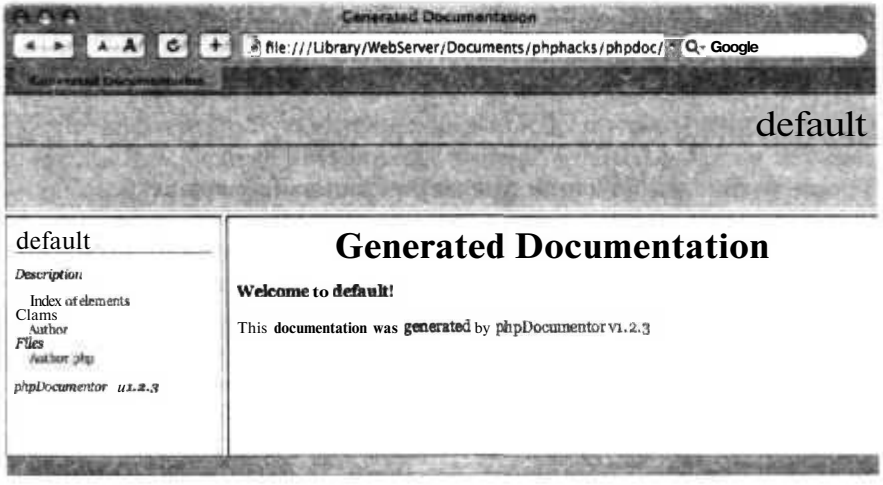


Рис. 8.8. Домашняя страница сгенерированной документации

Используя панель навигации в левой части окна, выберите ссылку на класс Author, и вы сможете выяснить о нем все, что вам требуется, или, по крайней мере, ровно столько, сколько комментариев для него создали программисты (рис. 8.9).

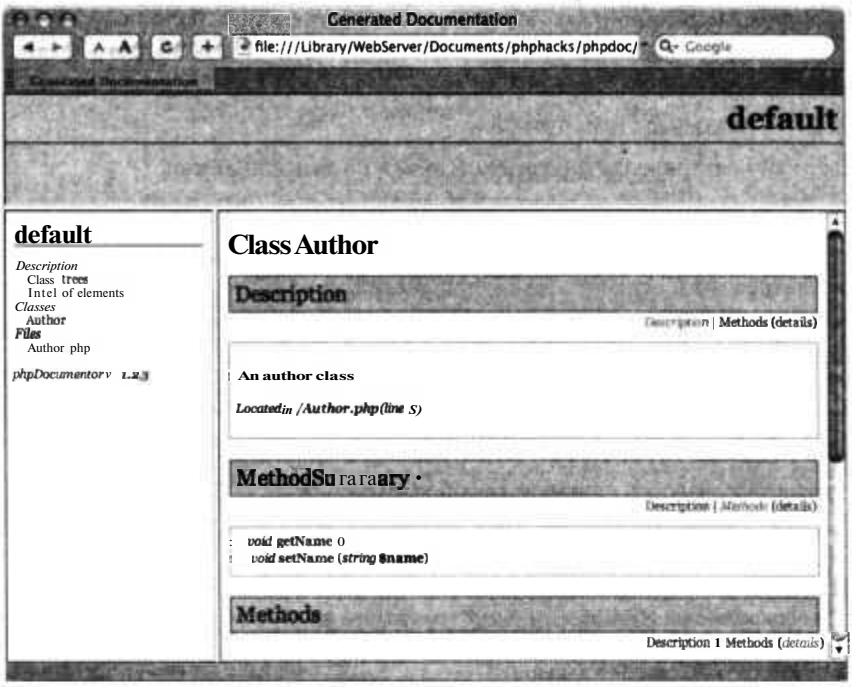


Рис. 8.9. Документация к классу Author

Я использовал несколько конструкций PHPDoc для создания комментариев к классу в файле `Author.php`. Можно воспользоваться еще многими ключевыми словами из PHPDoc при формировании документации, которые позволяют создавать перекрестные ссылки, гиперссылки и многое другое.

Смотрите также

- «Формирование компонентов для тестирования» (см. трюк 80).

Альтернативные пользовательские интерфейсы

Трюки 86-94

HTML и Динамический HTML (DHTML) не являются единственными технологиями, которыми вы можете воспользоваться для создания пользовательских интерфейсов в PHP. В этой главе вашему вниманию буду представлены несколько трюков, в которых на PHP будут созданы пользовательские интерфейсы для Рабочего стола Linux, Macintosh, Windows или даже портативной приставки PlayStation (PSP). Вы также узнаете, как общаться с вашими приложениями при помощи мгновенных сообщений.

ТРЮК №86

Создание пользовательских карт при помощи MapServer

Используйте MapServer и PHP для создания динамических карт в вашем интернет-приложении.

В последнее время огромную популярность приобрели цифровые карты. Отчасти на это повлиял свободный доступ к средствам с открытым кодом для создания карт, а также к геопространственным данным. Появились такие мощные приложения, как Google Maps и MapQuest. Это одни из самых популярных приложений, использующих технологию для создания карт. Получив доступ к данным, библиотекам для создания карт и PHP-сценариев, практически любой может создавать обычные или интерактивные карты. Доступно несколько инструментов с открытым кодом для создания карт, начиная с приложений для Рабоч его стола и заканчивая доступными через Интернет сервисами. Одним из такого рода инструментов является MapServer университета Миннесоты (http://ms.gis.umn.edu/). Обладая большой базой пользователей, энергичным сообществом и квалифицированными разработчиками, его инструменты являются довольно мощными для размещения карт в Интернете. MapServer активно используется на многих интернет-страницах, написанных на PHP. Примерами могут служить продукты Chameleon (http://maptools.org/) и Mapbender (http://www.mapbender.org/index.php/Main_Page) — оба активно использующие PHP. Кроме того, доступна мощная реализация средств для работы с картами на базе технологии Ajax под названием ka-Map.

ПРИМЕЧАНИЕ

Чтобы ознакомиться с учебным пособием по работе с ka-Map, посетите страницу http://www.xml.com/pub/a/2005/08/10/ka-map.html.

Все эти инструменты позволяют программистам на PHP работать с картами и картографическими данными. MapServer выступает в качестве основного рабочего механизма для создания изображений и взаимодействия с элементами управления на PHP, а также запросов к посредникам. MapServer в основном используется для организации взаимодействия CGI приложений с интернет-сервером. Чтобы ощутить максимальную мощь и гибкость данного инструмента, вы можете воспользоваться MapServer API для определенного языка программирования (поддерживаются такие языки как PHP, Perl, Python, Ruby, Java и C#). MapScript предоставляет вам методы для взаимодействия с картографическими данными и оформлением полученных карт, а также для создания конечных изображений с картами.

Общее представление о MapServer

Основные настройки MapServer производятся при помощи текстового файла, в котором хранится конфигурация. Этот файл является ядром большинства приложений, использующих MapServer. CGI-программы или обычные приложения MapScript считывают настройки из этого файла, получают доступ к данным, отрисовывают карту и возвращают в качестве результата изображение, готовое к использованию в сети. Полученные в итоге карта и рисунок могут даже быть запущены автономно в качестве программы для командной строки.

Примеры в этом трюке используют PHP MapScript для командной строки. Вы же можете изменить их так, чтобы они подходили вам исходя из среды, в которой вы хотите их использовать. Файл с картой обладает простой иерархической объектной структурой с возможностью настроек наследования от родительского объекта. В данном трюке у нас есть простейший файл с картой, который предполагается использовать с PHP MapScript. Вы также узнаете, как создавать карты, не имея в своем распоряжении файла с картой, разрабатывать данные для настроек полностью в памяти и отрисовывать карту при помощи PHP-сценария.

Установка расширения MapScript для PHP

Прежде чем вы сможете использовать PHP MapScript, вам придется загрузить и установить инструменты MapScript. На домашнем сайте PHP MapScript http://maptools.org/php_mapscript/ есть инструкции по загрузке, а также документация для API. Есть также несколько других простых способов получить исполняемый дистрибутив PHP и MapScript, уже сконфигурированных для совместной работы.

- Для Windows вы можете использовать дистрибутив MapServer for Windows (MS4W), доступный по адресу http://maptools.org/ms4w/. Он поставляется в виде обычного файла с архивом в формате ZIP, в котором содержится все необходимое для начала работы с MapServer, а также с PHP MapScript.

- Для Linux вы можете использовать дистрибутив FGS Linux, размещенный по адресу http://maptools.org/fgs/. В него включен консольный сценарий для установки, позволяющий автоматизировать ее процесс. При запуске FGS использует отдельные библиотеки, приложения и даже веб-сервер, что позволяет вам приступить к работе без необходимости компилировать кучу дополнительных компонентов.
- В случае с другими операционными системами вам, должно быть, придется скомпилировать ваш собственный MapServer из исходных кодов. При компиляции вы также, как правило, можете указать о необходимости скомпилировать и библиотеки PHP MapScript.

Использование карт на PHP

В этом трюке будет рассказано о том, как использовать файлы с картами, как ими манипулировать, как создавать карты с нуля, — и все это при помощи простого сценария на PHP.

Шаг 1. Подготовка данных. Прежде чем вы сможете приступить к созданию карты, вам придется подготовить кое-какие базовые данные о ней. Для данных трюков требуется только один файл: фотография, сделанная при помощи спутника. Они хранятся в формате GeoTIFF. На рис. 9.1 показан пример фотографии облачности над поверхностью земного шара.

Рис. 9.1. Карта, отображающая участок Земли с Северной Америкой

ПРИМЕЧАНИЕ

Вы можете загрузить этот пример фотографии облаков по адресу http://examples.oreilly.com/phphks.

Формат рисунков TIFF, должно быть, вам уже знаком, но GeoTIFF — это более продвинутый формат, позволяющий хранить в изображении географические координаты. Это позволяет использовать файл с картографическими данными, когда вы хотите рассмотреть какую-то определенную географическую область (читай — определенную широту и долготу). Формат GeoTIFF может быстро поставить в соответствие пикселам из определенного столбца и строки географические координаты X и Y.

ПРИМЕЧАНИЕ

Чтобы получить остальные данные, включая спутниковые фотографии, пограничные линии стран, доступ к удаленным интернет-сервисам и т. д., используйте мою книгу «Картография в Интернете» (O'Reilly) или посетите сайт http://freegis.org/.

Для использования этих примеров создайте директорию, в которой вы будете хранить сценарии, и поддиректорию с названием data, в которой вы будете хранить ваши изображения. Распакуйте архив `cloud_image.zip` в директорию **data,** в результате в ней появятся два файла: с расширением TIF (само изображение) и TFW-файл, в котором хранится связанная с картографией информация.

Шаг 2. Отображение карты. Простой файл MapServer с картой и несколько строк кода на PHP — вот и все, что вам необходимо для создания карты. Небольшой сценарий на PHP, который загружает карту и выводит на экран ее изображение, находится в примере 9.1.

Пример 9.1. Простое использование MapServer

```
<?PHP
// ex1_map_basic.php
// Tyler Mitchell. August 2005
//Построение карты при использовании файла с заранее сделанной картой
// Загружаем расширение MapScript
if (!extension_loaded("MapScript"))
dl('php_mapscript.'.PHP_SHLIB_SUFFIX);
// Создаем карту
$oMap = ms_newMapObj("map_points.map");
//Отображаем карту как объект изображения
$oMapImage = $oMap->draw( );
//Сохраняем карту в качестве файла с изображением
$oMapImage->saveImage("worldmap.png");
?>
```

Как вы можете видеть, чтобы выполнить по порядку все шаги для создания карты, действительно понадобилось всего несколько строк кода на PHP.

1. Загрузка расширения MapScript.
2. Открытие файла с картой.
3. Отрисовка изображения с картой.
4. Сохранение изображения в файл.

Результат работы сценария — одно единственное изображение `worldmap.png`. Откройте его в вашем любимом приложении для просмотра изображений, и вы

увидите изображение поверхности Земли (как это показано на рис. 9.1). Все настройки карты находятся в конфигурационном файле. В примере 9.2 вы можете видеть содержимое файла с картой. В нем содержатся данные об одном слое карты, использующие для этого фотографию со спутника в формате GeoTIFF.

ПРИМЕЧАНИЕ

Это крайне простой пример, который, возможно, даже не будет работать вне среды MapScript (например, если вы будете использовать CGI MapServer). Для получения более детальной информации о возможных настройках, используемых в конфигурационном файле MapServer, изучите соответствующую документацию на сайте MapServer: http://ms.gis.umn.edu/.

Пример 9.2. Простой файл с картой

```
MAP
NAME MAP_POINTS
SIZE 600 300
EXTENT -180 -90 180 90
LAYER
NAME clouds
TYPE RASTER
STATUS DEFAULT
DATA "data/global_clouds.tif"
END
END
```

Шаг 3. Внесение изменений в карту. Конечно же, основные преимущества использования карт с PHP заключаются в возможности управления ими. В примере 9.1 мы не изменили никаких настроек и не добавили ничего к тому, что уже было в файле с картой. Этот файл статический, но, тем не менее, PHP может намного больше, чем просто считывать данные из уже существующего файла карты. Например, возможно, вам захочется изменить географические рамки, которые покрывает карта. Вместо использования стандартного ключевого слова EXTENT в файле карты, вы можете создать свой собственный. Все, что вам нужно, это задать в свойствах объекта карты в качестве области ваши собственные координаты (при помощи функции setExtent()). Для установки границ требуются две пары координат: первая, чтобы указать юго-западный угол карты, а вторая, чтобы указать северо-восточный угол. Вам придется добавить в ваш сценарий всего одну строку, как это показано в примере 9.3.

Пример 9.3. Изменение настроек карты

```
<?PHP
// ex3_map_change.php
// Tyler Mitchell, August 2005
//Построение карты с использованием файла с заранее сделанной картой.
//Изменяем одно свойство.
//Загрузка расширения MapScript
if  (!extension_loaded("MapScript"))
dl('php_mapscript.'.PHP_SHLIB_SUFFIX);
//Создаем карту
$oMap - ms_newMapObj("ex2_map_basic.map");
//Изменяем свойство объекта карты extent
$oMap->setExtent(-130.20.  -70.70);
```

```
//Отображаем карту в объект изображения
$oMapImage = $oMap->draw( );
//Сохраняем карту как файл изображения
$oMapImage->saveImage("worldmap.png");
?>
```

Полученная в результате карта показана на рис. 9.1.

Ваш сценарий сможет делать намного больше, если вы наладите взаимодействие между ним и пользователем. Создав в вашем приложении ссылку на код, использующий MapScript, вы можете добавить просто неограниченные возможности, позволяющие изменять размеры карты, включать и выключать слои, добавлять на карту дополнительные данные, изменять цвета и делать с ней все, что только придет вам на ум. Обратитесь к документации по PHP MapScript (http://maptools.org/php_mapscript/index.phtml?page=docs.html), чтобы узнать о других методах и свойствах, используемых при работе с картой.

Шаг 4. Создание карты с нуля. Вы также можете создать приложение на PHP MapScript с нуля, то есть не используя уже существующих файлов с картой. Это может понадобиться в двух ситуациях: вы будете использовать сценарий для создания новых файлов с картой либо захотите избежать использования внешних файлов карт. Как вы уже, возможно, догадались, вам придется создать объект и установить значения для всех его свойств при помощи кода на PHP. Таким образом, ваш сценарий будет слегка больше, чем если бы вы использовали существующий файл карты, но благодаря такому подходу вы можете управлять всеми настройками ваших карт прямо в коде PHP.

Очень часто приложения PHP MapScript используются для динамического формирования на карте месторасположения. Для их создания есть соответствующие методы, которые называются встроенными топографическими элементами, координаты месторасположения (точки, линии или многоугольные области) задаются прямо в файле карты. В примере 9.4 показан файл с картой, в который добавлена одна точка и сопровождающее ее текстовое пояснение. В файл карты также добавлен символьный объект, который сообщает MapScript, как следует отобразить точку на карте. Вся основная работа происходит в конце в объекте, представляющем из себя новый слой.

Пример 9.4. Дополнительный слой, размещенный поверх карты

```
MAP
NAME MAP_POINTS
SIZE 600 300
EXTENT -180 -90 180 90

SYMBOL
NAME "circle"
TYPE ELLIPSE
FILLED TRUE
POINTS
1 1
END
```

```
END

LAYER
NAME clouds
TYPE RASTER
STATUS DEFAULT
DATA "data/global_clouds.tif"
END

LAYER
NAME custom_points
TYPE POINT
STATUS DEFAULT
FEATURE # Inline feature definition
POINTS
-121 54
END
TEXT "My Place"
END
CLASS
COLOR 250 0 0
OUTLINECOLOR 255 255 255
SYMBOL "circle"
SIZE 10
LABEL
POSITION AUTO
COLOR 250 0 0
OUTLINECOLOR 255 255 255
END
END
END
END
```

Полученная в результате карта показана на рис. 9.2.

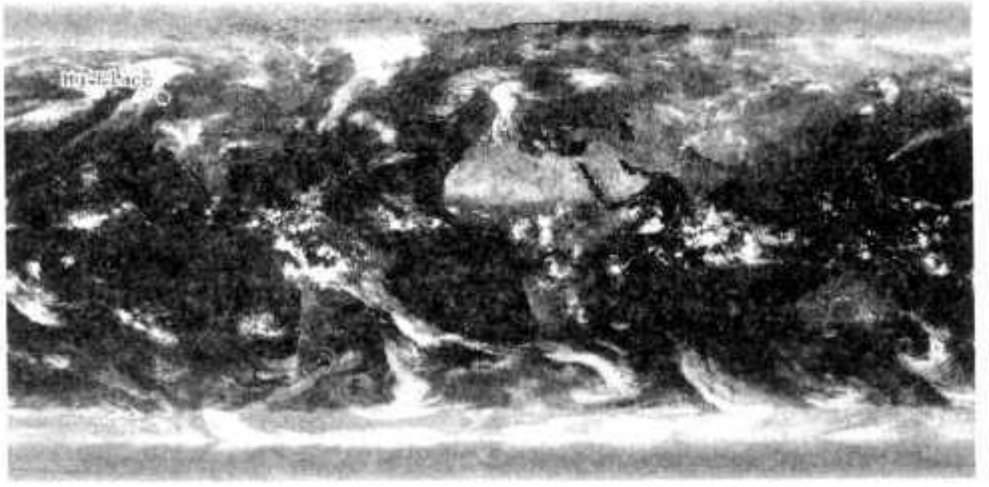


Рис. 9.2. Карта с добавленной на нее точкой и обозначением

ПРИМЕЧАНИЕ

Не смущайтесь, что здесь используются символьные настройки. Хотя это и выглядит слегка запутанным, скорее всего, вам не придется добавлять новые символы, разве что только вы не перейдете к использованию более продвинутых символов или разметки областей.

Хотя файл карты и очень удобный в использовании, по своей природе он не динамичный, именно здесь на помощь и приходит PHP. Пример 9.5 более сложный, в нем показано, как создать еще одну точно такую же карту (как на рис. 9.1), но затем нанести на нее дополнительный слой «точек». Этот сценарий получает координаты из текстового файла и динамически добавляет на карту при ее отрисовке. Все, что вам понадобится, чтобы указать месторасположение этих точек, это простой текстовый файл (points.txt), содержащий координаты точек в виде широты/долготы и текстовые обозначения, разделенные запятыми (и отсутствие какого-либо заголовка).

