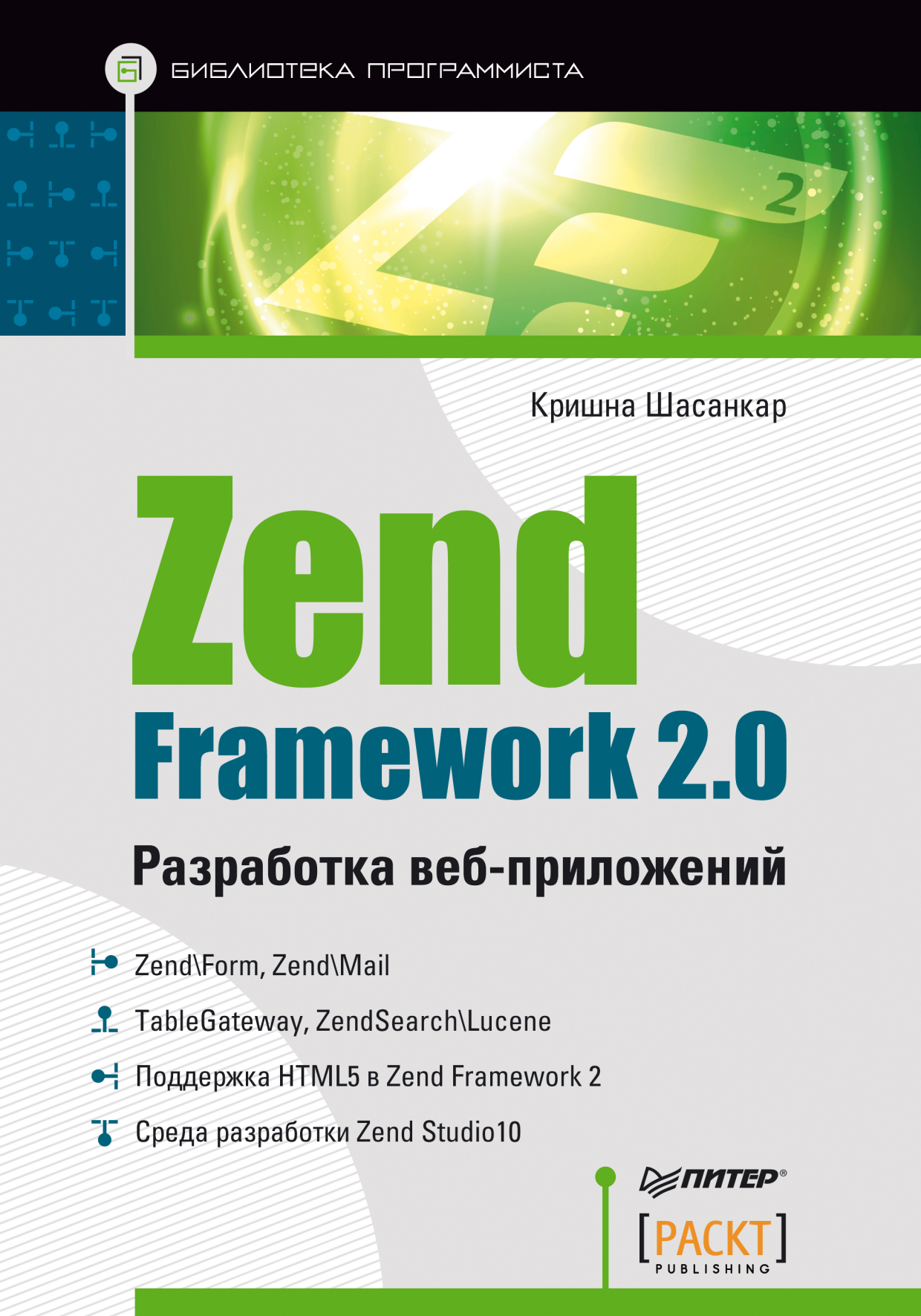

БИБЛИОТЕКА ПРОГРАММИСТА
Кришна Шасанкар
Zend
Framework 2.0
Разработка веб-приложений
Zend\Form, Zend\Mail
TableGateway, ZendSearch\Lucene
Поддержка HTML5 в Zend Framework 2
Среда разработки Zend Studio10
ПИТЕР
PACKT
PUBLISHING

Krishna Shasankar V

Zend Framework 2.0 by Example Beginner's Guide

A step-by-step guide to help you build full-scale web applications using Zend Framework 2.0

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

БИБЛИОТЕКА ПРОГРАММИСТА

Кришна Шасанкар

Zend Framework 2.0

Разработка веб-приложений

Москва · Санкт-Петербург · Нижний Новгород · Воронеж
Ростов-на-Дону · Екатеринбург · Самара · Новосибирск
Киев · Харьков · Минск
2014

ББК 32.988.02-018
УДК 004.738.5
Ш27

Шасанкар К.

Ш27 Zend Framework 2.0 разработка веб-приложений. — СПб.: Питер, 2014. — 208 с.: ил. — (Серия «Библиотека программиста»).

ISBN 978-5-496-00837-2

Zend Framework 2 представляет собой последнее обновление широко известного фреймворка Zend Framework. Эта версия значительно упростила процесс создания сложных веб-приложений, сведя к минимуму усилия разработчиков благодаря наличию готовых к использованию компонентов. Zend Framework 2 — это многофункциональный масштабируемый фреймворк для разработки веб-приложений.

Данная книга послужит для вас руководством по созданию мощных веб-приложений средствами Zend Framework 2. В ней рассматриваются все аспекты создания приложений на основе Zend Framework, начиная с установки и конфигурирования среды разработки, а имеющиеся упражнения позволят вам с легкостью разобраться в возможностях ZF и воспользоваться ими для создания собственных приложений.

12+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.988.02-018
УДК 004.738.5

ISBN 9781782161929 англ.
ISBN 978-5-496-00837-2

Краткое содержание

Оглавление

Об авторе

Кришна Шасанкар V — веб-разработчик с семилетним обширным опытом программирования на языке PHP. Он руководит группой инженеров в Lister Technologies, разрабатывающей решения для предприятий розничной и электронной торговли.

Кришна является сертифицированным инженером Zend по PHP и Zend Framework. Он получил ученую степень бакалавра информационных технологий в Университете Анна (Ченнаи, Индия) и степень магистра программного обеспечения в научно-технологическом институте Бирла (Пилани, Индия).

Свободное время Кришна посвящает музыке, фотографии и путешествиям, с особым удовольствием совмещая эти хобби. Он ведет блог www.clickoffline.com, через который с ним можно связаться и оставить комментарии.

Я хотел бы поблагодарить своих родителей, брата и друзей, которые вдохновляли и поддерживали меня на протяжении всей моей жизни.

Спасибо Мукунду Деверайану (Mukund Deverajan) за горячую и самоотверженную поддержку, без которой создание этой книги было бы невозможным. Я благодарю Апурва Бхаргаву (Apoorv Bhargava), Джаябхарати и Сувика Сенгупта (Jayabharathi and Souvik Sengupta) за вдохновение и помощь в переработке значительной части текста книги. Особой благодарности заслуживает моя замечательная команда в Lister Technologies за прекрасную поддержку и веселое настроение — вы отличные ребята!

Спасибо рецензентам — Уэнберту Дель Розарио (Wenbert S. Del Rosario), Алексу Френкелю (Alex Frenkel) и Исламу Мохамеду Абдель-Азизу (Islam Mohamed Abdelaziz) за ценные замечания в процессе редактирования.

Наконец, я благодарю Энтони Лау (Antony Lowe), Нишму Рамакришнан (Neeshma Ramakrishnan), Вину Пагаре (Veena Pagare) и других участников замечательного коллектива издательства Packt Publishing, которые внесли вклад в создание этой книги, обеспечив качество на всех этапах работы над ней. Я очень признателен сотруднице Packt Publishing Ануге Хуране (Anugya Khurana) — без ее терпения и упорства выход книги пришлось бы многократно откладывать. Особая благодарность Вине Манжрекар (Veena Manjrekar) за то, что дала мне эту возможность.

О рецензентах

Уэнберт Дель Розарио — веб-разработчик, имеющий двухлетний опыт работы над технологиями с открытым исходным кодом (Linux, CakePHP, Code Igniter, MySQL, jQuery, Knockout JS и WordPress). В свободное время он занимается личными проектами, ведет внештатную и консультационную деятельность. Уэнберт также выступил в качестве рецензента двух книг издательства Packt Publishing:

- Кит Поп, «Разработка веб-приложений с помощью Zend Framework 1.8» (Keith Pope, Zend Framework 1.8 Web Application Development);
- Тим Юравич, «Веб-разработка на CouchDB и PHP. Руководство для начинающих» (Tim Juravich, CouchDB and PHP Web Development Beginner's Guide).

Уэнберт ведет блог http://blog.ekini.net, в котором делится своими идеями, решениями и своим мнением о текущих событиях. С ним также можно связаться через Twitter @wenbert.

Посвящается Ноэме и нашему сыну Лукасу.

Алекс Френкель работает в сфере разработки веб-приложений с 1998 года (со времен создания PHP 3.X) и имеет богатый опыт системного анализа и управления проектами. Алекс — сертифицированный Zend-инженер по PHP 5.3 и считается одним из наиболее авторитетных LAMP-разработчиков в Израиле.

В прошлом Алекс был техническим директором ReutNet — одной из передовых израильских веб-компаний, а также совмещал должности генерального и технического директора в инновационной компании OpenIview LTD., занимающейся вытеснением с рынка мейнфреймов IBM с помощью PHP-приложений. Алекс также выступал в роли эксперта, консультировавшего различные компании по вопросам, связанным с веб-технологиями.

В настоящее время Алекс возглавляет недавно созданную компанию GBooking и является владельцем небольшой консультационной фирмы Frenkel-Online.

Компания GBooking предоставляет клиентам возможность искать, сравнивать и приобретать разнообразные интернет-услуги, предлагает скидки в периоды низкого спроса и направляет клиентов на сайты партнеров.

Frenkel-Online — проектная компания, работающая с внештатными консультантами из Израиля и других стран. В настоящее время в компании сложился постоянный коллектив, работающий над PHP-проектами, имеются также специалисты по другим языкам программирования, которые привлекаются к остальным проектам по необходимости.

Ислам Абдель-Азиз — старший разработчик программ с открытым исходным кодом и соавтор Zend Framework, имеющий статус сертифицированного инженера Zend с 2009 года.

У Ислама девятилетний опыт преподавания и консультирования в сфере современных веб-технологий и корпоративных информационных систем. Он занимается методиками разработки программ, в том числе построением баз данных по принципу NoSQL, вопросами масштабируемости веб-систем, а также параллельными и распределенными вычислениями на основе модели MapReduce.

В течение последних семи лет Ислам принимал участие в многочисленных проектах по разработке программ с открытым исходным кодом и имеет опыт работы с большинством используемых в них технологий, в том числе языками PHP5, Python и Java.

В 2008 году Ислам стал старшим инженером-программистом в компании Oracle. Он работал в команде, создававшей наиболее стабильную платформу для облачных вычислений на языке Python.