ПРИМЕЧАНИЕ

Вам следует сохранить файл points.txt в поддиректории data.

Пример 9.5. Пример карты, использующей GeoTIFF

```
<?PHP
// ex5_map_points.php
// Построение карты с использованием GeoTIFF
// и текстового файла с координатами/метками.
// Не требуется запускать файл с картой mapserver.
// Tyler Mitchell, август. 2005
// Загрузка расширения MapScript
if  (!extension_loaded("MapScript"))
  dl('php_mapscript.'.PHP_SHLIB_SUFFIX);
// Создаем объект карты. Пустая строка нужна, если не
// используем существующий файл с картой
$oMap = ms_newMapObj("");
// Задаем размер изображения карты
$oMap->setSize(600,300);
// Задаем географические размеры карты.
$oMap->setExtent(-180,-90,180,90);
// Создаем обозначение карты,используемое как шаблон кисти
// для отрисовки возможносей карты (линии, точки, и пр.)
$nSymbolId = ms_newSymbolObj($oMap, "circle");
$oSymbol = $oMap->getsymbolobjectbyid($nSymbolId);
$oSymbol->set("type", MS_SYMBOL_ELLIPSE);
$oSymbol->set("filled", MS_TRUE);
$aPoints[0] = 1;
$aPoints[1] = 1;
$oSymbol->setpoints($aPoints);
// Создаем уровень данных и ассоциируем его с картой.
// Это растровый уровень, показывающий некоторую облачность
$oLayerClouds = ms_newLayerObj($oMap);
$oLayerClouds->set( "name", "clouds");
$oLayerClouds->set( "type", MS_LAYER_RASTER);
$oLayerClouds->set( "status", MS_DEFAULT);
$ol_ayerClouds->set(  "data","data/global_clouds.tif");
```

```
// Создаем другой уровень для нанесения точек
$oLayerPoints = ms_newLayerObj($oMap);
$oLayerPoints->set( "name". "custom_points");
$oLayerPoints->set( "type". MS_LAYER_POINT);
$oLayerPoints->set( "status", MS_DEFAULT);
// Открываем файл с координатами и метками (x,y,label)
$fPointList = file("data/points.txt");
// Для каждой строки в текстовом файле
foreach ($fPointList as $sPointItem)
{
  $aPointArray = explode(",".$sPointItem);
  // :Уловка: Хотя мы создаем точки.
  // мы обязаны использовать объект линии (newLineObj)
  // только с одной точкой. Я называю этот объект CoordList
  // для простоты, поскольку мы рисуем не совсем линию.
  $oCoordList - ms_newLineObj( ):
  $oPointShape = ms_newShapeObj(MS_SHAPE_POINT);
  $oCoordList->addXY($aPointArray[0],$aPointArray[1]);
  $oPointShape->add($oCoordList);
  $oPointShape->set( "text". chop($aPointArray[2]));
  $oLayerPoints->addFeature($oPointShape);
}
// Создаем объект для задания особенностей отрисовки стилей
$oMapClass = ms_newClassObj($oLayerPoints);
// Создаем объект стиля, определяющий отрисовку особенностей
$oPointStyle = ms_newStyleObj($oMapClass);
SoPointStyle->color->setRGB(250.0.0);
SoPointStyle->outlinecolor->setRGB(255,255,255);
$oPointStyle->set( "symbolname", "circle");
$oPointStyle->set( "size". "10");
// Создаем настройки меток для отрисовки тестовых меток
$oMapClass->label->set( "position". MS_AUTO);
$oMapClass->label->color->setRGB(250,0,0);
$oMapClass->label->outlinecolor->setRGB(255,255,255);
// Отображаем карту в объекте изображения
$oMapImage = $oMap->draw( ):
// Сохраняем карту в файл
$oMapImage->saveImage("worldmap.png");
?>
```

Рассмотрим список точек, которые следует нанести на карту. Он хранится в файле points.txt.

```
-118.35,34.06,Angie
-118.40,34.03,Ray
-111.99,33.52,Alice
-95.45,29.75,David
144.85,-37.85,Mark
```

Результат работы сценария показан на рис. 9.3. Чтобы еще немного улучшить этот пример, вы можете вынести ту часть кода, которая отображает точки, в отдельный класс. В таком случае вы сможете использовать его в других PHP-приложениях, например в сценариях, которые в качестве входных данных принимают введенные пользователем координаты и отображают на вашей карте указанные точки.

Рис. 9.3. Карта с нанесенными на нее точками и обозначениями, взятыми из текстового файла

Узнать больше

Есть множество источников, из которых можно получить дополнительную информацию о MapServer и других геопространственных технологиях с открытым кодом. Такого рода сообщества активно используют почтовые рассылки и каналы IRC для обсуждений и даже проводят ежегодные конференции.

Чтобы получить информацию о пользователях, живущих неподалеку от вас, запросите список адресатов. Эти списки в MapServer являются отправной точкой для многих других проектов, так как они и предназначены для этих целей. Это и будет лучшее место, где вы сможете найти помощь. В книге «Картография в Интернете» (O'Reilly) есть множество информации о MapServer, географических базах данных, сетевых службах OGC, преобразовании данных, проекциях карт и многом другом. Есть еще несколько книг по схожей тематике (с открытым кодом), такие как «Трюки в картографировании» (O'Reilly), «MapServer для начинающих» (Apress) и «Прикладные ГИС» (Pragmatic Bookshelf).

Смотрите также

- «Создание пользовательских карт Google» (см. трюк 95).

ТРЮК №87

Создание графических пользовательских интерфейсов при помощи GTk

Используйте GTk для создания кроссплатформенных графических пользовательских интерфейсов в вашем PHP-коде.

Не будем останавливаться на использовании разметки для Интернета или тегов. Почему бы не воспользоваться самим Рабочим столом в вашем коде на PHP? С таким инструментом, как GTk, вы можете сделать это очень просто. В этом трюке рассказано, как использовать GTk для разработки приложения, которое будет использоваться для проверки простых регулярных выражений.

Код

Сохраните исходный код из примера 9.6 в файл retest.php.

Пример 9.6. Программа для проверки регулярных выражений, написанная с использованием GTk

```
<?php
if( !extension_loaded('gtk')) {
  dl( 'php_gtk.'.PHP_SHLIB_SUFFIX);
}
$start_regex = "/name:\\s*(.*?)\\n/";
$start_text = "name: Jack\nname:Lori\nname:Megan\n";

function delete_event( ) { return false; }
function shutdown( ) { gtk::main_quit( ); }
function run( )
{
  global $rb_regex, $tb_text, $ft;
         $regex = $rb_regex->get_chars(0, -1);
         $text = $tb_text->get_chars(0, -1);
  preg_match_all( $regex, $text, $found );
  $ft->clear( );
  $i = 0;
  foreach( $found[1] as $f )
  {
    $ft->insert( $i, array( $f ) );
    $i++;
  }
}
$window = &new GtkWindow( );
$window->set_usize( 700, 400 );
$window->set_title( "Regular Expression Tester" );
$window->connect('destroy', 'shutdown');
$window->connect('delete-event', 'delete_event');
$bb = new GtkTable( );
$rb = new GtkTable( );
$rb_label = new GtkLabel( "Regex:" );
$rb->attach( $rb_label, 0, 1, 0, 1, GTK_SHRINK, GTK_SHRINK, 5, 5 );
$rb_regex = new GtkEntry( );
$rb_regex->insert_text( $start_regex, 0 );
$rb->attach( $rb_regex, 1, 2, 0, 1, GTK_FILL, GTK_SHRINK, 5, 5 );
$rb_run = new GtkButton( "Run" );
$rb_run->connect('clicked', 'run');
$rb->attach( $rb_run, 2, 3, 0, 1, GTK_SHRINK, GTK_SHRINK, 5, 5 );
$tb_label = new GtkLabeK "Text:" );
$rb->attach( Stbjabel, 0, 1, 1, 2, GTK_SHRINK, GTK_SHRINK, 5, 5 );
$tb_text = new GtkText( );
$tb_text->set_editable( true );
$tb_text->insert_text( $start_text, 0 );
```

```
$rb->attach( $tb_text. 1. 2. 1, 2. GTKJILL. GTKJILL. 5, 5 );
$bb->attach( $rb,0,1,0,1,GTK_SHRINK,GTK_FILL,5,5 ):
$ft = new GtkCList( 1 ):
$bb->attach( $ft.I.2.0,I );
$w1ndow->add( $bb ):
$window->show_all( ):
gtk::main( ):
?>
```

Большинство кода в этом примере предназначено для создания графического пользовательского интерфейса. Для начала создается новый объект типа GtkWindow. После этого для окна создаются элементы управления, которые размещаются на нем. Код для регулярных выражений начинает свою работу, когда пользователь нажимает кнопку Run. Она связана с функцией run() при помощи метода connect() объекта $rb_run. Функция гип() получает текущее содержимое переменной $rb_regex и содержимое текстовых полей, а затем выполняет регулярное выражение. После этого функция очищает содержимое таблицы и добавляет в нее все значения, совпадающие с регулярным выражением при помощи метода insertO.

Запуск трюка

Этот трюк следует запускать из командной строки при помощи специальной версии интерпретатора PHP 4 с поддержкой GTk, который вы можете загрузить по адресу http://gtk.php.net/:

```
c:\retest\ > php retest.php
```

ВНИМАНИЕ

Несмотря на то, что вы все равно работаете с PHP, вам нужна версия интерпретатора с поддержкой GTk. Иначе при попытке запуска файла retest.php в результате вы получите несколько неприятных ошибок.

В результате выполнения этой команды загрузится окно (рис. 9.4). Далее нажмите кнопку Run, чтобы запустить регулярное выражение, расположенное в самом верхнем текстовом поле применительно к текстовой строке, которая находится в нижнем, более крупном текстовом поле. Результат вы можете видеть на рис. 9.5.

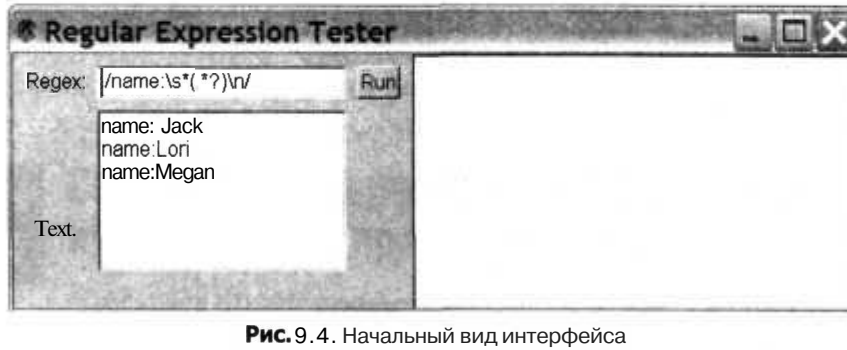


Рис. 9.4. Начальный вид интерфейса

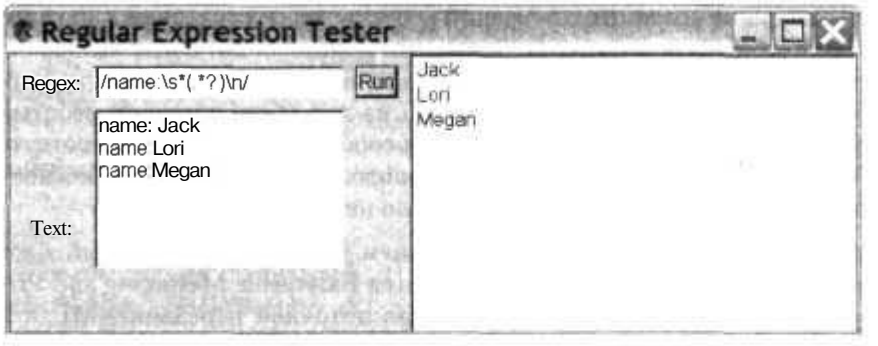


Рис. 9.5. После нажатия кнопки Run

Здесь говорится, что были найдены три строки: Jack, Lori и Megan. Вы можете изменять регулярное выражение и текст, как вам заблагорассудится. На самом деле у нас получилась очень полезная утилита, и она может вам очень сильно пригодиться, если вы только приступили к изучению таинственного мира регулярных выражений, например при работе с трюком для переименования URL-адресов (см. трюк 60).

Более подробное объяснение принципов работы GTk выходит за рамки этого трюка. Достаточно будет сказать, что там присутствуют все стандартные элементы управления Windows, а также возможности настройки рисования. Более того, этот инструмент можно использовать в Windows, Mac и Linux, что дает вашим PHP-сценариям намного большую переносимость, чем могут предоставить другие языки.

Однако, стоит отметить, что GTk не идеален. В целом вы обладаете меньшими возможностями по управлению форматированием, чем при использовании для программирования на C или C++ родного Windows API. Но для таких приложений, как это, или для создания удобного виджета, который, например, будет проверять состояние вашего сервера, возможности располагать элементы с точностью до пикселя, скорее всего, и не понадобится.

Смотрите также

- «Упрощенная работа с XML при помощи регулярных выражений» (см. трюк 38).

ТРЮК № 88

Передача данных из RSS-источников в ваше приложение для отправки сообщений при помощи Jabber

Используйте PHP и Jabber для передачи данных из RSS-источников в ваше приложение для отправки мгновенных сообщений.

Использование мгновенных сообщений — очень распространенный способ общения. Некоторые исследования показали, что пользователи с небольшим опытом работы в Интернете больше полагаются на мгновенные сообщения, чем на сообщения по электронной почте. К сожалению, из-за особого принципа работы большинства популярных систем для отправки сообщений, а также из-за отсутствия в HTTP-соединении информации о местоположении встроить возможность отправки сообщений в PHP-приложения было не очень просто.

Протокол с открытым кодом под названием Jabber, разработанный Джереми Миллером в 1998 году (теперь он называется Extensible Messaging and Presence Protocol [XMPP]), — это родной протокол поточной передачи XML, а также утвержденный IETF стандарт для указания местоположения и передачи сообщений в Интернете. К тому же, что является особо важным для нас, XMPP позволяет PHP-сценариям получать доступ к приложениям для отправки мгновенных сообщений. В этом трюке на PHP будет создан Jabber-клиент для командной строки, использующий доступный бесплатно модуль `class.jabber.php` в качестве связующего звена с протоколом XMPP.

Другой очень популярный XML-протокол под названием RSS позволяет сайту публиковать свое содержимое сразу в нескольких изданиях в виде отдельных источников. Программы для чтения новостей и веб-страницы в поисках новостей периодически опрашивают эти URL-адреса. Jabber-клиент, который мы создадим, будет получать данные о погоде из нескольких RSS-источников и рассылать эту информацию в виде мгновенных сообщений.

Код

Сохраните исходный код из примера 9.7 в файл `client.php`.

Пример 9.7. Пример Jabber-клиента

```
<?php
/* CONFIG VARIABLES */
// jabber-сервер, на котором вы регистрируетесь
$SERVER = 'yourserver';
//Имя и пароль для вашей учетной записи
$USERNAME = 'yourusername';
$PASSWORD = 'yourpassword';
// jabber id для вашей учетной записи
$PERSONAL = 'username@yourserver';
//rss url для оповещений
$NOAA = 'http://www.nws.noaa.gov/alerts/ct.rss';
/* END CONFIG */
function send($to, $msg) {
  global $JABBER;
        $JABBER->SendMessage("$to","normal", NULL, array("body" =>
          htmlspecialchars($msg)),$payload);
}

//Замещаем обработчик jabber.class.php
function Handler_message_normal($message) {
```

```
  global SJABBER;
         $from = SJABBER->GetInfoFromMessageFrom(Smessage);
         Sbody = $JABBER->GetInfoFromMessageBody($message);
  if (substr ($body ,0.3) — SMS) {
    $bodyparts = explode(":", $body):
    $zip = $bodyparts[1];
    weatherize($from, $zip);
  }
}

function Handler_message_chat($message) {
  Handler_message_normal($message);
}

// RSS-функции адаптированы к PHP RSS Reader v1.1 (Richard James Kendall)
function startElement($parser, Sname. $attrs) {
  global $rss_channel, $currently_writing, $main;
  switch($name) {
    case "RSS":
    case "RDF:RDF":
    case "ITEMS":
      $currently_writing = "";
      break:
    case "CHANNEL":
      $main = "CHANNEL":
      break;
    case "IMAGE":
      $main = "IMAGE";
      $rss_channel["IMAGE"] - array( ):
      break;
    case "ITEM":
      $main = "ITEMS":
      break:
    default:
      $currently_writing » Sname;
      break;
  }
}

function endElement($parser. Sname) {
  global $rss_channel, $currently_writing, $item_counter;
         $currently_writing = "":
  if (Sname — "ITEM") {
    $itern_counter++;
  }
}

function characterData($parser, $data) {
  global $rss_channel, $currently_writing, Smain, $item_counter;
  if ($currently_writing != "") {
    switch($main) {
      case "CHANNEL":
        if (isset(Srss_channel[$currently_writing])) {
          $rss_channel[$currently_writing] .= Sdata;
        }else {
```

```
            $rss_channel[$currently_writing] = $data;
          }
          break;
        case "IMAGE":
          if (isset($rss_channel[$main][$currently_writing])) {
            $rss_channel[$main][$currently_writing] .= $data;
          } else {
            $rss_channel[$main][$currently_writing] = $data;
          }
          break;
        case "ITEMS":
          if (isset($rss_channel[$main][$item_counter][$currently_writing])) {
            $rss_channel[$main][$item_counter][$currently_writing] .- $data:
          } else {
            $rss_channel[$main][$item_counter][$currently_writing] - $data;
          }
          break;
      }
    }
}

function parseXML($url) {
  global $rss_channel, $currently_writing. $main, $item_counter;
         $file = $url:
         $last_item - $_REQUEST['last_item'];
         $rss_channel = array( );
         $currently_writing - "";
         $main - "";
         $item_counter = 0;
         $xml_parser - xml_parser_create( );
  xml_set_element_handler($xml_parser, "startElement", "endElement");
  xml_set_character_data_handler($xml_parser, "characterData");
  if (!($fp - fopen($file. "r"))) {
    die("could not open XML input");
  }
  while ($data = fread($fp. 4096)) {
    if (!xml_parse($xml_parser, $data. feof($fp))) {
      die(sprintf("XML error: %s at line %d",
      xml_error_string(xml_get_error_code($xml_parser)),
      xml_get_current_line_number($xml_parser)));
    }
  }
  xml_parser_free($xml_parser);
}

function NOAA( ) {
  global $rss_channel, $currently_writing, $main, $item_counter;
  global $last_item;
  global $message;
         $message="";
  parseXML($NOAA);
  if (isset($rss_channel["ITEMS"])) {
    if (count($rss_channel["ITEMS"]) > 0) {
      for($i = 0;$i < count($rss_channel["ITEMS"]);$i++) {
```

```
        if ($rss_channel["ITEMS"][count($rss_channel["ITEMS"])-
          1]["TITLE"] == $last_item) { break; } //ничего нового
        Smessage .= $rss_channel["ITEMS"][$i]["TITLE"]."\r\n".$rss_
        channel["ITEMS"][$i]["LINK"]."\r\n\r\n";
      }
    } else {
      Smessage = "There are no articles in this feed.";
    }
  }
$last_item = $rss_channel["ITEMS"][count($rss_channel["ITEMS"])-1]["TITLE"];
  If (Smessage !- '') {
    send($PERSONAL, Smessage);
  }
}

function weatherize($from, $zip) {
  global $rss_channel, $currently_writing, $main, $item_counter;
        $wunderurl = 'http://www.wunderground.com/cgi-bin/findweather/
  getForecast?brand=rss_full&query='.$zip;
  parseXML($wunderurl);
  if (isset($rss_channel["ITEMS"])) {
    if (count($rss_channel["ITEMS"]) > 0) {
      for(Si = 0;$i < count($rss_channel["ITEMS"]);$i++) {
        $wunderground .= $rss_channel["ITEMS"][$i]["TITLE"]."\r\n".$rss_
        channel["ITEMS"][$i]["LINK"]."\r\n".$rss_
        channel["ITEMS"][$i]["DESCRIPTION"]."\r\n\r\n";
      }
    } else {
      Smessage = "There are no articles in this feed.";
    }
  }
  send(Sfrom, Swunderground);
}

// Конец RSS-функций
require("class.jabber.php");
$JABBER = new Jabber;
SJABBER->server = $SERVER;
SJABBER->port = "5222";
SJABBER->username - SUSERNAME;
SJABBER->password = $PASSWORD;
$JABBER->resource = "client.php";
$JABBER->enable_logging - FALSE;
SJABBER->Connect( ) or die("Couldn't connect!");
SJABBER->SendAuth( ) or dieC'Couldn't authenticate!");
$JABBER->SubscriptionAcceptRequest($PERSONAL);
while(true) {
  $JABBER->SendPresence(NULL, NULL, "online");
  NOAA( ):
  SJABBER->CruiseControl(15 * 60);
}
// может, никогда не получим, но...
SJABBER->Disconnect( );
?>
```

Запуск трюка

Чтобы завершить этот трюк, вам понадобится по крайней мере одна новая учетная запись в системе Jabber, а также ваша собственная учетная запись для получения сообщения. У вас есть богатый выбор открытых сервисов и Jabber-клиентов.