В настоящее время Ислам возглавляет группу арабских разработчиков в ADVFN — самой популярной компании по разработке программного обеспечения для финансов в Великобритании. Он отвечает за разработку версий продуктов ADVFN для стран Ближнего Востока.

Я хотел бы выразить благодарность моей жене за то, что была рядом, пока я рецензировал эту книгу.

Моим родителям, Парамаджоти
и Анурагалате, чьи абсолютная любовь
и самопожертвование дали мне
возможность стать тем, кем я стал.

Предисловие

Zend Framework 2 представляет собой последнее обновление широко известного фреймворка Zend Framework. Эта версия значительно упростила процесс создания сложных веб-приложений, сведя к минимуму усилия разработчиков благодаря наличию готовых к использованию компонентов. Zend Framework 2 — это многофункциональный масштабируемый фреймворк для разработки веб-приложений.

Данная книга послужит для вас руководством по разработке мощных веб-приложений средствами Zend Framework 2. В ней рассматриваются все аспекты создания приложений на основе Zend Framework, начиная с установки и конфигурирования среды разработки, а имеющиеся в ней упражнения построены таким образом, что читатели смогут с легкостью разобраться в них и воспользоваться ими для создания собственных приложений.

Книга начинается с описания процедур базовой установки и настройки Zend Framework. Выполняя упражнения, вы детально познакомитесь с Zend Framework 2. С помощью этой книги вы освоите основные концепции создания добротных MVC-приложений для Интернета на базе Zend Framework 2. Подробные пошаговые инструкции позволят вам разработать такие программные продукты, как групповой чат, служба совместного доступа к файлам и мультимедиа, поисковая система и простой онлайновый магазин. Вы также воспользуетесь широким кругом внешних библиотек, чтобы реализовать те функциональные возможности, которые не поддерживаются в штатной комплектации фреймворка.

По окончании работы над книгой вы будете уверенно разбираться в вопросах создания сложных и многофункциональных веб-приложений с помощью Zend Framework 2.

О чем эта книга

Глава 1, «Начало работы с Zend Framework», посвящена конфигурированию среды разработки. В ней мы настроим сервер PHP-приложений, установим СУБД MySQL и создадим базу данных, которая будет использоваться в практических упражнениях по Zend Framework в последующих главах.

Глава 2, «Создание первого приложения с помощью Zend Framework», рассказывает о создании проекта Zend Framework 2; мы познакомимся с ключевыми аспектами разработки MVC-приложения в Zend Framework 2, создавая модули, контроллеры и представления. Модуль, который мы создадим, будет расширяться в последующих главах этой книги.

Глава 3, «Создание коммуникационного приложения», знакомит читателя с компонентом Zend/Form. В этой главе мы создадим нашу первую регистрационную форму и настроим механизм входа в систему и аутентификации зарегистрированных пользователей с помощью компонентов Zend Framework.

Глава 4, «Управление данными и совместное использование документов», посвящена концепциям управления данными и файлами в Zend Framework. В этой главе мы познакомимся с различными аспектами применения Zend Framework, в том числе с менеджером служб, паттерном TableGateway, обработкой выгрузки файлов и их совместным использованием.

Глава 5, «Чат и электронная почта», рассказывает об использовании языка JavaScript в вашем приложении. В качестве примера, на котором объясняется применение возможностей JavaScript в разработке веб-приложений, используется реализация группового чата. Кроме того, в этой главе рассматривается отправка сообщений электронной почты с помощью компонента Zend\Mail, а также менеджер событий Zend Framework.

Глава 6, «Совместный доступ к мультимедиа», знакомит разработчиков с управлением и организацией общего доступа к изображениям и видеороликам в Zend Framework. В этой главе используются различные внешние модули Zend Framework, предназначенные для работы с изображениями и видеороликами.

Глава 7, «Поиск с помощью библиотеки Lucene», рассказывает о разработке поисковой системы на основе библиотеки Lucene с использованием Zend Framework. В начале главы речь идет об установке модуля ZendSearch\Lucene, а затем подробно рассматривается реализация механизмов поиска в базе данных и файлах документов.

Глава 8, «Создание простого магазина», служит введением в системы электронной коммерции. В этой главе мы создадим простой онлайновый магазин, что позволит продемонстрировать процесс разработки товарной корзины. Для обработки платежей мы воспользуемся платежной системой PayPal Express Checkout.

Глава 9, «Поддержка HTML5», посвящена поддержке спецификации HTML5 в Zend Framework 2. В отличие от предыдущей версии Zend Framework, версия 2 обеспечивает всестороннюю поддержку различных функциональных возможностей HTML5; в этой главе рассматриваются два ключевых аспекта применения HTML5 — новые элементы ввода данных и множественная выгрузка файлов.

Глава 10, «Создание мобильных приложений», знакомит читателя с созданием «родных» мобильных приложений средствами Zend Framework 2 и среды разработки Zend Studio 10. В этой главе мы изучим основы создания облачных мобильных приложений в Zend Framework и рассмотрим вопросы настройки облачной среды разработки PHP-приложений.

Что нужно для работы над книгой

Вам потребуется компьютер, на котором сможет работать стек Zend Server CE и СУБД MySQL. Программы, необходимые для выполнения упражнений, перечислены в главе 1.

Для кого написана эта книга

Если вы — PHP-разработчик, не знакомый с Zend Framework, но желающий быстро освоить этот фреймворк на практике, то эта книга для вас. Предполагается, что вы обладаете базовыми знаниями в области объектно-ориентированного программирования на языке PHP.