ПРИМЕЧАНИЕ

Вы можете получить список открытых сервисов по адресу http://www.jabber.org/network/, а также список клиентов по адресу http://www.jabber.org/software/clients.shtml.

Загрузите и установите программу-клиента, если у вас ее еще нет, и создайте новую учетную запись в одном из сервисов Jabber. Ее вы будете использовать в качестве веб-агента (иногда называемого роботом). Если вам нужна личная учетная запись в системе Jabber, зарегистрируйте и ее.

Далее вам необходимо загрузить доступный бесплатно модуль `class.jabber.php`, который изначально был создан Карлом «Gossip» Зоттманом, а теперь его поддержкой занимается Натан «Fritzy» Фриц. Этот модуль доступен в сети по адресу http://code.blitzaffe.com/pages/phpclasses/files/jabber_client_52-11. Поместите файл в ту же директорию, в которую вы загрузили остальные файлы из трюка.

В целом этот класс — упрощенная версия процесса взаимодействия с протоколом XMPP. Используя его, мы собираемся создать простую программу-клиента для командной строки, которая будет выполняться в виде связки-демона (не того демона, о котором некоторые из вас подумали) и будет предоставлять нам доступ из PHP к клиенту для работы с мгновенными сообщениями.

Скопируйте исходный код из примера 9.7 в файл с названием `client.php`. Измените переменные настройки (выделенные полужирным шрифтом в начале сценария), указав в них данные вашей учетной записи в системе Jabber и информацию о вашем сервере. После этого поместите этот сценарий в той же директории, в которую вы скопировали модуль `class.jabber.php`. Добавьте учетную запись, которую вы используете в сценарии для связи с вашей собственной учетной записью в системе Jabber, в список контактов (это по сути то же самое, что добавить приятеля в список контактов Jabber-клиента). Затем запустите сценарий из командной строки при помощи PHP-интерпретатора:

```
php client.php &
```

Вы должны будете получить сообщение о текущей погоде. Каждые 15 минут сценарий будет следить, не произошли ли какие-либо изменения. Если изменения будут, он автоматически уведомит вас об этом. Пример RSS-источника вы можете видеть на рис. 9.6.

ПРИМЕЧАНИЕ

Отдельное спасибо хочется сказать Ричарду Джеймсу Кенделлу. Функции для сценария, анализирующие RSS-источники, были взяты из приложения PHP RSS Reader, которое вы можете найти on адресу http://richardjameskendall.com/. \

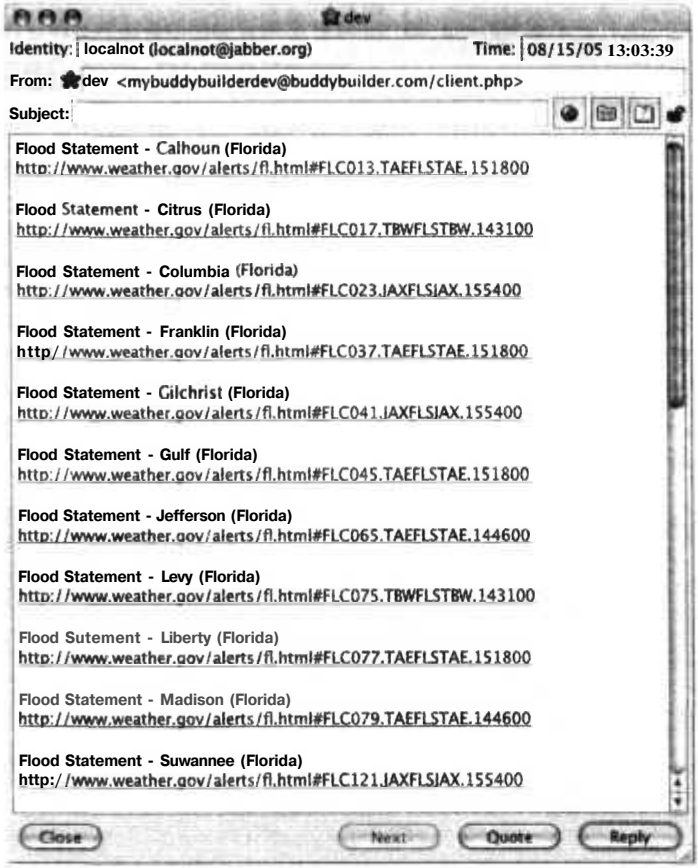


Рис. 9.6. RSS-источник в вашей программе для обмена сообщениями

Улучшаем трюк

Я уверен, что большинству было бы очень интересна возможность получать ежедневную информацию о погоде, когда им этого захочется. Я именно из этой категории людей, поэтому и воспользовался справочными средствами Weather Underground (http://www.wunderground.com/). Они любезно предоставляют доступ к RSS, который принимает в качестве параметра почтовый индекс и возвращает информацию о текущей погоде в виде XML. Единственной хитростью, которой нам придется воспользоваться, будет изменение клиента так, чтобы он также ожидал получения сообщений. У модуля `class.jabber.php` есть очень удобная особенность, позволяющая определять функции, замещающие стандартные обработчики сообщений:

```
function Handler_message_normal($message) {
  global $JABBER;
         $from = $JABBER->GetInfoFromMessageFrom($message);
         $body = $JABBER->GetInfoFromMessageBody($message);
```

```
  if (substr ($body ,0.8) — weather:) {
    $bodyparts = explode(":", $body);
    $zip - $bodyparts[l]:
    weatherize($from, $zip);
  }
}

function Handler_message_chat($message) {
  Handler_message_normal($message);
}
```

Мы хотим отвечать только на сообщения, запрашивающие прогноз погоды, и для них мы будем использовать довольно простой формат:

```
weather:zipcode
```

Все входящие сообщения, которые начинаются со слова weather вместе со следующим за ним двоеточием, будут вызывать функцию `weatherize()`. Пример этого сообщения вы можете видеть на рис. 9.7.

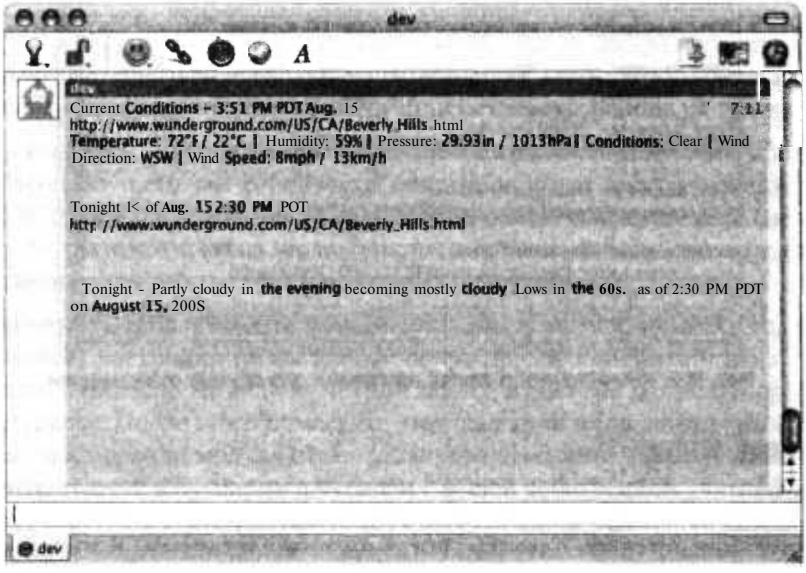


Рис. 9.7. Общение с вашим PHP-сценарием посредством системы Jabber

Похожим способом вы можете добавить любые специфические функции, какие хотите, которые будут реагировать на другие ключевые слова и возвращать определенные сообщения в качестве ответа.

Смотрите также

- «Использование IRC в ваших веб-приложениях» (см. трюк 89).
- «Получение информации из источников RSS в PSP» (см. трюк 90).

ТРЮК №89

Использование IRC в ваших веб-приложениях

Используйте Net_SmartIRC для общения с вашими интернет-приложениями через ваш веб-сервер.

Иногда веб-страница может быть не лучшим способом общения с приложением. Множество людей используют мгновенные сообщения и чаты (например, IRC) для общения друг с другом. Так почему же не позволить им общаться с вашим веб-приложением? На рис. 9.8 показаны пользователь и робот, общающиеся через IRC-сервер. Робот запускается из командной строки в качестве отдельного процесса PHP. Модуль PEAR Net_SmartIRC (см. трюк 2) позволяет вашему интернет-приложению авторизоваться на IRC-сервере и реагировать на посылаемые ему команды.

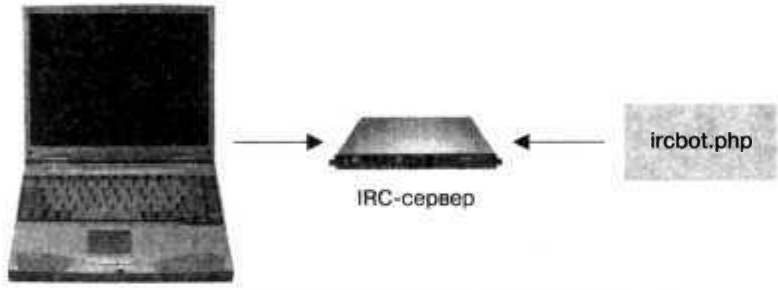


Рис. 9.8. Пользователь и робот общаются через IRC

Код

Сохраните исходный код из примера 9.8 в файл с именем ircbot.php.

Пример 9.8. Простой IRC-робот

```
<?php
include_once('Net/SmartIRC.php');
require_once('DB.php');
$dsn = 'mysql://root:password@localhost/books';
$db =& DB::Connect( $dsn, array( ) );
if (PEAR::isError($db)) { die($db->getMessage( )); }
class dbbot
{
  function listdata(&$irc, &$data)
  {
    global $db;
           $irc->message(SMARTIRC_TYPE_CHANNEL, $data->channel, 'Books: ' ) ;
           $res = $db->query( "SELECT name FROM book", array( ) );
    while( $res->fetchInto( $row, DB_FETCHMODE_ASSOC ) )
    {
      $irc->message(SMARTIRC_TYPE_CHANNEL, $data->channel, ' '.
      $row['name'] ):
```

```
    }
  }
}

$host = "localhost";
$port = 6667;
$nick = "DBBot";
$chan = "#db";
$bot = &new dbbot( );
$irc = &new Net_SmartIRC( );
$irc->setUseSockets( TRUE );
$irc->registerActionhandler( SMARTIRC_TYPE_CHANNEL, '^list', $bot, 'listdata' );
$irc->connect( $host, $port );
$irc->login( $nick, 'Database bot', 0, $nick );
$irc->join( array( $chan ) );
$irc->listen( );
$irc->disconnect( );
?>
```

Запуск трюка

Установите модуль PEAR Net_SmartIRC (см. трюк 2), а затем запустите сценарий ircbot.php:

```
% php ircbot.php
```

Используйте ваш IRC-клиент, чтобы связаться с IRC-сервером, указанным в сценарии. Затем зайдите на канал #db и отправьте в него текст list. Как вы можете видеть на рис. 9.9, IRC-робот ответит вам, выдав список записей из базы данных books.

Рис. 9.9. Общение с вашим приложением при помощи IRC

Без сомнения, ваше приложение будет обладать своей собственной бизнес-логикой и иметь другой набор команд и вариантов ответа на них. Вы можете добавить дополнительные команды, вызвав функцию `registerActionHandler()` и передав в нее другие обработчики команд в качестве параметров.

ПРИМЕЧАНИЕ

В основном мы использовали этот механизм для взаимодействия с базой данных ошибок. IRC использовал имя ошибки, которое совпадало с ее названием в базе данных. Затем при помощи простого набора команд можно было вывести список существующих в данный момент ошибок или ошибок, которые были исправлены на этой неделе, детальное описание ошибки, пометить ошибку как исправленную или даже загрузить файл, который непосредственно связан с этой ошибкой.

Смотрите также

- «Передача данных от RSS-источников в ваше приложение для отправки мгновенных сообщений при помощи Jabber» (см. трюк 88).

ТРЮК №90

Получение информации из RSS-источников в PSP

Используйте PHP и браузер системного программного обеспечения PSP 2.0 для получения информации из RSS-источников.

Портативная приставка Sony PlayStation (PSP) была убойным устройством еще до того, как заполучила в свое распоряжение встроенный интернет-браузер. Но теперь, получив такие дополнительные функции, это карманное устройство, при помощи которого можно играть и бродить по Интернету, стало просто незаменимым. В этом трюке создается RSS-источник данных, отформатированный таким образом, чтобы он нормально отображался в виде одной страницы на экране PSP.

Код

Сохраните исходный код из примера 9.9 в файл `index.php`.

Пример 9.9. HTML, читающий RSS и отформатированный для отображения на PSP

```
<?php
require_once( 'XML/RSS.php' );
function getValue( $node, $name )
{
  $nl = $node->getElementsByTagName( $name );
  return $nl->item(0)->nodeValue;
}
$feeds = array( );
$feeds []= array(
  'name' => 'Top Stories',
```

```
  'url' -> 'http://rss.cnn.com/rss/cnn_topstories.rss'
):
$feeds []= array(
  'name' => 'World'.
  'url' => 'http://rss.cnn.com/rss/cnn_world.rss'
):
$feeds []= array(
  'name' => 'U.S.',
  'url' => 'http://rss.cnn.com/rss/cnn_us.rss'
):
$feeds [] - array(
  'name' => 'Tech'.
  'url' => 'http://rss.cnn.com/rss/cnn_tech.rss'
):
$feed = 0;
if ( isset( $_GET['feed'] ) ) Sfeed - $_GET['feed'];
ob_start( );
$ch = curl_init( );
curl_setopt($ch. CURLOPT_URL, $feeds[$feed]['url'] );
curl_setopt($ch. CURLOPT_HEADER, 0):
curl_exec($ch);
curl_close($ch):
$rsstext = ob_get_clean( );
$cols = array( );
$cols []= "":
$cols []= "";
$cols []= "":
$col - 0:
Sdoc = new DOMDocument( ):
$doc->loadXML( $rsstext ):
$il = $doc->getElementsByTagName( "item" );
for( $i = 0: $i < $il->length; $i++ )
{
  $item = $il->item( $i );
  $title - getValue( $item. "title" ):
  Slink - getValue( $item, "link" ):
  $description = getValue( $itern, "description" );
  $html - "<p class='story'><a href=\"$link\">$title</a></p>";
  $cols[ $col ] .= $html;
  $col++:
  if ( $col >- 3 ) $col =0:
}
?>
<html>
<head>
<title><?php echo( $feeds[$feed]['name'] ) ?></title>
<style>
body { margin: 0px; padding: Opx; }
.link { font-weight: bold: margin-left: 10px; margin-right: 10px; }
</style>
</head>
<body>
<div style="width:478px;">
<div style="width:478px;border-bottom:1px solid black;margin-bottom: 5px:">
<?php Sid = 0: foreach( Sfeeds as Sf ) { ?>
<a class='link' href="index.php?feed=<?php echo($id); ?>">
<?php echo( Sf['name'] ); ?></a>
```

```
<?php $id++; } ?>
</div>
<table>
<tr>
<td valign="top" width="33%"><?php echo( $cols[0] ); ?></td>
<td valign="top" width="33%"><?php echo( $cols[1] ); ?></td>
<td valign="top" width="33%"><?php echo( $cols[2] ); ?></td>
</tr>
</table>
</div>
</body>
</html>
```

Действительно интересные части этого сценария расположены в его самом начале, где он находит выбранный в данный момент источник и загружает его при помощи CURL. Затем сценарий использует систему XML DOM для анализа RSS XML. При помощи функции `getElementByTagName()` находятся все элементы `INTEM` в XML и данные из них сохраняются в нескольких текстовых столбцах. Каждый столбец представляет собой запись элемента массива `$cols`. Во второй части сценария на экран выводится список доступных RSS-источников, а также заголовки статей выбранного в данный момент RSS-источника. Они представлены в HTML-формате, адаптированном для отображения на PSP.

Запуск трюка

Загрузите файл `index.php` на сервер и первым делом попробуйте открыть страницу в браузере Firefox (рис. 9.10).

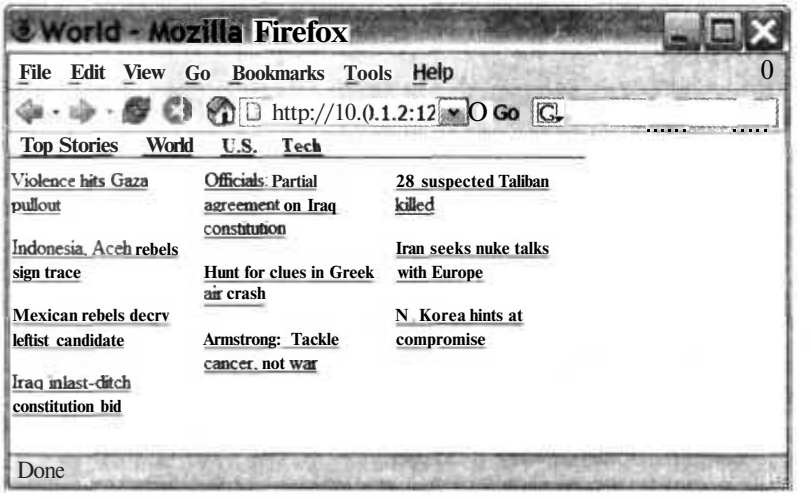


Рис. 9.10. Приложение для чтения RSS, открытое в браузере Firefox

Отлично, все выглядит довольно неплохо. Теперь взглянем на эту страницу на самом PSP. Откройте URL-адрес, указав его в соответствующем поле. Затем нажмите кнопку Остановить на панели инструментов браузера и загрузите URL (рис. 9.11).