Соглашения

Четкие инструкции по выполнению процедуры или упражнения приводятся в разделах, название которых начинается с фразы «Время действовать...». Инструкции часто требуют дополнительных комментариев, поясняющих их смысл; эти комментарии находятся в разделах «Что сейчас произошло?».

Для того чтобы вы могли проверить собственные знания, в конце каждой главы есть контрольные вопросы, в которых приведены короткие вопросы с вариантами ответов на них.

Практические задания, позволяющие поэкспериментировать с изученным материалом, можно найти в разделах «Самостоятельная работа».

Различные виды информации, встречающиеся в тексте книги, оформлены в тексте особым образом. Далее приведены примеры этих вариантов оформления с пояснениями их смысла.

Фрагменты кода в тексте имеют следующий вид: «класс `TableGateway` расширяет класс `AbstractTableGateway`, реализующий интерфейс `TableGatewayInterface`».

Блок кода выглядит следующим образом:

```
  // Добавление документа в индекс
  $indexDoc = new Lucene\Document();
  $indexDoc->addField($label);
  $indexDoc->addField($owner);
  $indexDoc->addField($fileUploadId);
  $index->addDocument($indexDoc);
}
// Запись индекса
$index->commit();
```

Чтобы привлечь внимание читателя к конкретным фрагментам блока кода, соответствующие строки или позиции в строках выделены полужирным начертанием:

```
  // Добавление документа в индекс
  $indexDoc = new Lucene\Document();
  $indexDoc->addField($label);
  $indexDoc->addField($owner);
  $indexDoc->addField($fileUploadId);
  $index->addDocument($indexDoc);
}
// Запись индекса
$index->commit();
```

Команды, набираемые в командной строке, и результаты их выполнения имеют вид:

```
$ sudo apt-get install php5-cli
$ sudo apt-get install git
$ curl -s https://getcomposer.org/installer | php
```

Новые термины или **важные слова** выделяются в тексте полужирным начертанием.

Элементы интерфейсов, появляющиеся на экране, например в меню или диалоговом окне, а также адреса в Интернете и названия файлов выделены особым шрифтом: «в окне Select Destination Location (Выбор адреса назначения) щелкните на кнопке Next (Далее), чтобы согласиться с адресом, предлагаемым по умолчанию».

ПРИМЕЧАНИЕ

В виде таких врезок оформляются предупреждения, практические рекомендации и важные комментарии.

От издательства

Исходные тексты листингов, приведенные в книге, можно получить, зарегистрировавшись на сайте http://www.packtpub.com.

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

Мы будем рады узнать ваше мнение!

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

Начало работы с Zend Framework

1

В этой главе мы создадим и настроим среду разработки, чтобы приступить к созданию программ с помощью Zend Framework 2.0. Мы настроим сервер PHP-приложений, установим СУБД MySQL и сформируем базу данных, которая будет использоваться в упражнениях по Zend Framework в следующих главах. Итак, начнем!

Zend Framework 2.0

Последний главный релиз Zend Framework (версия 1.0) состоялся в 2007 году. За прошедшие 5 лет среда Zend Framework была подвергнута существенной переработке и стала развитым PHP-фреймворком. Тем не менее обычные обновления не устраняют ряд проблем, изначально присущих Zend Framework 1.0.

Zend Framework 2.0 — это попытка улучшить среду Zend Framework, переконструировав ее изнутри. Вот некоторые новые ключевые возможности Zend Framework 2.0:

- средства языка PHP 5.3, такие как пространства имен и замыкания;
- модульная архитектура приложений;
- менеджер событий;
- вставка зависимостей (Dependency Injection, DI).

Мы познакомимся с реализацией новых возможностей Zend Framework 2.0 в следующих главах.

В этой главе мы рассмотрим вопросы установки и конфигурирования некоторых инструментов, необходимых для работы с Zend Framework 2.0. Zend Framework 2 можно установить на большинство веб-серверов, поддерживающих PHP версии 5.3.3 и выше.

Мы использовали программу Zend Server Community Edition в качестве веб-сервера, предлагаемого по умолчанию. Тем не менее вместо нее можно задействовать любой PHP-стек, поддерживающий язык PHP 5.3.3. В качестве альтернативы вы можете загрузить Apache и PHP по отдельности и установить PHP поверх Apache.

ПРИМЕЧАНИЕ

Чтобы упростить процесс установки, в этой книге я использую операционную систему Linux как основную среду разработки. Все инструменты, описываемые в книге, доступны для ОС Windows и могут быть использованы для выполнения тех же действий.

Знакомство со стеком Zend Server CE

Zend Server Community Edition (CE) — это бесплатная версия популярного стека Zend Server. Zend Server представляет собой готовый стек для PHP-приложений, который можно использовать в процессе их разработки, тестирования и выпуска. Эта программа предоставляет группам разработчиков приложений единую рабочую среду на всех стадиях разработки.

Инструментальные средства Zend Server также включают в себя Zend Optimizer+ для кэширования байт-кода PHP-приложений и Zend Guard для шифрования файлов.

Системные требования к Zend Server CE

В Zend Server имеются установщики для операционных систем Windows, Mac OS X, а также универсальный установочный пакет, совместимый с большинством дистрибутивов Linux.

Более подробную информацию о требованиях к установке Zend Server CE можно найти по ссылке http://www.zend.com/en/products/server/system-requirements.

Время действовать — установка Zend Server CE

Наш следующий шаг — загрузить и установить Zend Server CE. Я работаю с операционной системой Ubuntu 12.04 Precise Pangolin; процедура установки для других ОС может быть иной. За инструкциями по установке можно обратиться на веб-сайт Zend Server. Чтобы установить программу Zend Server, выполните следующие шаги.