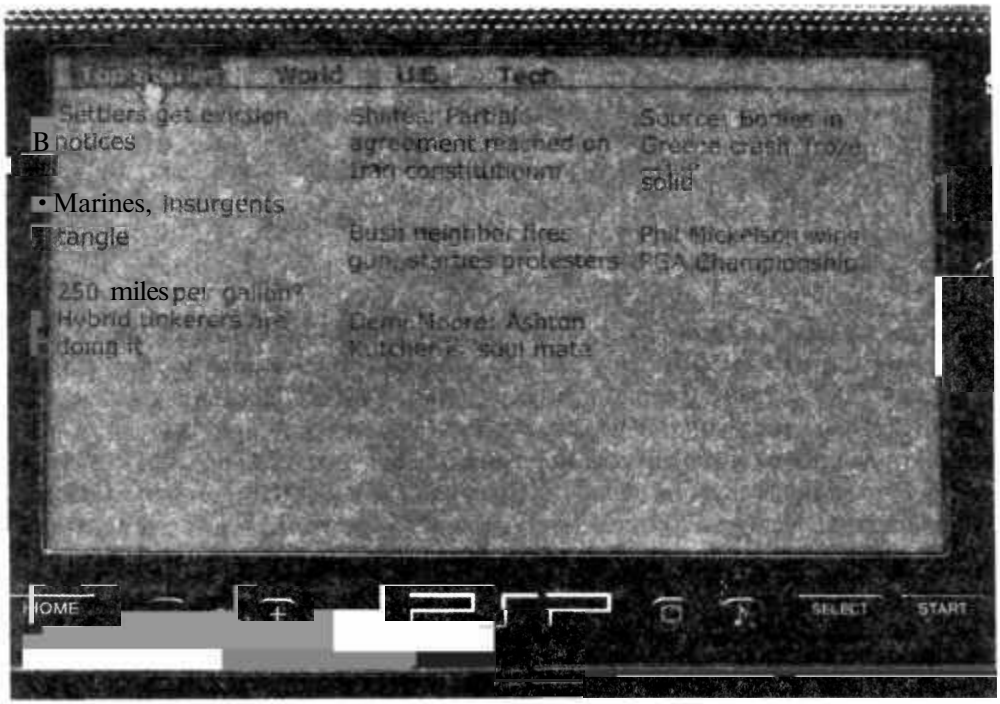


Рис. 9.11. Приложение для чтения RSS, загруженное в PSP

Вы можете пользоваться ссылками вверху страницы, чтобы выбирать различные категории новостей от источника CNN. Ссылки в центре страницы ведут к самим статьям из источника.

Смотрите также

- «Следите за вашей сетевой игрой при помощи PHP» (см. трюк 98).
- ❑ «Просмотр Википедии при помощи PSP» (см. трюк 99).

ТРЮК №91

Организация поиска в Google при помощи диаграммы ссылок

Используйте веб-сервис Google API и диаграмму ссылок в стиле Flikr, чтобы реализовать поиск в Google.

Google — это очень мощный поисковый механизм, но иногда я замечаю, что на просмотр отрывков из страниц затрачиваю времени больше, чем на просмотр самих страниц. Этот трюк просматривает фрагменты страниц и ищет в них повторяющиеся слова, заданные в качестве критерия для поиска. Это отличный способ провести более тщательный поиск.

Код

Сохраните исходный код из примера 9.10 в файл index.php.

Пример 9.10. Диаграмма ссылок, реализованная при помощи DHTML и использующая в качестве источника данных Google

```
<?php
require_once("Services/Google.php");
$ignore = array(
   'the','for','and','with','the','new','are','but','its','that','was',
   'your', 'yours', 'also', 'all', 'use', 'could', 'would', 'should', 'when',
   'they',
   'far', 'one', 'two', 'three', 'you', 'most', 'how', 'these', 'there', 'now',
   'our',
   'from', 'only', 'here', 'will' );
$ignorehash - array( ):
foreach( $ignore as $word ) { $ignorehash[ $word ] - 1; }

$term - "Code Generation";
if( array_key_exists( 'term', $_GET ) )
  $term = $_GET['term']:

$key = "GOOGLE KEY";

Sgoogle - new Services_Google( $key );
$google->queryOptions['limit'] = 50;
$google->search( $term );
$data = array( ):
foreach($google as $key -> $result)
{
  $data []= array(
    'title' -> $result->title.
    'snippet' => $result->snippet.
    'URL' => $result->URL
  ):
}
function jsencode( $text )
{
  $text = preg_replace( '/\'/', ". $text );
  return $text:
}
function get_words( $text )
{
  $text = preg_replaceC '/\<(.*?)\>/', ". $text );
  $text = preg_replace( '/[.]/', ''. $text ):
  $text = preg_replace( '/,/'. ". $text ) :
  $text = html_entity_decode( $text ):
  $text - preg_replace( '/\<(.*?)\>/', ", $text );
  $text - preg_replace( '/[\'|\"|\-|\+|\:|\[\@|\/|\\\\|\#|\!|\(|\)]/', '',
  $text );
  $text = preg_replace( 7\s+/', ' '. $text );
  Swords = array( );
  foreach( split( ' '. $text ) as $word )
  {
    Sword = strtolower( Sword );
```

```
    $word - preg_replace( '/^\s+/', '', $word );
    $word - preg_replace( '/\s+$/', '', $word );
    if( strlen( $word ) > 2 )
      $words [ ] - $word;
  }
  return $words;
}
$found - array( );
$id - 0;
foreach( $data as $row )
{
  $row['id'] - $id; $id += 1;
  $words - @get_words( $row['snippet'] );
  foreach( $words as $word )
  {
    if ( !array_key_exists( $word, $found ) )
    {
      $found[$word] - array( );
      $found[$word]['word'] - $word;
      $found[$word]['count'] - 0;
      $found[$word]['rows'] - array( );
    }
    $found[$word]['count'] += 1;
    $found[$word]['rows'][$row['URL']] - $row;
  }
}
$good - array( );
foreach( array_keys( $found ) as $text )
{
  if ( $found[$text]['count'] > 1 && array_key_exists( $text, $ignorehash )
    - false )
    $good [ ] - $found[$text];
}
$min - 1000000;
$max - -1000000;
function row_compare( $a, $b ) { return strcmp( $a['word'], $b['word'] ); }
usort( $good, 'row_compare' );
foreach( $good as $row )
{
  if ( $row['count'] < $min ) $min - $row['count'];
  if ( $row['count'] > $max ) $max - $row['count'];
}
$ratio = 10.0 / (float)( $max - $min );
?>
<html>
<head>
<style type-"text/css">
.word-link { line-height: 18pt; }
.title { border-bottom: 1px dotted black; margin-top: 5px; }
.snippet { margin-left: 20px; font-size:small; margin-top: 5px; margin-bottom:
5px; }
</style>
<script language-"Javascript">
var pages - [
<?php
foreach( $data as $row ) {
?>
```

```
{
url: '<?php echo( $row['URL'] ): ?>'.
snippet: '<?php echo( jsencode( $row['snippet'] ) ); ?>',
title: '<?php echo( jsencode( $row['title'] ) ):?>'
}.
<?php
}
?>
];
function display( items )
{
  var obj - document.getElementById( 'found' ):
  var html - "":
  for( i in items )
  {
    var p » pages[ items[ 1 ] ];
    html +- "<div class-\"title\"><a href-\""+p.url+"\" target-\"_blank\">"+p.
    html +« "<div class=\"snippet\">"+p.snippet+"</div>";
  }
  obj.innerHTML = html;
}
</script>
</head>
<body>
<table width="600" cellspacing="0" cellpadding-"5">
<tr>
<td colspan="2">
<form>
Search term: <input type="text" name="term" value="<?php echo($term); ?>" />
 :
<input type="submit" value="Search">
</form>
</td>
</tr>
<tr>
<td width="50%" valign="top">
<?php
foreach( $good as $row )
{
$val = (float)C $row['count'] - $min );
$fontsize - floor( 10.0 + ( $val * $ratio ) ):
$row_ids - array( );
foreach( $row['rows'] as $r ) { $row_ids []- $r['id']; }
$rows " join(',', $row_ids );
?>
<a class="word-link" href="javascript:display([<?php echo($rows); ?>]);"  ?></a>
style="font-size:<?php echo($fontsize); ?>pt:"><?php echo( $row['word'] ):
 
<?php } ?>
</td>
<td width="50%" id="found" valign="top">
</td>
</tr>
</table>
</body>
</html>
```

В этом сценарии используется комбинация из PHP и JavaScript. PHP использует модуль PEAR Services_Google (см. трюк 2) для загрузки нескольких результатов поиска. Затем из полученных в результате страниц удаляется HTML-разметка и текст разбивается на слова. Подсчитывается частота использования каждого слова, которая потом сохраняется вместе с URL-адресами относящихся к теме статей и описаниями, и все это при помощи массивов JavaScript на странице. После этого управление передается браузеру, который отображает найденные термины в левой части экрана. JavaScript обрабатывает щелчок кнопкой мыши пользователя на термине и задает внутреннюю разметку в свойстве innerHTML элемента FOUND, расположенном в правой части экрана и отображающем найденные статьи.

ПРИМЕЧАНИЕ

Вся работа происходит в функции display(), реализованной при помощи JavaScript.

Запуск трюка

Измените в файле значение переменной $key на полученное вами при регистрации в системе Google's Web API (http://www.google.com/apis/). Затем установите модуль PEAR Services_Google (см. трюк 2). На последнем шаге загрузите файл index.php на сервер и откройте его в вашем браузере. Результат должен выглядеть так, как на рис. 9.12.

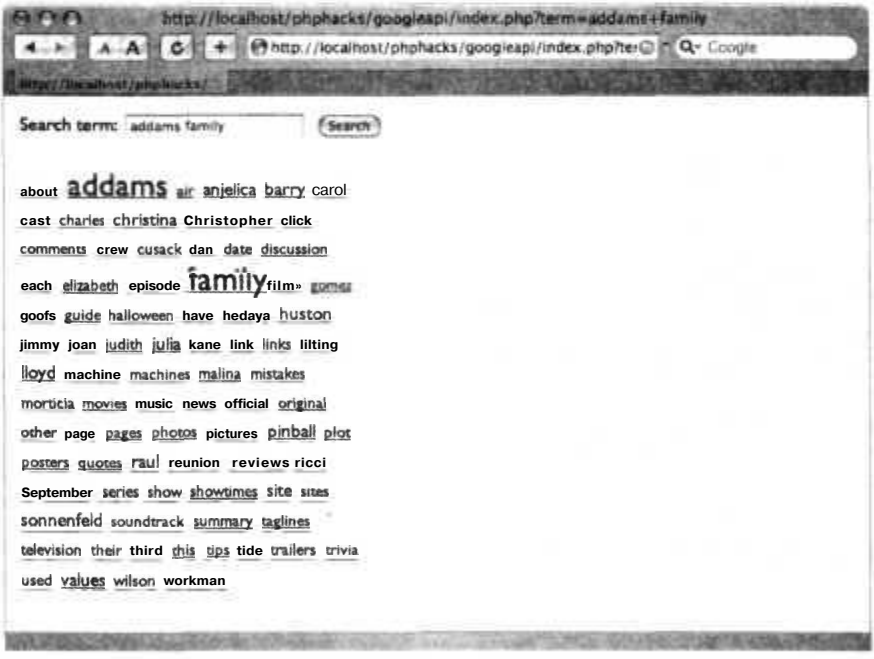


Рис. 9.12. Поиск словосочетания Addams Family

В левой колонке отображаются слова, которые встречались несколько раз в фрагментах страниц, полученных в результате каждого поиска. Как вы можете видеть, два наиболее часто встречающихся слова — Addams и Family, как и должно быть. Но есть и другие интересные слова, такие как имена других персонажей из шоу, а также слова review, cast и (что удивительно) goofs. Щелкнув кнопкой мыши на любом из этих элементов, вы увидите список страниц, в фрагментах которых встречалось это слово, как это показано на рис. 9.13. Я писал эту небольшую страницу для книги в качестве примера работы с Google Web Services API, но она оказалась намного полезнее. Визуализация в стиле диаграммы ссылок (см. трюк 24) может преподнести вам информацию в совсем другом свете.

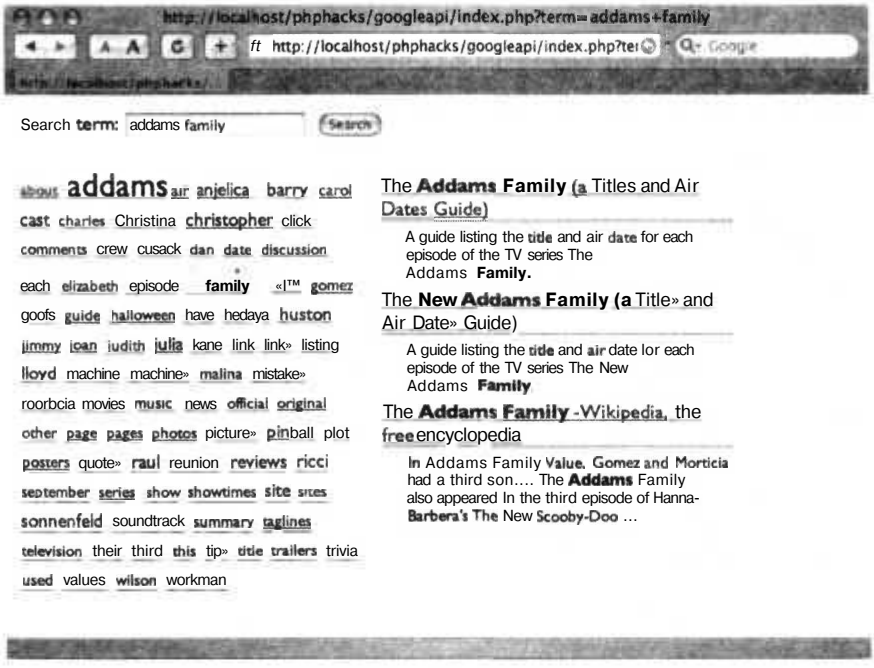


Рис. 9.13. Щелкнув кнопкой мыши на термине из отрывка, можно увидеть страницы, имеющие к нему отношение

Смотрите также

- «Создание диаграмм ссылок» (см. трюк 24).

ТРЮК №92

Создание нового интерфейса для Amazon.com

Используйте Web Services API для Amazon.com, чтобы создать новый поисковый механизм для книг.

Amazon.com стал одной из первых крупных компаний в зоне .com, которые начали предоставлять широкий доступ к своим интернет-сервисам. Его API, с одной стороны, очень обширный, а с другой стороны — очень простой в использовании. Для этого трюка я воспользуюсь API лишь при создании механизма поиска книги, который будет отображать результаты сразу двух поисков в двух колонках одновременно. Любая книга, которая будет найдена в результате двух поисков, будет помещена вверху страницы за пределами этих двух колонок. Теоретически таким образом вы получите книги, соответствующие заданным критериям поиска в наибольшей степени и затрагивающие обе интересующие вас темы.

Код

Сохраните исходный код из примера 9.11 в файл index.php.

Пример 9.11. HTML-интерфейс для сравнения двух результатов поиска книг на сайте Amazon

```
<?php
require_once 'PEAR.php';
require once 'Services/Amazon.php';
$devtokin - "XXXXXXX";
$userid - "USERID";
$amazon - &new Services_Amazon( $devtoken, $userid );
function list_products( $products )
{
?>
<table width="100%" cellspacing="0" cellpadding="3" border-"0">
<?php
foreach( $products as $product ) {
?>
<tr>
<td valign-"top" width="10%" valign="middle" align="center">
<a href-"<?php echo($product['url']); ?>" target="_new"><img src-"<?php
echo($product['imagesmall']); ?>" border»"0" /></a>
</td>
<td valign="top" width="90%">
<div class-"title"><a href-"<?php echo($product['url']); ?>" target="_new"><?php
echo( $product['name'] ); ?x/a></div>
<div class="author"><?php echo($product['creator']); ?x/div>
<div class-"date">Release Date: <?php echo($product['release']); ?></div>
<div class="price">Price:<?php echo($product['price']); ?></div>
</td>
</tr>
<?php } ?>
</table>
<?php
}

function find_books( $keyword )
{
  global $amazon;
  $out - array( );
  $products - $amazon->searchKeyword($keyword, "books", 1);
  if( $products !- null )
```

```
  {
    foreach($products as $product)
    {
      $creator - 'by ' . implode(', ', $product['authors']);
      $price - '';
      if( $product['listprice'] !- $product['ourprice'] )
        Sprice - '<strike>'.$product['listprice'].'</strike> '.
        $product['ourprice']:
      else
        Sprice = $product['listprice']:
      if ( strlen( $product['name'] ) > 0 )
        $out[ $product['asin'] ] - array(
          'url' => $product['url'],
          'imagesmall' -> $product['imagesmall'],
          'name' -> $product['name'],
          'release' -> $product['release'],
          'manufacturer' -> $product['manufacturer'],
          'asin' => $product['asin'],
          'creator' -> $creator.
          'price' -> Sprice
        ):
    }
  }
  return $out;
}
$terma - isset( $_GET['terma'] ) ? $_GET['terma'] : 'mysql';
$termb - isset( $_GET['termb'] ) ? $_GET['termb'] : 'php';
$lista - find_books( Sterma ):
$listb = find_books( $termb ):
$overlaps - array( ):
Sonlya = array( ):
Sonlyb » array( );
foreach( array_keys( Slista ) as $asin )
{
  if ( array_key_exists( Sasin, Slistb ) )
    Soverlaps[ Sasin ] = Slista[Sasin]:
  else
    $onlya[ Sasin ] - $lista[$asin];
}

foreach( array_keys( Slistb ) as Sasin )
{
  if ( !array_key_exists( Sasin, Slista ) )
    $onlyb[ Sasin ] = Slistb[$asin]:
}
?>
<html>
<head>
<title>Amazon Search Compare</title>
<style type="text/css">
th { border-bottom: 1px solid black: background: #eee: margin-bottom: 5px: fontsize:
small; }
td { font-size: small; }
.title { font-weight: bold: font-size: medium: }
.author { margin-left: 30px; font-style: italic; }
```

```
</style>
</head>
<body>
<form>
<div style="width:600px;">
<table width="100%" cellspacing="0" cellpadding="0" border="0">
<tr><th width="50%"><input type="text" name="terma" value="<?php echo( $terma ):
?>" /></th>
<th width="50%"><input type="text" name="termb" value="<?php echo( $termb ): ?>"
/><input type="submit" value="Go" /></th></tr>
</table>
<table width="100%" cellspacing="0" cellpadding="0" border="0">
<tr><td><?php list_products( array_values( $overlaps ) ): ?></td></tr>
</table>
<table width="100%" cellspacing="0" cellpadding="0" border="0">
<tr><td valign="top" width="50%"><?php list_products( array_values( $onlya ) ):
?></td>
<td valign="top" width="50%"><?php list_products( array_values( $onlyb ) ): ?></
td></tr>
</table>
</div>
</form>
</body>
</html>
```

Этот трюк использует модуль PEAR Services_Amazon для поиска по двум заданным терминам одновременно. Функция find_books() принимает в качестве параметра ключевое слово для поиска и возвращает все подходящие книги. Сценарий создает два таких списка $lista и $listb на основе результатов двух поисков. Затем формируются три новых списка: те книги, которые окажутся в обоих списках, помещаются в список $overlaps, а те, которые вошли только в список $lista или $listb, помещаются в список $onlya или $onlyb соответственно. В оставшейся части сценария эти списки форматируются при помощи HTML, используя стандартную методику работы с текстом в PHP.