1. Посетите веб-сайт Zend Server Community Edition (http://www.zend.com/en/community/zend-server-ce) и загрузите последнюю версию Zend Server, совместимую с вашей операционной системой. В этом примере мы загружаем установщик для Linux.

2. По завершении загрузки извлеките содержимое установщика во временное хранилище:

```
$ tar -zxvf ZendServer-5.6.0-RepositoryInstaller-linux.tar.gz
```

3. После извлечения нужно запустить установщик с правами администратора:

```
$ cd ZendServer-RepositoryInstaller-linux/
$ sudo ./install_zs.sh 5.3 ce
```

ПРИМЕЧАНИЕ

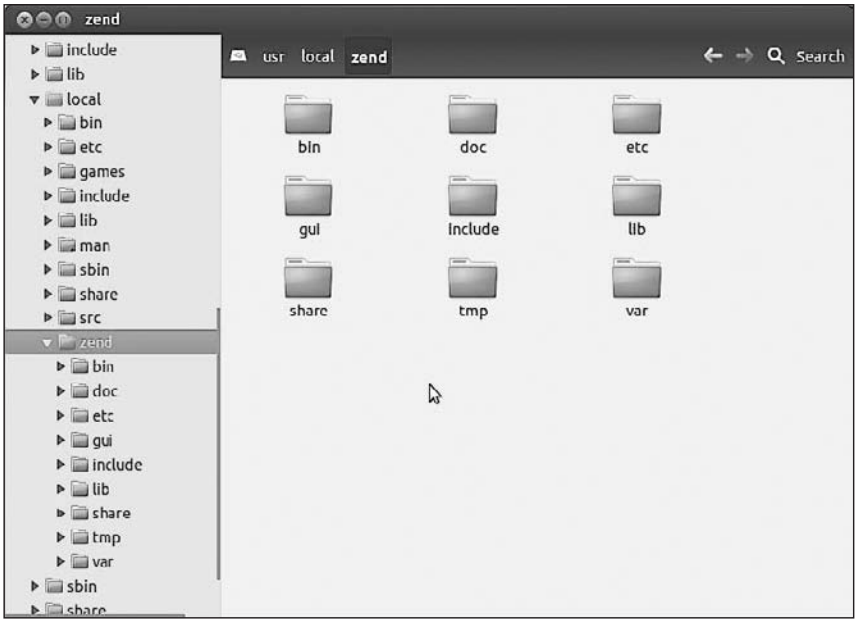
Мы передаем установщику два параметра. Первый — это устанавливаемая версия языка PHP; в данном случае — версия 5.3. Второй параметр определяет устанавливаемую редакцию Zend Server; в данном случае — ce (сокращение от Community Edition).

4. В процессе установки установщик потребует от вас загрузить различные пакеты.

5. По умолчанию программа Zend Server устанавливается в каталог /usr/local/zend, а корневым каталогом для документов является /var/www. Для изменения конфигураций экземпляра Zend Server можно воспользоваться следующими файлами:

   - основные варианты конфигурации сервера Apache находятся в файле /etc/apache2/apache2.conf;
   - PHP-конфигурации определяются в файле /var/local/zend/etc/php.ini.

   На следующем рисунке показан каталог с установленной программой Zend Server.



6. После завершения установки у вас должна появиться возможность открыть ссылку http://localhost в веб-браузере. Она должна привести вас на тестовую страницу.

СОВЕТ

Чтобы перезапустить Zend Server, выполните команду $ sudo service zend-server restart.

Что сейчас произошло?

Программа Zend Server CE установлена и готова к использованию. Теперь у нас есть веб-сервер и запущена совместимая версия PHP, которая удовлетворяет основным требованиям Zend Framework 2.0.

Самостоятельная работа

Для управления проектами Zend Framework через сервис Github мы будем использовать систему управления версиями файлов Git; одно из ключевых нововведений Zend Framework 2.0 заключается в том, что система контроля исходного кода SVN заменена на Git.

СОВЕТ

Бинарные файлы Git можно загрузить с сайта http://www.git-scm.com/ или установить из репозитариев вашей ОС.

Инструкции по установке Git можно найти по ссылке http: //git-scm.com/book/en/Getting-Started-Installing-Git.

Конфигурирование стека Zend Server CE

Наш следующий шаг — настроить Zend Server CE и внести некоторые изменения в конфигурацию, которые позволят нам запускать другие PHP-приложения.

Административный интерфейс Zend Server CE

Административный интерфейс Zend Server CE представляет собой пользовательский веб-интерфейс со следующими возможностями:

- управление PHP-расширениями;
- конфигурирование PHP-директив;
- управление компонентами Zend Server;
- мониторинг состояний PHP и состояний расширений, а также журналов приложений и серверов.

В нашей следующей задаче мы изменим конфигурацию Zend Server, используя административный интерфейс этой программы.

Время действовать — конфигурирование Zend Server CE

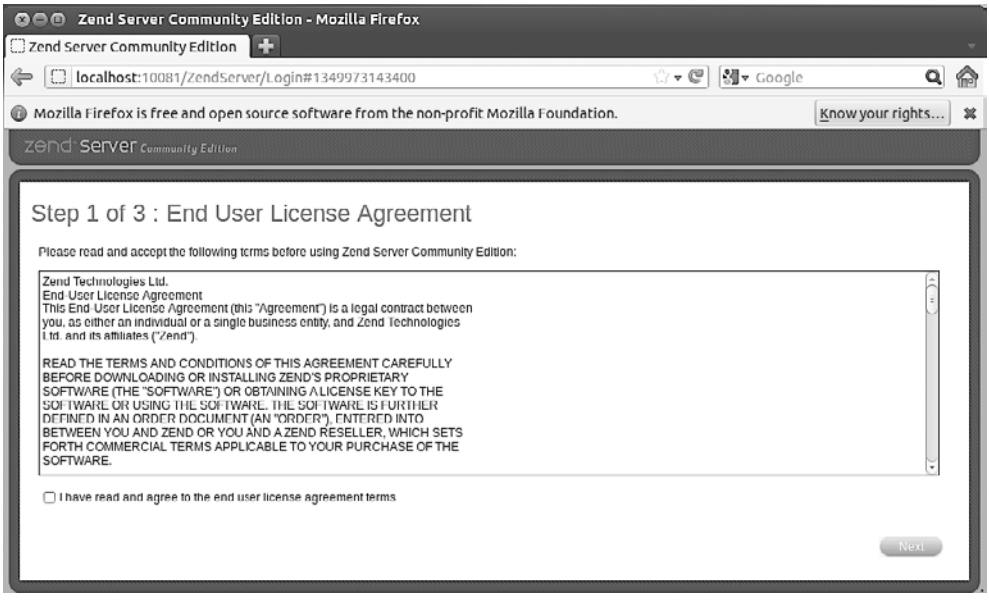
После завершения установки Zend Server программу необходимо настроить.

1. В браузере, предлагаемом по умолчанию, откройте консоль администрирования Zend Server (http://localhost:10081/).

ПРИМЕЧАНИЕ

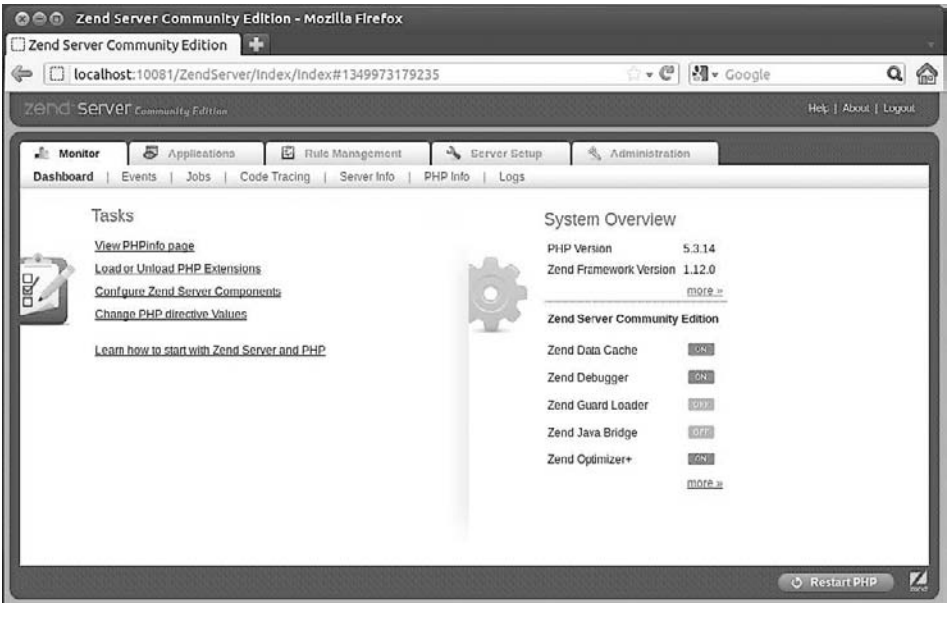
Консоль пользовательского интерфейса Zend Server задействует порт 10081, в то время как веб-сервер — порт 80. По этой причине нам необходимо явно указывать номер порта в URL-адресе для доступа к консоли пользовательского интерфейса.

2. При первом открытии административного интерфейса Zend Server запускается мастер конфигурирования. Ознакомьтесь с условиями лицензионного соглашения с конечным пользователем на странице End User License Agreement для Zend Server и примите их.



3. Мастер попросит вас задать пароль для установки Zend Server.

4. После завершения работы мастера начального конфигурирования вы будете перенаправлены на домашнюю страницу административного интерфейса Zend Server.



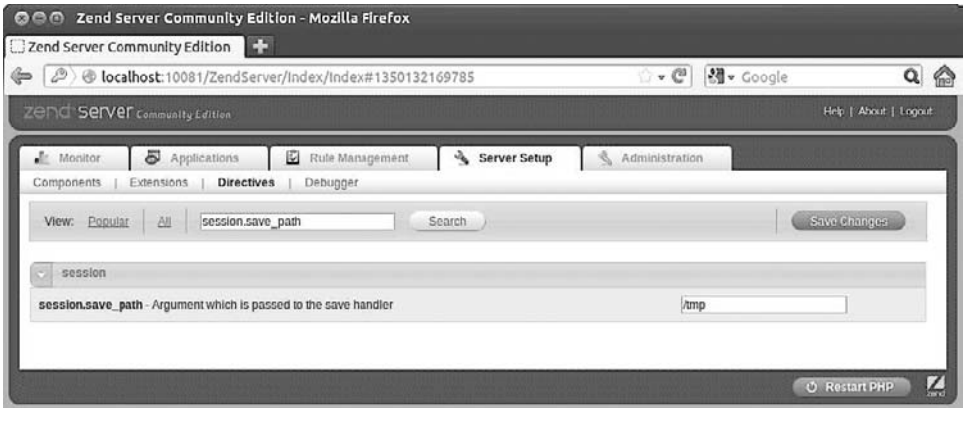
5. Нам необходимо задать путь для сохранения сеансов. Для этого выполните следующие шаги:

   1) перейдите в раздел Directives (Директивы) на вкладке Server Setup (Настройка сервера);

   2) найдите переменную `session.save_path`;

   3) присвойте ей значение `/tmp`;

   4) щелкните на кнопке Save Changes (Сохранить изменения), а затем — на кнопке Restart PHP (Перезапустить PHP).