Запуск трюка

Измените в коде значения переменных $devtoken и $userid на те, которые вы получите при регистрации в Amazon Web Services (http://amazon.com/webservices). Затем установите модуль PEAR ServicesAmazon (см. трюк 2), загрузите страницу index.php на ваш сайт и откройте ее в браузере. В результате вы должны увидеть страницу, похожую на изображенную на рис. 9.14. Вы можете изменить используемые для поиска термины, задав другие значения в заголовках колонок и нажав кнопку Go.

Смотрите также

- «Организация поиска в Google при помощи диаграмм ссылок» (см. трюк 91).
- «Отслеживание погоды» (см. трюк 100).

Рис. 9.14. Одновременный поиск с использованием PHP и MySQL

ТРЮК №93

Отправка SMS при помощи клиента для обмена мгновенными сообщениями

Используйте PHP для отправки SMS на мобильные телефоны при помощи программы-клиента для отправки сообщений через Jabber.

Одно из самых главных преимуществ обмена мгновенными сообщениями — это присутствие пользователя (и знание того, что он в сети). Конечно же, если пользователь недоступен, а вы очень сильно хотите пообщаться с ним, было бы круто отправить сообщение ему на мобильный телефон.

Многие последние модели телефонов поддерживают обмен мгновенными сообщениями, но SMS-сообщения поддерживают вообще все телефоны. Было бы

классно отправить сообщение на сотовый телефон пользователя, и в этом трюке будет рассказано, как это сделать при помощи шлюза, связывающего электронную почту и службу отправки SMS (который поддерживают практически все основные операторы мобильной связи).

Код

Сохраните исходный код из примера 9.12 в файл smsclient.php.

Пример 9.12. Программа-клиент на PHP для отправки SMS-сообщений

```
<?php
/* CONFIG VARIABLES */
// jabber-сервер, на котором вы регистрируетесь
$SERVER - 'your server';
//Имя и пароль для вашей учетной записи
$USERNAME - 'yourusername';
$PASSWORD - 'yourpassword';
// jabber id для вашей учетной записи
$PERSONAL- 'username@yourserver';
//эти значения могут измениться
global $cingular;
global $verizon;
global $nextel;
global $tmobile;
global $ATT;
$cingular - '@cingularME.com';
$verizon - '@vtext.com';
$nextel - '@messaging.nextel.com';
$tmobile - '@tmomail.com';
$ATT - '@mmode.com';
// Сохраним здесь любые числа и их носители
global $cell;
// Поместим любое десятичное число и носитель в соответствии
// с глобальной переменной выше как пару имя-значение
//например, "2125551234" -> "cingular"
$cell - array(
"10digitnumber" -> "carrier"
);
/* END CONFIG */

function send($to, $msg) {
  global $JABBER;
        $JABBER->SendMessage("$to","normal", NULL, array("body" ->
          htmlspecialchars($msg)),$payload);
}

// Заместим обработчик jabber.class.php
function Handler_message_normal($message) {
  global $JABBER;
        $body = $JABBER->GetInfoFromMessageBody($message);
  if (substr ($body ,0,3) - SMS) {
    $bodyparts - explode(":", $body);
    $tokenparts - $bodyparts[1];
    $tokens - explodeC ", $tokenparts, 3);
    $num = $tokens[0];
```

```
    $sub - $tokens[1];
    $bod = $tokens[2];
    sms($num, $sub, $bod);
  }
}

function sms($number, $subject, $body) {
  global $cingular;
  global $verizon;
  global $nextel;
  global $tmobile;
  global $ATT;
  global $cell;
  switch($cell[$number]) {
    case "cingular":
      Ssuffix = $cingular;
      break;
    case "verizon":
      Ssuffix - Sverizon;
      break;
    case "nextel":
      Ssuffix = $nextel;
      break;
    case "tmobile":
      Ssuffix - $tmobile;
      break;
    case "ATT":
      Ssuffix = SATT;
      break;
  }
  Saddress - $number.$suffix;
  mail($address, $subject, Sbody);
  echo $address.$subject.$body;
}

function Handler_message_chat($message) {
  Handler_message_normal($message);
}

require("class.jabber.php");
$JABBER = new Jabber;
SJABBER->server - $SERVER;
SJABBER->port - "5222";
$JABBER->username - $USERNAME;
SJABBER->password - $PASSWORD;
SJABBER->resource = "smsclient.php";
SJABBER->enable_logging - FALSE;
$JABBER->Connect( ) or die("Couldn't connect!");
$JABBER->SendAuth( ) or dieC'Couldn't authenticate!");
$JABBER->SubscriptionAcceptRequest($PERSONAL);
while(true) {
$JABBER->SendPresence(NULL, NULL, "online");
SJABBER->CruiseControl(15 * 60);
}
// могут никогда не понадобиться, но...
SJABBER->Disconnect( );
?>
```

Запуск трюка

Сперва вам понадобится немного настроить вашу учетную запись в системе Jabber, а затем загрузить модуль `class.jabber.php` (см. трюк 88) и поместить его в директорию, в которой находится трюк. Затем измените переменные для настройки в начале сценария, в которых хранится формат шлюзов для связи электронной почты и SMS, поддерживаемый большинством операторов мобильной связи. Эти переменные в исходном коде выделены полужирным шрифтом. Сохраните сценарий в файл `smsclient.php` и поместите его в директорию с модулем `class.jabber.php`.

ПРИМЕЧАНИЕ

Переменные заданы с учетом особенностей работы всех сетей мобильной связи на момент написания этой книги, но учитывайте, что возможны слияния операторов или другие изменения в непостоянном мире мобильной связи, что приведет и к изменению значений этих переменных. Возможно, вам придется получить обновленную информацию с сайта вашего оператора.

Надо отметить, что вам необходимо знать оператора мобильной связи. В свете последних законов, позволяющих пользователям сохранять свой номер при смене мобильного оператора, приходится быть предельно внимательным. Вы можете отправить сообщение сразу всем операторам, но это не очень изящное решение проблемы, а нам не следует забывать, что это трюк!

Добавьте учетную запись в системе Jabber, которую вы используете для связи со сценарием, в список контактов вашей личной учетной записи. Запустите сценарий из командной строки:

```
php smsclient.php &
```

Сообщение, которое вы отправляете на мобильный телефон, должно иметь следующий формат:

```
SMS:10digitnumberSubject Body
```

Просто введите ваше сообщение и наслаждайтесь результатом!

Смотрите также

- «Передача данных от RSS-источников в ваше приложение для отправки мгновенных сообщений при помощи Jabber» (см. трюк 88).
- «Использование IRC в ваших веб-приложениях» (см. трюк 89).

ТРЮК №94

Создание флэш-роликов на лету

Используйте Ming для динамического создания флэш-роликов на PHP.

Вам когда-нибудь хотелось создавать веб-графику, которая бы была более изящной и живой? Все мы хотели этого. Один из способов — использовать формат

Macromedia Flash (также известный как SWF-формат). Но как же им воспользоваться при помощи только средств с открытым кодом? Для этого есть модуль для PHP под названием Ming, который и спасает положение. Он позволяет вам формировать на лету полноценные файлы с флэш-роликами. В этом трюке будет рассказано, как успешно справиться с задачей создания приложения, динамически формирующего диаграммы.

Код

Сохраните исходный код из примера 9.13 в файл data.php.

Пример 9.13. Небольшой код на XML для отрисовки во Flash

```
<?php
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT"); //Старая дата
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT"); //Всегда изменяется
header("Cache-Control: no-store, no-cache, must-revalidate"); // HTTP/1.1
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache"); // HTTP/1.0
header('Content-type: application/xml');
echo("<?xml version=\"1.0\" ?>\n");
?>
<GRAPH TYPE="BAR">
<TITLE>Revenues 2005</TITLE>
<YAXIS>Dollars
<RANGE MIN="0" MAX="50000" />
</YAXIS>
<XAXIS>Period
</XAXIS>
<DATA>
<?php
$colors = array( "0xFF0000", "0xFFFF00", "0xFF00FF", "0x00FFFF", "0x00FF00" );
srand((double)microtime( )*1000000);
for ($i = 1; $i < 7; $i++)
{
  $clr = $colors[ ($i - 1) % count($colors) ] ;
  $val = rand(10000,45000);
  echo("<D$i>$val<COLOR C-\"$clr\" /></D$i>\n");
}
/*
<D1>20000<COLOR C="0xFF0000" /></D1>
<D2>25000<COLOR C="0xFFFF00" /></D2>
<D3>27000<COLOR C="0xFF00FF" /></03>
<D4>42000<COLOR C="0x00FFFF" /></D4>
<D5>48000<COLOR C="0x00FF00" /></D5>
*/
?>
</DATA>
</GRAPH>
```

Сценарий graph.php из примера 9.14 делает всю основную работу (с огромной помощью Ming).

Пример 9.14. Ming приходит на помощь PHP

```
<?
ming_useswfversion(6); // Важно!
```

```
$m - new SWFMovie( ):
$m->setBackground(0x80, 0x80. 0x80):
$m->setDimension(320, 240);
$m->setRate(30.0);
$s - new SWFShape( ):
$f - $s->addFill(0xff, 0xff. 0xff):
$s->setRightFill($f);
$s->movePenTo (-5. 0);
$s->drawLineTo( 5, 0);
$s->drawLineTo( 5. -10);
$s->drawLineTo(-5. -10):
$s->drawLineTo(-5, 0):
$p - new SWFSprite( ):
$i = $p->add($s);
$i->setDepth(1);
$p->nextFrame( ):
$i = $m->add($p);
$i->setDepth(1);
$i->moveTo(-10, -10):
$i->setName("box");
$m->add(new SWFAction("
var data - []:
var heights - [];
var actual - []:
var timerID;
// Анимировать панель
function anim ( )
{
  var done - true:
  for (var k - 0: k < data.length; k++)
  {
    var n - 'D' + k;
    if (heights[k] !- actual[k])
      done - false;
    else
      continue:
    var diff - (heights[k] - actual[k]) / 5:
    actual[k] +- diff:
    _root[n]._height - actual[k];
    Tf (diff < 0.1)
      actual[k] - heights[k]:
  }
  if (done)
  {
    clearInterval(timerID);
  } stop( ):
};

// Данные в XML преобразуем в JavaScript/ActionScript
// объекты и отобразим их
function doit(o)
{
  // Берем данные и отрисовываем
  // Рисуем оси
  createEmptyMovieClip ('grp', 1);
```

```
with (grp)
{
  lineStyle (1. 0xFFFFFF. 100):
  // X-ось
  moveTo (40. 200):
  lineTo (280. 200):
  // Y-ось
  moveTo (40. 200);
  lineTo (40. 20):
  // Рисуем заголовок
  createTextField('title'. 1. 100. 00. 100. 100):
  var fmt - new TextFormat( ):
  fmt.color = 0xffffff:
  fmt.font = 'Arial';
  title.text = o['TITLE']['_txt'];
  title.setTextFormat(fmt);
  // Рисуем название X-оси
  createTextField('xtitle'. 2. 120. 220. 100. 100):
  var fmt = new TextFormat( ):
  fmt.color - 0xffffff:
  fmt.font - 'Arial';
  fmt.size = '10';
  xtitle.text = o['XAXIS']['_txt'];
  xtitle.setTextFormat(fmt);
  // Рисуем название Y-оси
  createTextField('ytitle'. 3. 0. 100. 100. 100);
  var fmt - new TextFormat( );
  fmt.color = 0xffffff;
  fmt.font = 'Arial';
  fmt.size = '10';
  ytitle.text - o['YAXIS']['_txt'];
  ytitle.setTextFormat(fmt);
  // Рисуем метки Y-оси
  createTextField('ylabeltop'. 5. 20. 16. 20. 20):
  var fmt = new TextFormat( ):
  fmt.color - 0xffffff:
  fmt.font = 'Arial';
  fmt.size - '6';
  fmt.align - 'right':
  ylabeltop.text - o['YAXIS']['RANGE']['1'];
  ylabeltop.setTextFormat(fmt);
  createTextField('ylabelbot'. 6. 20. 193. 20. 20):
  var fmt = new TextFormat( );
  fmt.color - 0xffffff:
  fmt.font = 'Arial';
  fmt.size - '6':
  fmt.align = 'right';
  ylabelbot.text - '0';
  ylabelbot.setTextFormat(fmt);
}: // Конец with(grp)
// Рисуем данные
// Определяем, как много элементов данных у нас есть
for (var k - 0: k < 10; k++)
{
  if (typeof(o['DATA']['D'+k]) == 'object')
    data.push(o['DATA']['D'+k]);
```

```
  }
  // Рисуем данные
  // Просматриваем каждый элемент данных и позицию ролика
  var increment - 180 / data.length;
  var width = increment / 2;
  var max = Number(o['YAXIS']['RANGE']['1']);
  for (var k = 0; k < data.length; k++)
  {
    // дублируем box в новый столбец
    var n - 'D' + k;
    duplicateMovieClipCbox. n. 10+k);
    // перемещаемся в конечную позицию
    _root[n]._x = 40 + ((k + 1) * increment);
    _root[n]._y - 199.5;
    _root[n]._width = width:
    // Задаем высоту, равную нулю
    _root[n]._height = 0;
    // Получаем значение и преобразуем в viewport
    var n2 = 'D' + (k+1);
    var h = 180 / max * Number(o['DATA'][n2]['_txt']);
    heights.push(h);
    actual.push(0);
    var c = new Color(_root[n]);
    c.setRGB(Number(o['DATA'][n2]['COLOR']['1']));
  }
  // Все, обновляем таймер анимации
  // ролик меняется каждые 32 мс до тех пор, пока не достигнет
  // окончательного значения.
  timerID = setInterval(anim, 32);
}

function convertToJS(nodes. o)
{
  if (arguments.length — 1)
    o — {};
  for (var i = 0; i < nodes.length; i++)
  {
    if (nodes[i].nodeType == 1)
    {
      var tmp = convertToJS(nodes[i].childNodes):
      // Добавляем атрибуты
      var attribs = nodes[i].attributes;
      for (var j in attribs)
        tmp[i] = attribs[j];
        var name - nodes[i].nodeName;
        o[name] = tmp:
    }
    else
    {
      var v = nodes[i].nodeValue;
      o._txt — v:
    }
  }
  return o:
}
var xml = new XML( ) ;
```

```
xml.ignoreWhite = true; // Иначе создается много пустых узлов
xml.onLoad = function(success)
{
  if (!success)
  {
    // Выполните Javascript-оповещение через url
    return;
  }
  var o = convertToJS(this.childNodes[0].childNodes);
  doit(o);
};
xml.load('data.php');
$m->nextFrame( );
header('Content-type: application/x-shockwave-flash');
$m->output( );
?>
```

Процесс, который я использовал здесь, проиллюстрирован на рис. 9.15. Файл graph.php вызывается браузером для отрисовки. Затем сценарий использует библиотеку Ming, чтобы создать ролик с простейшими спрайтами, невидимыми на экране. В качестве завершающей фазы сценарий вызовет ActionScript для загрузки XML-файла из URL. Этот XML будет динамически сгенерирован PHP в сценарии data.php. В нашем случае data.php генерирует случайные данные для диаграммы. Когда данные загружены и обработаны флэш-проигрывателем, столбцы в диаграмме будут анимированы и отразят свои окончательные значения.

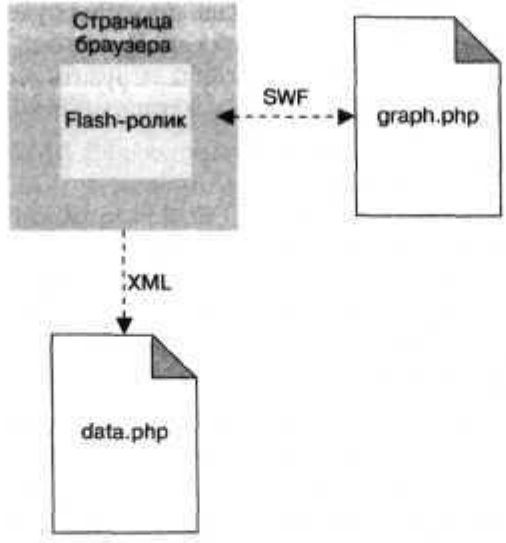


Рис. 9.15. Взаимодействие между проигрывателем и **PHP-сценариями**

Очень хорошо, что библиотека Ming для PHP является объектно-ориентированной, поэтому в процессе работы она использует классы. Это позволяет видеть, какие методы поддерживаются каждым объектом. Самый важный класс называется

SWFMovie. Он будет отрисовывать элементы в SWF-файле при помощи функции output O. Однако, чтобы что-то увидеть, вам необходимо добавить в ролик объекты. Я создал простой спрайт в виде белого прямоугольника, который будет помещен вне пределов видимости экрана и будет использоваться в качестве прототипа для создания столбиков в диаграмме. И, наконец, добавляется объект ActionScript (SWFAction).

Код ActionScript во Flash-объекте выполняет большой объем работы и демонстрирует, что вы можете делать в флэш-приложениях.

ПРИМЕЧАНИЕ

Технически вы можете создать флэш-приложение, не используя SWFAction, но это будет крайне нецелесообразно.

В коде определяются несколько функций, которые будут использоваться в дальнейшем при отрисовке. И, наконец, сценарий создает XML-объект и занимается самой интересной работой, получая через Сеть XML-файл (трюк не будет крутым, если вы не работаете с Сетью).

ПРИМЕЧАНИЕ

Помните, что XML-документ должен быть получен с того же сервера, с которого получается и SWF-файл (из соображений безопасности).

Поскольку на получение XML-файла может потребоваться некоторое время, этот процесс асинхронный. В другом случае флэш-анимация будет приостанавливаться, ожидая, пока ActionScript загрузит файл из Интернета. Когда файл считан, XML-объект вызывает функцию onLoad(), чтобы загрузить данные из data.php. При этом запускается цепочка событий, которые и отрисовывают данные.

Был написан простой PHP-сценарий, формирующий XML-файл с диаграммой для получения XML-объекта. PHP может отображать XML так же просто, как и HTML. Однако есть некоторые особенности. Заметьте, что в заголовке в примере 9.13 произошли некоторые изменения. Без этих заголовков объект Flash XML будет активно кэшировать данные (и, как назло, именно в тот момент, когда они будут меняться). Кроме того, стоит обратить внимание на оператор echo с данными об XML. Если вы не добавите эту строку, а, наоборот, уберете ее, то PHP откажется запускать ваш сценарий.