Что сейчас произошло?

Мы успешно изменили конфигурацию сервера с помощью административного интерфейса Zend Server и перезапустили экземпляр PHP, работающий на Zend Server.

MySQL

MySQL не нуждается в представлении — это самая популярная в мире СУБД с открытым исходным кодом. Она бесплатна, доступна в Интернете независимым разработчикам и компаниям, желающим разрабатывать веб-сайты и приложения с помощью базы данных MySQL.

Zend Framework 2.0 поддерживает драйверы для MySQL, а также SQLite, PostgreSQL и Microsoft SQL Server.

Наше следующее упражнение будет заключаться в установке MySQL на инструментальный компьютер. СУБД MySQL доступна для загрузки из всех Linux-репозиториев. Пользователям операционных систем Windows и Mac необходимо загрузить установщик с веб-сайта MySQL http://dev.mysql.com/downloads.

ПРИМЕЧАНИЕ

Пользователи Windows и Mac могут пропустить этот раздел, если выбрали установку сервера MySQL при установке Zend Server CE. Установщик Zend Server позволяет пользователям Windows и Mac загружать и устанавливать MySQL Server в процессе установки.

Время действовать — установка MySQL

Необходимо установить серверную и клиентскую части MySQL с помощью приведенных здесь шагов; в этой книге мы будем использовать MySQL в качестве основной базы данных.

1. В стандартной установке Ubuntu можно установить MySQL, выполнив в командном интерпретаторе следующую команду:

```
$ sudo apt-get install mysql-server mysql-client
```

2. После завершения установки MySQL-сервер запустится автоматически. Чтобы проверить, работает ли MySQL-сервер, выполните следующую команду:

```
$ sudo netstat -tap | grep mysql
```

3. Эта команда должна генерировать вывод, схожий со следующим и означающий, что MySQL-демон работает:

```
tcp 0 0 localhost:mysql *:* LISTEN 923/mysqld
```

4. Если MySQL-сервер по какой-либо причине не работает, то его можно принудительно запустить командой `restart`:

```
$ sudo service mysql restart
```

Что сейчас произошло?

Мы установили СУБД MySQL, а также подготовили стек LAMP1. Наш следующий шаг — создать базу данных на MySQL-сервере.

ПРИМЕЧАНИЕ

Поскольку мы используем Zend Server, нам не нужно устанавливать пакет php5-mysql. Если вы используете стек, в котором поддержка MySQL по умолчанию не предлагается, вам придется установить необходимые пакеты вручную.

Самостоятельная работа

После завершения работы с этим разделом попробуйте выполнить задание, описанное в следующем разделе.

Программа phpMyAdmin

Программа phpMyAdmin — это бесплатный веб-инструмент для управления базами данных с открытым исходным кодом, написанный на языке PHP. Программа phpMyAdmin предоставляет пользовательский веб-интерфейс для управления MySQL-сервером, для добавления и удаления баз данных, для управления базами данных, пользователями, привилегиями и др. В этой книге мы будем использовать phpMyAdmin в качестве административного интерфейса для управления нашими базами данных.

Теперь, когда мы установили Apache, PHP и MySQL, наш следующий шаг — создать пустую базу данных на MySQL-сервере.

Для этой цели нам нужно установить и сконфигурировать инструмент phpMyAdmin в программе Zend Server.

---

[1] Сокращение от Linux/Apache/PHP/MySQL. — *Примеч. перев.*

СОВЕТ

Программу phpMyAdmin можно загрузить с сайта http://www.phpmyadmin.net/ или установить из репозиториев вашей операционной системы.

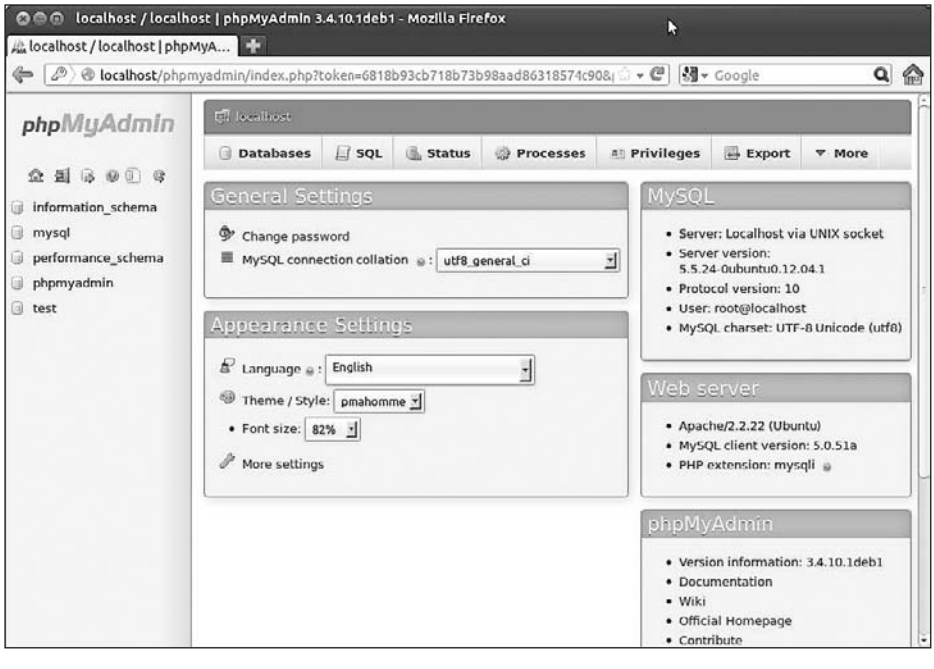
Инструкции по установке phpMyAdmin можно найти по ссылке http://docs.phpmyadmin.net/en/latest/setup.html.