После загрузки XML-файла и его анализа XML DOM трансформируется в объекты ActionScript с иерархической структурой. Благодаря этому нам намного проще получить доступ ко всем данным, использующим родные для ActionScript типы. После этого при помощи функции doit() отображается начальная диаграмма. В ней создается новый ролик, и в него добавляются координатные оси и надписи. Затем перебираются все данные, полученные из XML-файла, и в диаграмме формируются отдельные полосы. Для этого создаются копии спрайта box, которым присваиваются имена и глубина (то есть порядок размещения по координате z). Затем эти объекты занимают свои места в диаграмме. Здесь для этих прямоуголь-

ников задается новая ширина, высота (которая изначально равна нулю) и цвет. И, наконец, в функции doitO устанавливается таймер для вызова функции anim() каждые 32 мс. Таким образом, размеры полос в диаграмме как бы вырастут до своих реальных значений. В anim() используется простая функция для увеличения размеров с постепенно уменьшающейся скоростью, что создает эффект аккуратного плавного роста высоты столбцов в графе. Когда они достигнут своих окончательных размеров, функция anim() убирает таймер и отрисовка диаграммы завершается.

Очень неплохо для 5 Кбайт исходного кода, не так ли?

Запуск трюка

Чтобы запустить этот трюк, вам необходимо установить Ming. Официальный сайт — http://ming.sourceforge.net/. Там вы можете найти исходные коды, из которых создадите библиотеку для вашей платформы PHP (она очень проста в настройке и установке).

ПРИМЕЧАНИЕ

Если вы используете Windows, то можете получить уже готовые DLL для дистрибутива PHP по адресу http://kromann.info/php.php.

После того как вы установили Ming, все, что вам надо, — это загрузить ваш любимый текстовый редактор поверх окна браузера (рис. 9.16).

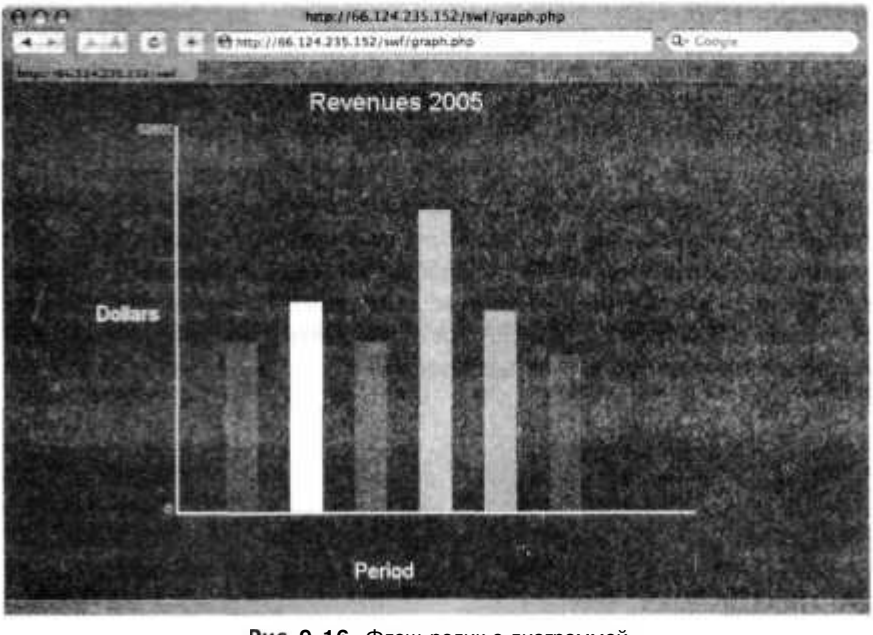


Рис. 9.16. Флэш-ролик с диаграммой

Улучшаем трюк

Самое тяжелое при создании приложения, использующего PHP-Ming-Flash, — получить в результате на экране то, что вы и задумывали. Флэш-проигрыватель очень чувствителен к ошибкам, и в результате этого он очень изящно вылетает, если вы неправильно воспользуетесь какими-либо операторами. Это, конечно же, хорошо для конечного пользователя, но в итоге программистам (таким как вы!) только и остается, что чесать затылок и размышлять, когда же появится ожидаемый вами вращающийся треугольник. Изучив этот пример, вы получите базовые знания о приложениях, использующих флэш, и это будет отличной отправной точкой для создания новых трюков.

В качестве отличного дополнения к трюку можно было бы использовать плавное изменение фонового цвета для столбцов. После этого запросто можно бы было добавить небольшие изменения, после которых ваше приложение выглядело бы трехмерным. Вы также можете добавить такие стандартные для диаграмм возможности, как использование линии координатной сетки или отметки на осях, или даже таблицу условных обозначений. Флэш обладает богатыми возможностями по работе с мультимедиа, поддерживая различные форматы аудио и видео. Вы можете воспользоваться и этим. Вы также могли бы анимировать внешний вид диаграммы при ее первом отображении. Если же вы действительно хотите улучшить трюк, то можете добавить поддержу восьмой версии флэш, позволяющей использовать фильтры с различными эффектами в режиме реального времени (например, расплывчатость). Возможности по работе с графикой во флэше просто огромны, в Интернете есть множество примеров, которые могут вдохновить вас на создание чего-то нового.

На этом мы не заканчиваем, так как я рассказал только о визуальной части трюка. Я просто отобразил на экране случайные данные. При помощи PHP можно запросто вывести на экран какие-либо настоящие данные из базы, используя XML-файл. Еще было бы неплохо добавить таймер для периодической перерисовки диаграммы. Вы также могли бы передавать в PHP-сценарий параметры, на основе которых будет формироваться диаграмма. Если вы пойдете по этому пути, то можете не использовать XML-файл, а просто запрашивать данные, которые следует отобразить. Если вы добавите обработку событий графического интерфейса (кнопок, мыши, клавиатуры), то можете даже передавать различные параметры в XML через URL-адрес и динамически отображать данные пользователю. Есть много, очень много различных возможностей.

Забавные возможности

Трюки 95-100

PHP используется не только для создания расчетных приложений. В этой главе мы рассмотрим трюки, которые раскрывают «веселую» сторону PHP, — от создания ваших собственных карт Google до MP3-сервера и загрузки Википедии на вашу портативную приставку PlayStation (PSP).

ТРЮК №95

Создание пользовательских карт Google

Используйте Google Maps API для встраивания динамических карт в ваши приложения с пользовательской разметкой, слоями и интерактивностью.

В те редкие промежутки времени, когда я свободен, я люблю совершать походы в окрестности Вермонта в Калифорнии. К счастью, лучшая для походов местность находится недалеко. В частности, больше всего мне нравится совершать прогулки к пику Миссия из-за его живописных пейзажей и хорошей нагрузки в пути. Чтобы показать мои походы к пику, я всегда использовал страницы Вики, размещая на них множество фотографий. Но я всегда хотел чего-то более интерактивного, и с появлением карт Google и обширного API я смог воспользоваться комбинацией PHP и JavaScript, чтобы рассказать поподробнее о моих походах в горы, используя спутниковые съемки и интерактивные пометки (вот так и происходит слияние современных технологий и природы!).

Код

Для начала сохраните исходный код из примера 10.1 в файл `index.php`.

Пример 10.1. Установка широты и долготы при использовании карт Google

```
<?php
$images = array(
  array( Mat' => -121.9033, 'lon' => 37.5029, 'img' => "mpO.jpg" ),
  array( 'lat' => -121.8949, 'lon' => 37.5050, 'img' => "mp1.jpg" ),
  array( 'lat' => -121.8889, 'lon' => 37.5060, 'img' => "mp2.jpg" ),
  array( 'laf => -121.8855, 'lon' => 37.5076, 'img' -> "mp3.jpg" ),
  array( 'lat' -> -121.8835, 'lon' => 37.5115, 'img' => "mp4.jpg" ),
  array( 'lat' => -121.8805, 'lon' => 37.5120, 'img' => "mp5.jpg" )
);
?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Simple Google Maps Page</title>
<script src="http://maps.google.com/maps?file=api&v=1&key=<mapskey>" type="text/
javascript"></script>
</head>
<body>
<table>
<tr>
<td valign="top">
<div id="map" style="width: 300px; height: 300px"></div>
</td>
<td valign="top">
<img src="mp0.jpg" id="mpimg">
</td>
</tr>
</table>
<script type="text/javascript">
var mp_images = [
<?php $first = true: foreach( $images as $img ) { ?>
<?php if ( $first — false ) { echo( '.' ): } ?>
{ lat: <?php echo( $img['lat'] ) ?>,
lon: <?php echo( $img['lon'] ) ?>.
img: "<?php echo( $img['img'] ) ?>" }.
<?php $first - false; } ?>
];
var map - new GMap(document.getElementById("map")):
map.addControKnew GSmallMapControl( ));
map.centerAndZoom(new GPoint(-121.8858, 37.5088), 4):
map.setMapType( G_SATELLITE_TYPE ):
var icon = new GIcon( );
icon.shadow = "http://www.google.com/mapfiles/shadow50.png";
icon.iconSize = new GSize(20. 34);
icon.shadowSize = new GSize(37. 34):
icon.iconAnchor - new GPoint(9, 34);
icon.infoWindowAnchor = new GPoint(9. 2);
icon.infoShadowAnchor = new GPoint(18. 25);
icon.image = "http://www.google.com/mapfiles/marker.png":
var markers = {}:
for( i in mp_images )
{
  markers[i] - new GMarker( new GPoint( mp_images[i].lat, mp_images[i].lon ),
   icon );
  GEvent.addListener(markers[i]. "click", function( ) {
    for( m in markers ) {
      if ( markers[m] == this ) {
        document.getElementById( "mpimg" ).src - mp_images[m].img;
      }
    }
  } );
  map.addOverlay(markers[i]);
}
</script>
</body>
</html>
```

Запуск трюка

Прежде чем запустить этот трюк, вам необходимо получить ваш собственный ключ для Google Maps API. Просто посетите сайт Google Maps http://google.com/**apis/maps** (рис. 10.1).

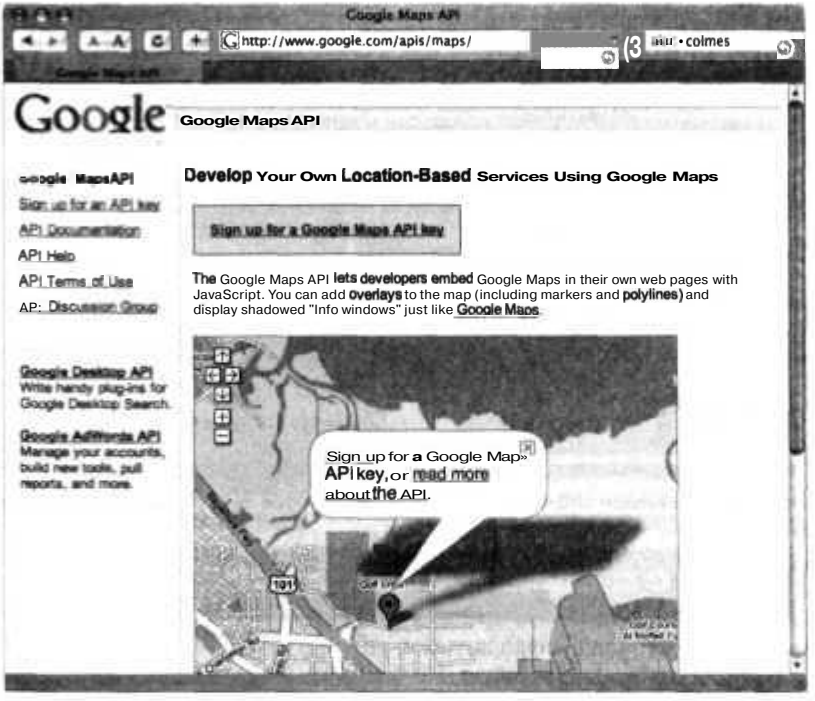


Рис. 10.1. Домашняя страница Google Maps API

На этом сайте выберите ссылку Sign up for **a** Google Maps API key. Щелкнув на ней кнопкой мыши, вы перейдете к странице с лицензионным соглашением, где можете подтвердить, что ознакомились с ним и согласны со всеми условиями. Вам также придется указать URL-адрес вашего сайта, на котором вы будете размещать карты. Эту страницу вы можете видеть на рис. 10.2.

Вам придется также указать директорию. Если адрес страницы для карт будет выглядеть как **http://www.mysite.com/maps/mymap.php,** то вам следует указать директорию **http://www.mysite.com/maps/.** Когда вы нажмете кнопку Generate API Key, вас могут попросить авторизоваться. Если вы все еще не создали учетной записи в системе Google, вам придется это сделать. После регистрации вы перейдете к странице, на которой будет расположен ваш ключ для карт. Эта страница показана на рис. 10.3. В данном случае ключ не виден (у вас есть свой собственный!). В верхнем поле находится ключ API, который вам нужно будет вставить в исходный код трюка вместо слова `<mapskey>`, как это показано в примере 10.1 (в выделенной части кода).

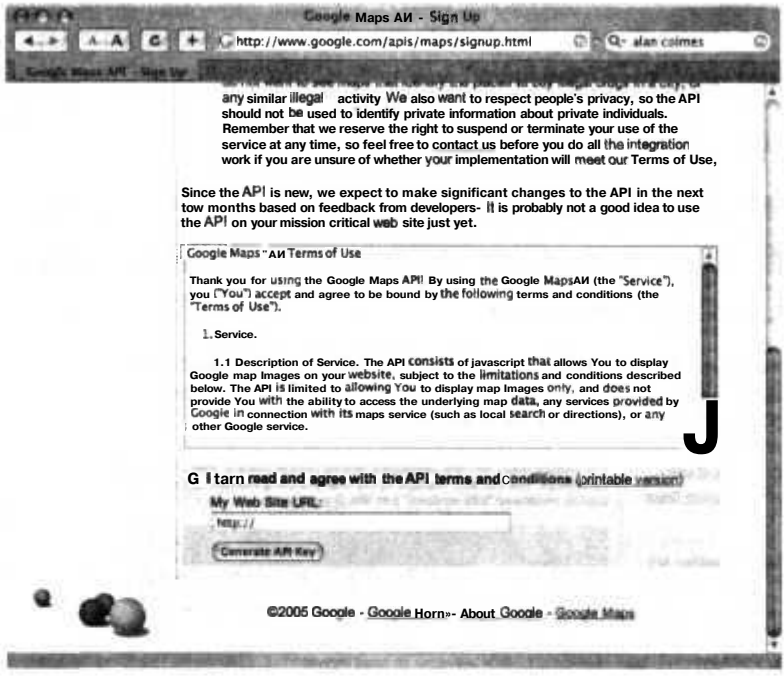


Рис. 10.2. Указываем URL-адрес страницы, на которой будут размещаться карты

Рис. 10.3. Ключ API и фрагмент кода для указанного URL-адреса

ПРИМЕЧАНИЕ

Google также предоставляет небольшой пример кода внизу страницы, на которой находится ключ API, чтобы помочь вам начать работу. Ну разве Google не крут?

После того как вы внесете изменения в исходный код трюка, добавив туда ключ Google, загрузите его и изображения на сервер. Затем откройте вашу страницу на PHP в браузере (как это выглядит у меня, вы можете увидеть на рис. 10.4).

Рис. 10.4. Домашняя страница карты пика Миссия

На странице в ее левой части видна карта, а также изображение из базы с фотографией гор. Когда вы выберете одну из пометок слева, изображение в правой части страницы изменится на соответствующую этой метке фотографию. На рис. 10.5 я выбрал маркер, расположенный на полпути к горе.

Рис. 10.5. Изображение после щелчка кнопкой мыши на одном из маркеров

Этот пример демонстрирует, как на самом деле просто создать действительно интерактивную карту для вашего приложения на PHP и добавить в нее возможность взаимодействия с посетителями. Я встречал сайты, на которых назначаются свидания, для чего используются карты, и даже сайты, на которых графически при помощи слоев можно просмотреть, какова будет зона поражения при взрывах ядерных устройств различной мощности в заданных вам точках на карте! Google, карты (см. трюк 86) и PHP уже повсюду!

Смотрите также

- «Выясните, откуда пришли ваши посетители» (см. трюк 63).
- «Создание пользовательских карт при помощи MapServer» (см. трюк 86).

ТРЮК №96

Создание динамических списков воспроизведения

Используйте XML Simple Playlist Format (XSPF) для создания списков воспроизведения при помощи PHP.

На PHP можно легко создавать для вас и ваших друзей приложения, проигрывающие MP3. Доступен новый стандарт для списков воспроизведения под названием XSPF (http://www.xspf.org/), и он может быть использован в проигрывателе Flash MP3 (http://musicplayer.sf.net). И, конечно же, с ним может работать PHP. На рис. 10.6 показано, как плагин Flash-видео запрашивает XML со списком воспроизведения XSPF из сценария `playlist.php`.

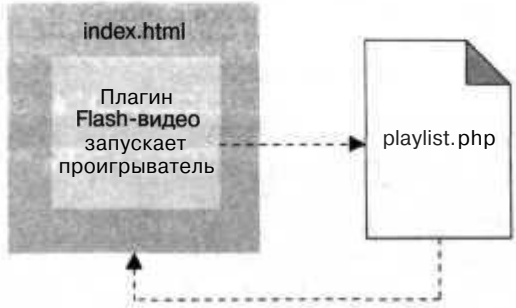


Рис. 10.6. Плагин Flash-видео, запрашивающий список воспроизведения из PHP-сценария

Исходный код

Сохраните исходный код из примера 10.2 в файл `index.html`.

Пример 10.2. Начало настройки MP3-проигрывателя

```
<html>
<body>
```

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.
cab#version=7,0,0,0"
width="400" height="153" id="xspf_player" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="http://localhost/xspf/xspf_player.
swf?autoload=true&playlist_url=http://localhost/xspf/playlist.php" />
<param name="quality" value="high" />
<param name="bgcolor" value="#e6e6e6" />
<embed src="http://localhost/xspf/xspf_player.swf?autoload=true&playlist_
url=http://localhost/xspf/playlist.php"
quality="high" bgcolor="#e6e6e6" width="400" height="153"
name="xspf_player" align="middle" allowScriptAccess="sameDomain"
type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
</body>
</html>
```

Теперь сохраните исходный код из примера 10.3 в файл playlist.php.

Пример 10.3. Дополнительный код на PHP для создания в XML списка воспроизведения

```
<? echo( "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>" ) ?>
<playlist version="1" xmlns="http://xspf.org/ns/0/">
<title>MyRadio</title>
<trackList>
<?
$dir = opendir( "." );
while ($file = readdir($dir)) {
if ( preg_match( '/[.]mp3$/i". $file ) ) {
?>
<track>
<location>http://localhost/xspf/<? print($file)?></location>
<annotationx? print( $file ) ?></annotation>
</track>
<?}}?>
</trackList>
</playlist>
```

Запуск трюка

Загрузите файлы с PHP и HTML-кодом на сервер, а также поместите в ту же директорию XSPF Flash-видео с сайта проигрывателя (http://musicplayer.sf.net/) и любые MP3-файлы. Затем откройте страницу в браузере (рис. 10.7).

ПРИМЕЧАНИЕ

В этом примере в качестве общей директории использовалась XSPF. Если вы измените ее имя, вам также придется изменить и ссылки в файле index.html.

Файлы, которые вы увидите в списке воспроизведения проигрывателя, — это те самые MP3-файлы, которые вы поместили в ту же папку, что и PHP-сценарий. Сценарий просматривает содержимое директории, находит в ней MP3-файлы и добавляет их в список воспроизведения.

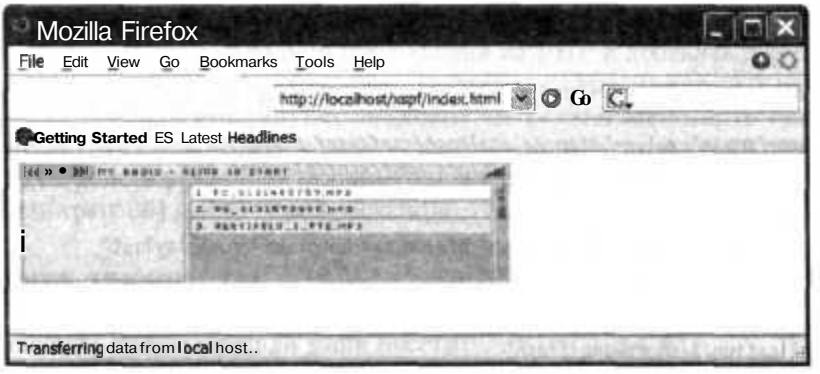


Рис. 10.7. Flash MP3-проигрыватель с отображенным в нем списком воспроизведения

Этот сценарий очень полезен, если вы размещаете в Интернете оцифрованные записи и хотите, чтобы во всех ваших блогах был Flash-проигрыватель. В качестве примера на рис. 10.8 показан проигрыватель, который используется на сайте артиста Джошуа Армстронга (http://joshuaarmstrong.net/).

Рис. 10.8. Flash-проигрыватель, используемый на сайте Джошуа Армстронга

Проигрыватель интегрирован в домашнюю страницу в правом нижнем углу поверх фонового рисунка. Перед нами пример ресурса, построенного по схеме «все в одном»: фотография артиста, раздел, где можно купить его альбом, а также музыка из альбома — и все это доступно с домашней страницы.

Смотрите также

- «Создание медиацентра загрузок и выгрузок» (см. трюк 97).

ТРЮК №97

Создание медиацентра загрузок и выгрузок

Позвольте пользователям загружать и выгружать медиафайлы через ваше приложение.

Когда-нибудь посетители вашего сайта захотят обмениваться чем-то большим, чем просто текстовые сообщения. Они захотят получить доступ к медиафайлам, изображениям и т. д. Оставим пока в стороне легальность такого рода обмена файлами. В этом трюке я задался целью рассказать вам, как создать простой медиацентр загрузок и выгрузок для вашего сайта. На рис. 10.9 отображено взаимодействие страниц в такого рода системе обмена файлами. Для начала пользователь загружает страницу `index.php`, через которую он отправляет на сервер файлы при помощи сценария `upload.php`. Затем этот сценарий направляет его на страницу `dir.php`, отображающую доступные для загрузки с сервера файлы. Щелкнув кнопкой мыши на любом из файлов, вы загрузите его себе на компьютер при помощи сценария `download.php`.

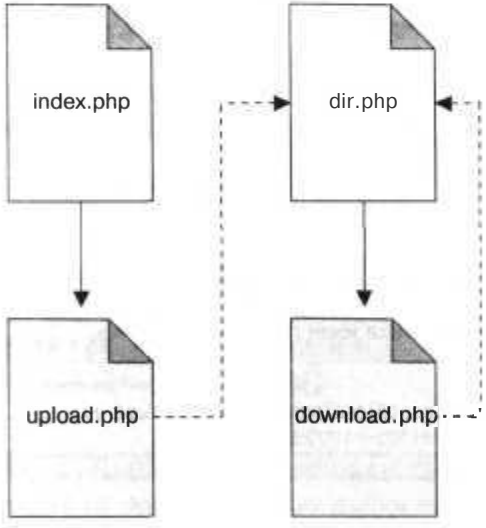


Рис. 10.9. Взаимодействие между страницами медиацентра загрузок и выгрузок файлов

Код

Сохраните исходный код из примера 10.4 в файл с именем `media.sql`. Это всего лишь небольшой набор SQL-команд для создания новой таблицы.

Пример 10.4. Создание таблицы для учета загруженных медиафайлов

```
DROP TABLE IF EXISTS media:
CREATE TABLE media (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  filename TEXT,
  mime_type TEXT.
  PRIMARY KEY( id )
);
```

Сохраните HTML-код из примера 10.5 в файл index.php.

Пример 10.5. Форма для загрузки файлов на сервер

```
<html>
<body>
<form enctype="multipart/form-data" action="upload.php" method="post">
  <input type="hidden" name="MAX_FILE_SIZE" value="2000000" />
  <input type="file" name="file" />
  <input type="submit" value="Upload" />
</form>
</body>
</html>
```

PHP-код в примере 10.6 управляет загрузками; сохраните его в файл download.php.

Пример 10.6. Соединение с базой данных и установка разрешения на закачку файла

```
<?php
require_once( "db.php" );
$db -& DB::connect("mysql://root@localhost/media", array( ));
if (PEAR::isError($db)) { die($db->getMessage( )); }
$res = $db->query( "SELECT filename, mime_type FROM media WHERE id = ?", array(
$_GET['id'] ) );
$res->fetchInto($row);
$filename - $row[0];
$type - $row[1];
Sdatafile - "media/".$_GET['id'].".dat";
header( "Content-type: $type" );
header( "Content-Length: ".@filesize( Sdatafile ) );
header( 'Content-Disposition: attachment: filename="'.$filename."" );
readfile( Sdatafile);
?>
```

PHP-файл dir.php, созданный из примера 10.7, управляет списком файлов.

Пример 10.7. Перечисление списка имен доступных файлов

```
<?php
require_once( "db.php" );
$db =& DB::connect("mysql://root@localhost/media", array( ));
if (PEAR::isError($db)) { die($db->getMessage( )); }
?>
<html>
<body>
<?php
Sres = $db->query( "SELECT * FROM media" );
while (Sres->fetchInto(Srow)) { ?>
<a href="download.php?id=<?php echo( $row[0] ); ?>">
<?php echo( $row[1] ); ?></a><br/>
<?php } ?>
</body>
</html>
```

Исходный код в примере 10.8 — наиболее сложный сценарий в этом трюке. Он управляет загрузкой файлов на сервер. Сохраните его в файл с именем upload.php.

Пример 10.8. Загрузка файла в базу данных

```
<?php
require_once( "db.php" );
$db =& DB::connect("mysql://root@localhost/media", array( ));
if (PEAR::isError($db)) { die($db->getMessage( )); }
if ( $_FILES['file']['tmp_name'] )
{
  $sth - $db->prepare( "INSERT INTO media VALUES ( 0. ?, ? )" );
  $db->execute( $sth. array( $_FILES['file']['name'], $_
  FILES['file']['type'] ) );
  $res = $db->query( "SELECT last_insert_id( )" );
  $res->fetchInto( $row );
  $newid = $row[0];
  move_uploaded_file( $_FILES['file']['tmp_name'], "media/".$newid.".dat"
  );
}
header( "location: dir.php" );
?>
```

Запуск трюка

Загрузите все эти файлы на сервер. Воспользуйтесь командой mysql, чтобы загрузить схему базы данных под названием media:

```
% mysql --user=myusername --password=mypassword media < media. sql
```

После того как база данных будет готова, вам надо будет создать директорию с именем media в той же директории, в которой вы разместили сценарий и которая будет использоваться для хранения загруженных на сервер файлов. Затем откройте в браузере **страницу** index.php (рис. 10.10). Нажмите кнопку Browse и выберите какой-нибудь **файл.** Достаточно будет небольшого JPEG-файла. При помощи поиска изображений в Google я нашел фотографию рыбы, которую и использовал для проверки.

Рис. 10.10. Форма для загрузки файлов на сервер

Нажмите кнопку Upload (Загрузить), и вы получите страницу, похожую на изображенную на рис. 10.11.

При помощи этого списка вы можете узнать, что файл загружен на сервер без ошибок и база данных, в которую был добавлен новый файл, обновлена. Файл скопирован в директорию для мультимедиа, и ему присвоено имя <id>.dat, где <id> — это

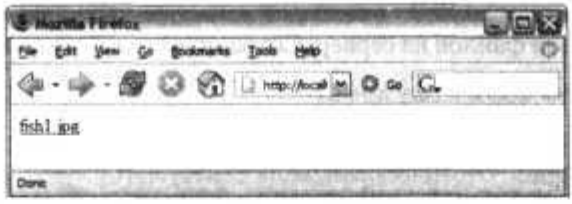


Рис. 10.11. Открыта папка, в которой хранятся загруженные на сервер **медиафайлы**

номер ID записи в базе данных. Далее щелкните кнопкой мыши на имени файла. Вы должны будете увидеть всплывающее окно, в котором спрашивается, хотите ли вы загрузить файл (рис. 10.12).

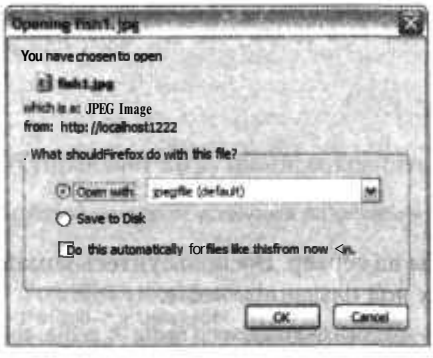


Рис 10.12. Окно загрузки, появляющееся при выборе ссылки на **медиафайл**

Я воспользовался командой Open with (Открыть с помощью), чтобы просмотреть JPEG-файл в специально предназначенной для этого программе и убедиться, что файл был загружен корректно. Результат показан на рис. 10.13. Как вы можете видеть, после нажатия кнопки **OK** изображение было получено, так что можете приступать к загрузке MP3-файлов (см. трюк 96).

Рис 10.13. Программа для просмотра **JPEG-файлов** в Windows, отображающая загруженный файл

Смотрите также

- «Создание динамических списков воспроизведения» (см. трюк 96).

ТРЮК №98

Следите за вашей сетевой игрой при помощи PHP

Используйте модуль PEAR Net_GameServerQuery, чтобы следить за вашей сетевой игрой при помощи PHP.

Складывается впечатление, что модули PEAR (см. трюк 2) есть практически для; всего! Чтобы доказать, что это так, в этом трюке мы воспользуемся модулем' Net_GameServerQuery для проверки сервера Half Life, чтобы просто узнать, сколько игроков сейчас находятся в игре (а также чтобы продемонстрировать возможности этого и других крутых модулей PEAR).

Код

Сохраните исходный код из примера 10.9 в файл index.php.

Пример 10.9. Автоматическая проверка сервера Half Life

```
<?php
require( 'Net. GameServerQuery.php' );
$protocol = 'halflife';
$ip = ' 66.159.222.15';
$gsq = new Net_GameServerQuery( );
$gsq->addServer( $protocol, Sip ):
$res = $gsq->execute( ):
?>
<html>
<head>
<title>Game Server Status</title>
</head>
<style>
body { font-family: arial, verdana, sans-serif: }
th { font-size: xx-small; border-bottom: 1px solid black: }
td { font-size: xx-small; vertical-align: top; }
.num-players { text-align: center; }
.header { font-weight: bold: }
</style>
<body>
<table>
<tr>
<td class="header">Protocol</td>
<td><?php echo($protocol); ?></td>
</tr>
<tr>
<td class="header">IP</td>
<td><?php echo($ip): ?></td>
</tr>
</table>
<table width="100%" cellspacing="0" cellpadding="3">
<tr>
<th width="20%">IP/Port</th>
<th width="20%">Password</th>
<th width="20%">Hostname</th>
<th width="20%">Players</th>
```

```
<th width="20%">Mod</th>
</tr>
<?php foreach( $res[0] as $r ) { ?>
<tr>
<td width="20%"><?php echo($r['ip']); ?><br/><?php echo($r['port']); ?></td>
<td width="20%"><?php echo($r['password']); ?></td>
<td width="20%"><?php echo($r['hostname']); ?></td>
<td width="20%" class="num-players">
<?php echo($r['numplayers']); ?> current<br/>
<?php echo($r['maxplayers']); ?> max
</td>
<td width="20%"><?php echo($r['mod']); ?x/td>
</tr>
<?php } ?>
</table>
</body>
</html>
```

Запуск трюка

Для начала установите модуль PEAR Net_GameServerQuery для PHP (см. трюк 2). Затем измените значения переменных $ip и $protocol, указав в них IP-адрес вашего игрового сервера и тип игры.

ПРИМЕЧАНИЕ

Документация по модулю Net_GameServerQuery (http://pear.php.net/net_gameserverquery) также включает в себя список протоколов и информацию по ним.

Загрузите сценарий на PHP-сервер и откройте страницу в вашем браузере (рис. 10.14).

Рис. 10.14. Статус игрового сервера Half Life

Вы можете пользоваться такого рода данными в нескольких целях. Можно создавать небольшие WML-страницы для вашего телефона, чтобы следить за обстановкой в игре, причем вы даже можете создать маленькие страницы для вашей PSP. Благодаря этому вы сможете знать, когда лучше всего стоит зайти в игру, даже находясь далеко! Когда же вы отвечаете за сам сервер, то можете проверить

его и перезапустить, если игра не запущена. Это намного лучше, чем просыпаться посреди ночи, не так ли?

Смотрите также

- «Получение информации из RSS-источников в PSP» (см. трюк 90).

ТРЮК №99

Просмотр Википедии при помощи PSP

Используйте MySQL и PHP, чтобы создать словарь из Википедии, который подходил бы к вашему карманному устройству.

Википедия (http://www.wikipedia.org), наверное, является самым крупным одиночным сайтом в Интернете с точки зрения информативности. Это энциклопедия и словарь, в которые пользователи могут вносить изменения. Плюс ко всему вы можете загрузить содержимое Википедии целиком и использовать в ваших собственных целях.

В моем случае я хотел заполучить словарь из Википедии для своей PSP. Будучи хитрым PHP-разработчиком, я, конечно же, воспользовался PHP и MySQL. Я создал нескатько статических страниц для Википедии и загрузил их в карту памяти для PSP. Словарь получился не динамический, но он все равно производит впечатление на моих приятелей, когда я, например, нахожу для них значение слова «grok» при помощи PSP.

На рис. 10.15 показан принцип работы этого трюка. Содержимое словаря из Википедии загружается в мою базу данных MySQL (см. трюк 1). Сценарий `dict.php` получает содержимое базы данных и тщательно обрабатывает страницы при помощи форматирования в HTML, чтобы они корректно отображались на PSP.

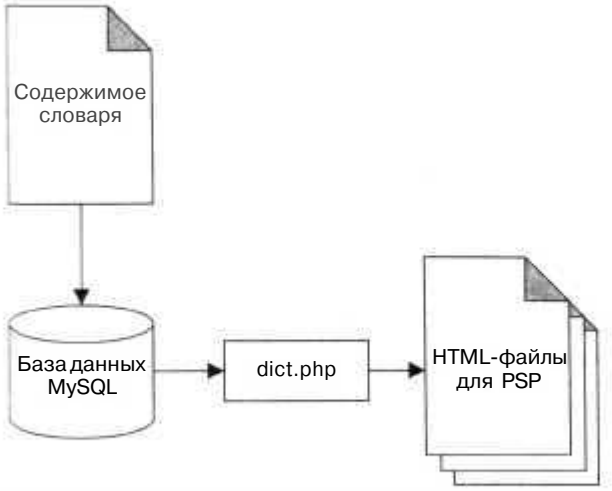


Рис. 10.15. Принцип работы трюка, создающего словарь для PSP

Код

Сохраните исходный код из примера 10.10 в файл dict.php.

Пример 10.10. Загрузка Википедии для создания статических HTML-страниц

```
<?php
require_once( "DB.php" );
require_once( "Text/Wiki.php" );
$g_wiki = new Text_wiki( );
$g_wiki->enableRule('html');
$g_wiki->enableRule('list');
function wikiToHTML( $text )
{
  global $g_wiki;
  $text = preg_replace( "/\-\-\-\s* Pronunciation.*?\n\=\=\=/is", "\n===",
    $text );
  $text = preg_replace( "/\=\=\=\=\=\s*(.*?)\s*\=\=\=\=\=/", "\n+++++ $1",
    $text );
  $text = preg_replace( V\-\-\-\-\s*(.*?)\s*\-\-\-W", "++++ $1", $text );
  $text = preg_replace( "/\=\=\=\s*(.*?)\s*\=\=\=/", "+++ $1", $text );
  $text = preg_replace( "/\=\=\s*(.*?)\s*\=\=/", "++ $1", $text );
  $text = preg_replace( "/\=\s*(.*?)\s*\=/", "++ $1", $text );
  $text = preg_replace( "/\[\[image:.*?\]\]/i", "", $text );
  $text = preg_replace( "/\[\[it:.*?\]\]/i", "", $text );
  $text = preg_replace( "/\[\[.*?\|(.*?)\]\]/", "$1", $text );
  $text = preg_replace( 7\[\[(.*?)\]\]/", "$1", $text );
  $text = preg_replace( 7\[(.*?)\]/", "$1", $text );
  $text = preg_replace( "/\n\#([^#])/", "\n# $1", $text );
  $text = preg_replace( "/\n\*([^*])/", "\n* $1", $text );
  $text = preg_replace( "/\<\!\-\-.*?\-\-\>/mi", "", $text );
  $text = preg_replace( 7\n\|.*?\|\s*\n/", "", $text );
  $text = preg_replace( 7\n\{\|.*\n/", "", $text );
  $text = preg_replace( "/\n\|\}.*\n/", "", $text );
  $text = preg_replace( "/\n\|\}\n/", "", $text );
  $text = preg_replace( .7\{\{.*?\}\}/", "", $text );
  $text = preg_replace( "/\|\}/", "", $text );
  $text = preg_replace( "/\|.*?\|/", "", $text );
  $text = preg_replace( "/\'\'\'\'\'/", "\n", $text );
  return $g_wiki->transform( $text, 'Xhtml' );
}

function goodWord( $word )
{
  if ( preg_match( "/^[A-Za-z]/", $word ) )
  {
    if ( preg_match( "/[^A-Za-z.-]/", $word ) ) return false;
    if ( preg_match( "/\-S/", $word ) ) return false;
    if ( preg_match( "/\[.]S/", $word ) ) return false;
    if ( preg_match( "/^.-/", $word ) ) return false;
    if ( preg_match( "/^.[.]/", $word ) ) return false;
    $cutword = preg_replace( "/[^A-Za-z]/", "", $word );
    if ( strlen( $cutword ) < 4 ) return false;
    if ( strlen( $cutword ) > 20 ) return false;
    return true;
  }
```

```
  return false:
}

function goodText( $text )
{
  if ( preg_match( "/#REDIRECT/i", $text ) )
    return false:
  return true:
}
$g_words = array( ):
$g_wurl = array( ):

$dsn = 'mysql://root:password@localhost/wp':
$db =& DB::Connect( $dsn, array( ) ):
if (PEAR::isError($db)) { die($db->getMessage( )): }
$blocksize = 100:
$total_html - "":
$block = :
$block_id = 1:
function writeBlock( $block. $html )
{
  $fh = fopen( "pages/words/".$block."html", "w" ):
  fwrite( $fh   "<html><head>\n" );
  fwrite( $fh    "<linkrel=\"stylesheet\"type=\"text/css\"
    href=\"../default.css\" />\n" );
  fwrite( $fh, "</head><body><div style='width:478px'>\n" );
  fwrite( $fh  $html ):
  fwrite( $fh "</div></body></html>\n" );
  fclose( $fh ):
}
$res = $db->query( "SELECT cur_title as word. cur_text as text FROM cur WHERE
cur_namespace=0" ):
while ( $res->fetchInto( $row. DB_FETCHMODE_ASSOC ) )
{
  Sword = $row[ word']:
  $text - $row[ text']:
  if ( r oword( $word ) J& goodText( $text ) )
  [
    $cl = strtolower( $word[0] ):
    if ( !isset( fcj_words[ $cl ] ) ) $g_words[ $cl ] = array( );
    $c2 = strtolower( $word[1] );
    if ( !isset( $g_words[ $cl ][ $c2 ] ) ) $g_words[ $cl ][ $c2 ] = array( ):
    $oword - $word:
    $word - strtolower( $word ):
    $g_words[ $cl ][ $c2 ] []= $oword;
    $g_wurl[ $word ] = "../words/".$block_id.".html#".$block:
    print( "$word\n" ):
    $total_html .- "<a name=\"".$block."\" />";
    $total_html .= "<div class='word-header'>".$oword."</div>";
    $total_html .= "<table width='100%' cellspacing='0'
      cellpadding='0'><tr><td>":
    $total_html .= wikiToHTML( $text );
    $total_html .- "</td></tr></table>":
    if ( $block >= $blocksize )
    {
      writeBlock( $block_id. $total_html ):
```

```
      $block_id++;
      $block = 0;
      $total_html = "";
    }
    else
      $block++;
  }
}
writeBlock( $block_id. $total_html ):
ob_start( ):
?>
<html><head><title>Index</title>
<link rel="stylesheet" type="text/css" href="default.css" />
</head><body><div style="width:478px;">
<div id="c1-header">
<?php
foreach( array_keys( $g_words ) as $cl )
{
  ?>
  <a href="lev1/<?php echo( $cl ); ?>.html"><?php echo( $cl ); ?></a>
  <?php
}
?>
</div></divx/bodyx/html>
<?php
$index - ob_get_clean( ):
$ih = fopen( "pages/index.html", "w" ):
fwrite( $ih, $index );
fclose( $ih );
ob_start( );
foreach( array_keys( $g_words ) as $cl )
{
  ?>
  <a href="../lev1/<?php echo( $cl ): ?>.html"><?php echo( $cl ); ?></a>
  <?php
}
$c1header - ob_get_clean( );
foreach( array_keys( $g_words ) as $cl )
{
  ob_start( );
  ?>
  <html><head><title><?php echo( $cl ): ?></title>
  <link rel="stylesheet" type="text/css" href="../default.css" />
  </head><body><div style="width:478px:">
  <div id="c1-header"><?php echo( $clheader ); ?></div>
  <?php foreach( array_keys( $g_words[$c1] ) as $c2 ) { ?>
  <a href="../lev2/<?php echo( $cl.$c2 ); ?>.html"><?php echo( $cl.$c2 ); ?></a>
  <?php } ?>
  </div></body></html>
  <?php
  $html = ob_get_clean( );
  $fh = fopen( "pages/lev1/".$c1.".html", "w" );
  fwrite( $fh, $html );
  fclose( $fh );
}
```

```
foreacht array_keys( $g_words ) as $cl )
{
  ob_start( ):
  foreacht array_keys( $g_words[$cl] ) as $c2 )
  {
    ?>
    <a href="<?php echot Scl.Sc2 ); ?>.html"><?php echot Scl.Sc2 ): ?></a>
    <?php
  }
  $c2header = ob_get_clean( ):
  foreacht array_keys( Sg_words[$cl] ) as Sc2 )
  {
    Swords = $g_words[ $c1 ][ Sc2 ]:
    ob_start( ):
    ?>
    <html><head><title><?php echot $cl.$c2 ); ?></title>
    <link rel-"stylesheet" type="text/css" href="../default.css" />
    </head><body><div style="width:478px;">
    <div id="c1-header"><?php echot Sclheader ); ?x/div>
    <div id="c2-header"><?php echot $c2header ); ?x/div>
    <?php foreacht Swords as Sword ) { ?>
    <a href="<?php echo( $g_wurl[ strtolowert Sword ) ] ): ?>"><?php echot Sword
      ): ?x/a>
    <?php } ?>
    </div></body></html>
    <?php
    $html - ob_get_clean( ):
    $fh = fopen( "pages/lev2/".$c1.$c2.".html", "w" ):
    fwrite( $fh. $html );
    fclose( $fh ):
  }
}
?>
```

Весь код делится на четыре основные части. В первой информация считывается из базы данных. Во второй части перебираются все записи и удаляются те из них, которые не подходят, а в отобранных записях убирается разметка HTML.

Для каждого блока из 100 слов создается отдельный HTML. Если сценарий будет создавать файл для каждого слова и даже если суммарный объем выходных данных будет таким же, все равно объема большинства карт памяти будет недостаточно. Итак, сценарий группирует слова по 100 штук на файл. Он также продолжает отслеживать, какие слова в каком файле используются, при помощи ассоциативной таблицы `g_wurl`, в которой в качестве значения для любого слова, используемого в качестве ключа, хранится URL-адрес.

В оставшихся двух частях кода сценарий создает первый и второй уровни страниц по буквам. На страницах первого уровня вверху расположен алфавит и ссылки на слова, начинающиеся с выбранной буквы и еще одной буквы из алфавита. На страницах второго уровня находятся все слова, которые начинаются с заданной двухбуквенной комбинации. Это разбиение на первую и вторую букву слова было сделано в целях экономии места и большей простоты использования словаря.

Запуск трюка

Для этого трюка вам потребуется модуль PEAR Text_Wiki (см. трюк 2). После этого вам придется скачать базу данных с самой последней версией английского словаря (Wiktionary) с сайта Википедии (http://download.wikipedia.org/). Затем вам надо загрузить словарь в вашу базу данных MySQL:

```
mysqladmin --user=root--password=password create wp
mysql --user=root--password=password create wp < 20050623_cur_table.sql
```

ПРИМЕЧАНИЕ

Имя файла будет варьироваться в зависимости от того, когда вы загрузили словарь из Википедии.

Когда словарь готов, запустите сценарий dict.php:

```
% php dict.php
aant
aave
abta
acas
acats
aclu
acme
acronym
...
zygapophysial
zygapophysis
zygote
zymurgy
zythum
zyzzyva
%
```

Пока можете выпить кофе, потому что сценарию потребуется время, чтобы завершить работу. Базы данных Википедии очень большие, и на их обработку уходит много времени. Например, на моем G4 PowerBook потребовалось около часа на формирование всех HTML-файлов. Затем загрузите эти файлы на карту памяти вашей PSP. Переключите ее в режим работы через USB и подключите к компьютеру. Загрузите браузер на PSP и откройте файл file://common/index.html (рис. 10.16).

Здесь вы можете выбрать первую и вторую буквы слова (чтобы задать более точный критерий для поиска). В итоге будет загружена страница со словами, которые начинаются с первых заданных букв. Эту страницу вы можете видеть на рис. 10.17.

Теперь найдите нужное слово и щелкните на нем кнопкой мыши. Вы перейдете к странице, на которой будет расположено определение этого слова. Пример такой страницы вы можете видеть на рис. 10.18. Скажите, круто! По мере того как Википедия продолжает расти, словарь будет изменяться в зависимости от того, когда вы запустите этот процесс для его создания. Когда я запустил этот сценарий, все HTML-файлы для словаря занимали 30 Мбайт, что вполне подходит даже для самых маленьких карт памяти.

Рис. 10.16. Главная страница словаря

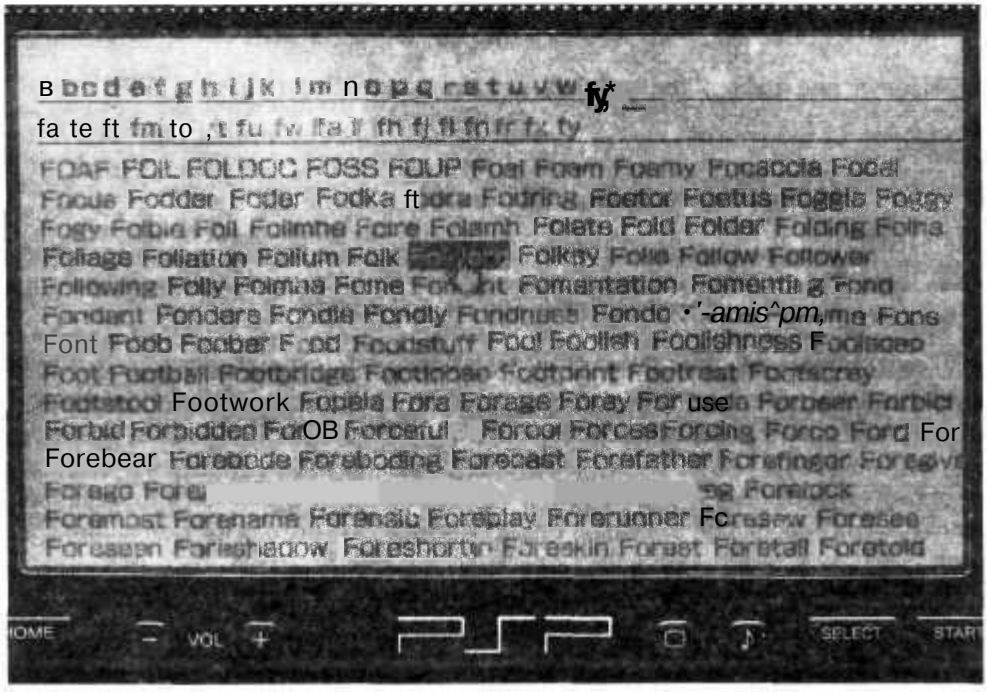


Рис. 10.17. Страница со словами, начинающимися с двух заданных букв

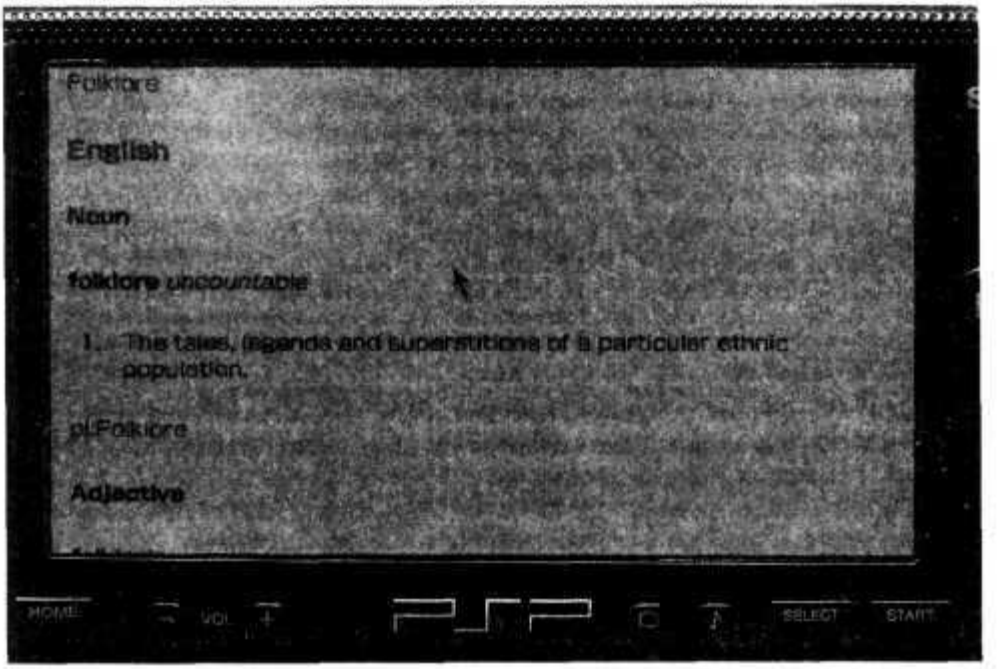


Рис. 10.18. Отображенное на экране слово folklore

Смотрите также

- «Получение информации из RSS-источников в PSP» (см. трюк 90).

ТРЮК №100

Отслеживание погоды

Используйте сервис сайта Weather.com для создания страницы, которая будет следить за прогнозом погоды и помогать вам решать, куда бы вам съездить на этой неделе (и что с собой взять).

При помощи модуля PEAR `Services_Weather` (см. трюк 2) можно запросто добавить информацию о прогнозе погоды в ваше интернет-приложение. В этом трюке используется информация, получаемая от сервисов сайта Weather.com при помощи модуля PEAR `Services_Weather`, чтобы узнать, какой будет погода в течение недели в двух разных городах. Затем эти значения сравниваются с заданными вами и в итоге сообщается, в какой город стоит поехать исходя из пожеланий к погоде.

Код

Сохраните исходный код из примера 10.11 в файл `index.php`.

Пример 10.11. Код на PHP для проверки прогноза погоды

```
<?php
require_once( "Services/Weather.php" ):
Sweather = new Services_Weather( );
Swdc = $weather->service( 'Weatherdotcom' );
function get_average( Szip )
{
  global Swdc:
         $fc = Swdc->getForecast( Szip, 7 );
         Sdayavg - 0;
  foreach ( $fc['days'] as Sday )
  {
    Shigh - $day['temperatureHigh'];
    Slow - $day['temperatureLow'];
    Sdayavg += Shigh:
    Sdayavg += $low;
  }
  return floor((Sdayavg/14)):
}
Szipa = isset( $_GET['zipa'] ) ? $_GET['zipa'] : '94587';
Szipb = issett $_GET['zipb'] ) ? $_GET['zipb'] : '19081';
Sdesired - issett $_GET['desired'] ) ? $_GET['desired'] : '65';
Stempa = get_average( Szipa ):
Stempb = get_average( Szipb ):
$da - abs( Sdesired - Stempa ):
Sdb - abs( $desired - Stempb ):
Svictor = ( Sda < Sdb ) ? 1 : 2:
$stylea = ( Svictor - 1 ) ? "background: #bbb:" : "";
Sstyleb - ( Svictor – 2 ) ? "background: #bbb:" : "";
?>
<html>
<head>
<title>Average Temperature Showdown</title>
<style type="text/css">
td { text-align: center; }
</style>
</head>
<body>
<form>
<table width="600">
<tr>
<th>Desired</th>
<th style="<?php echo($stylea); ?>"><input type="text" name="zipa" value="<?php
echo( Szipa ); ?>" size="6" /></th>
<th style="<?php echo($styleb); ?>"><input type="text" name="zipb" value="<?php
echo( Szipb ): ?>" size="6" /></th>
</tr>
<tr>
<td><input type="text" name="desired" value="<?php echo( Sdesired ): ?>" size="3"
/></td>
<td style="<?php echo($stylea); ?>"><?php echo( Stempa ); ?></td>
<td style="<?php echo($styleb); ?>"><?php echo( Stempb ): ?></td>
</tr>
<tr>
<td colspan="3"><input type="submit" value="Compare" /></td>
<tr>
```

```
</table>
</form>
</body>
</html>
```

Сервис, предоставляющий информацию о погоде, получает доступ к сайту при помощи модуля PEAR `Services_Weather`, которому в качестве ответа на введенный почтовый индекс возвращается структура, содержащая прогноз погоды. Информация в этой структуре не очень детальна. В ней есть массив с днями, каждый из которых содержит хеш-таблицу с ожидаемой температурой. На созданной странице высчитывается средняя температура за несколько дней, которая потом и выводится в виде сравнения для двух городов.

Запуск трюка

Загрузите и установите модуль PEAR `Services_Weather` (см. трюк 2) на ваш сервер, а также загрузите туда файл `index.php`. Откройте страницу в браузере (рис. 10.19).

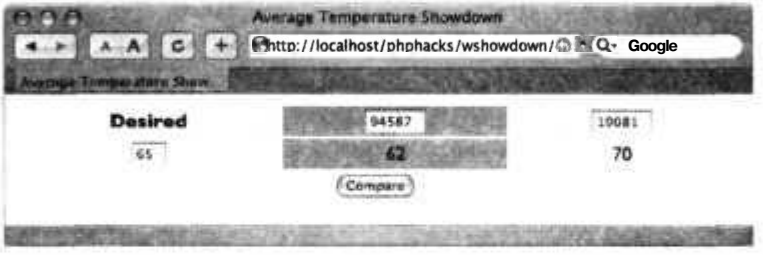


Рис. 10.19. Сравнение температур в Юнион-Сити (Калифорния) и Суортморе (Пенсильвания)

Измените в заголовке значение почтового кода на ваш и нажмите кнопку **Compare** — вы узнаете, в каком городе на следующей неделе температура будет наиболее благоприятна для того, чтобы его посетить. Если вы включите фантазию, то можете добавить, например, карты Google (см. трюк 95) и сделать действительно сногсшибательное приложение.

Смотрите также

- «Организация поиска в Google при помощи диаграмм ссылок» (см. трюк 91).
- «Создание нового интерфейса для Amazon.com» (см. трюк 92).

Алфавитный указатель

E

F

G

H

S

T

U

V

W

X

Z

А

Б

В

Г

К

P

C

T

У

Ф

O'REILLY®

PHP. ТРЮКИ

100 советов и рекомендаций профессионалов

Программисты любят PHP за его гибкость и скорость, дизайнеры — за его доступность и удобство. Когда речь заходит о создании сайтов, использование языка сценариев PHP — поистине верное решение. Сегодня PHP применяется в 19 миллионах веб-сайтов, превышая популярность технологии Microsoft ASP .NET. Неудивительно, что на рынке появилось огромное множество книг по PHP. Однако издание, которое вы держите в руках, дает действительно целостное понимание принципов применения этого языка — начиная от традиционного веб-программирования и заканчивая применением PHP в графике, мультимедиа и т. д.

При написании этой книги Джек Харрингтон опирался на свой 20-летний опыт создания программных кодов. Это позволило ему изложить самые разнообразные сведения — от базовых основ PHP, установки PEAR и написания простейших сценариев до работы с мультимедиа и способов оптимизации баз данных.

Прочитав эту книгу, вы научитесь разрабатывать мощные PHP-приложения, узнаете о том, как улучшить свою дизайнерскую базу данных, применять автоматизированное тестирование приложений и использовать шаблоны в сценариях и классах PHP. Кроме того, Джек Харрингтон объясняет, как обновлять веб-интерфейс путем создания закладок, всплывающих окон и календарей. Вы также узнаете о том, как использовать в PHP карты Google и графики.

В издании приводится множество иллюстраций и примеров программ, позволяющих вам:

- встраивать на свои сайты карты и спутниковые изображения Google;
- динамически просматривать фотографии из iPhoto в режиме on-line;
- добавлять в приложения возможность общения с использованием IRC, SMS, мгновенных сообщений;
- просматривать на Sony PlayStation статьи из Википедии;
- визуализировать графику и пользовательские интерфейсы с помощью SVG, DHTML и AJAX.

Книга «PHP. Трюки» окажется полезной и интересной всем читателям — от новичка до эксперта.

Тема:	Программирование для Интернета/ PHP. Серверный скриптовый язык	Уровень пользователя:	начинающий/опытный

ПИТЕР®

Заказ книг:

197198, Санкт-Петербург, а/я 619
тел.: (812) 703-73-74, postbook@piter.com
61093, Харьков-93, а/я 9130