В следующем упражнении мы создадим базу данных MySQL и пользователей на MySQL-сервере, после чего дадим этим пользователям права доступа для подключения к базе данных и выполнения операций с ней.

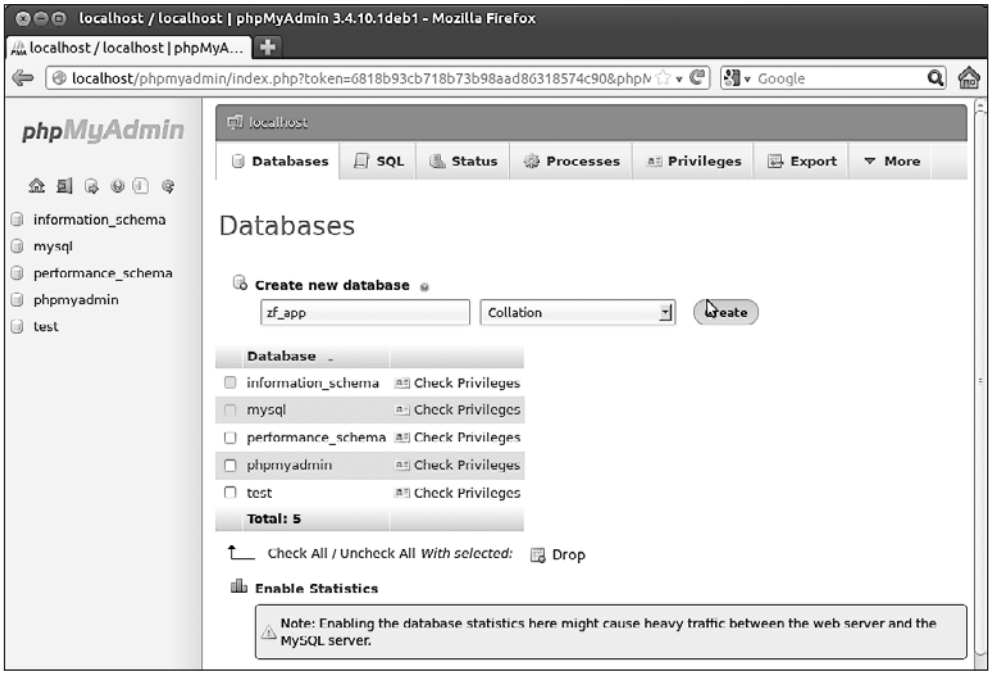
Время действовать — создание базы данных

Чтобы создать новую базу данных, откройте инструмент phpMyAdmin в веб-браузере и выполните следующие шаги.

1. Откройте phpMyAdmin в веб-браузере с помощью ссылки http://localhost/phpmyadmin.



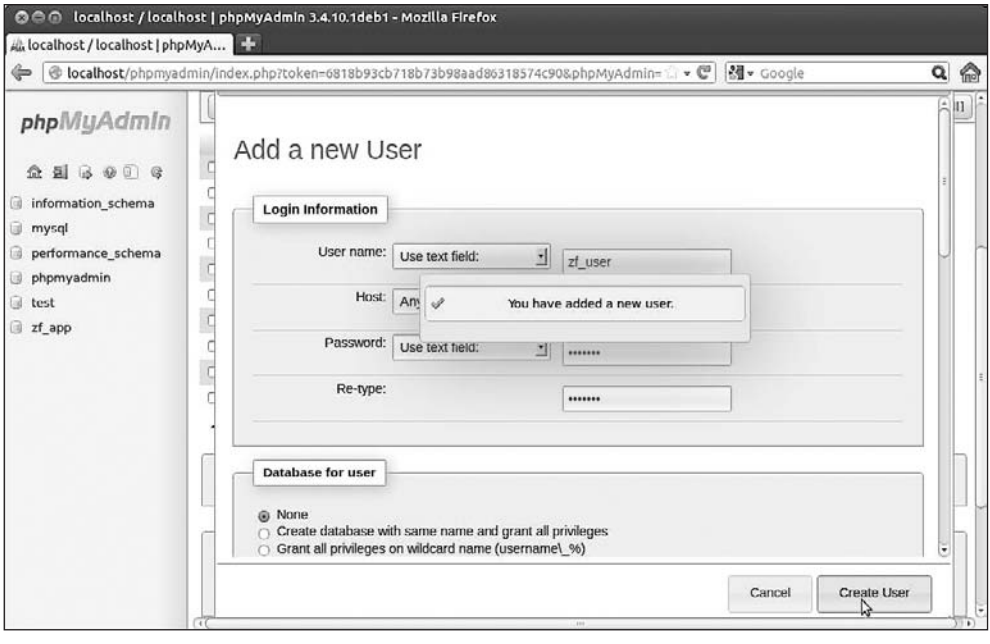
2. Перейдите в раздел Databases (Базы данных), введите имя новой базы данных `zf_app` в поле Create new database (Создать новую базу данных) и щелкните на кнопке Create (Создать).



3. После создания базы данных создайте пользователя для нее. Это можно сделать с помощью команды Add a new user (Добавить нового пользователя) в разделе Privileges (Привилегии). Укажите следующие параметры:

Поле	**Значение**
User name (Имя пользователя)	zf_user
Host (Хост)	localhost
Password (Пароль)	zf_pass

После этого ваш экран должен выглядеть следующим образом.

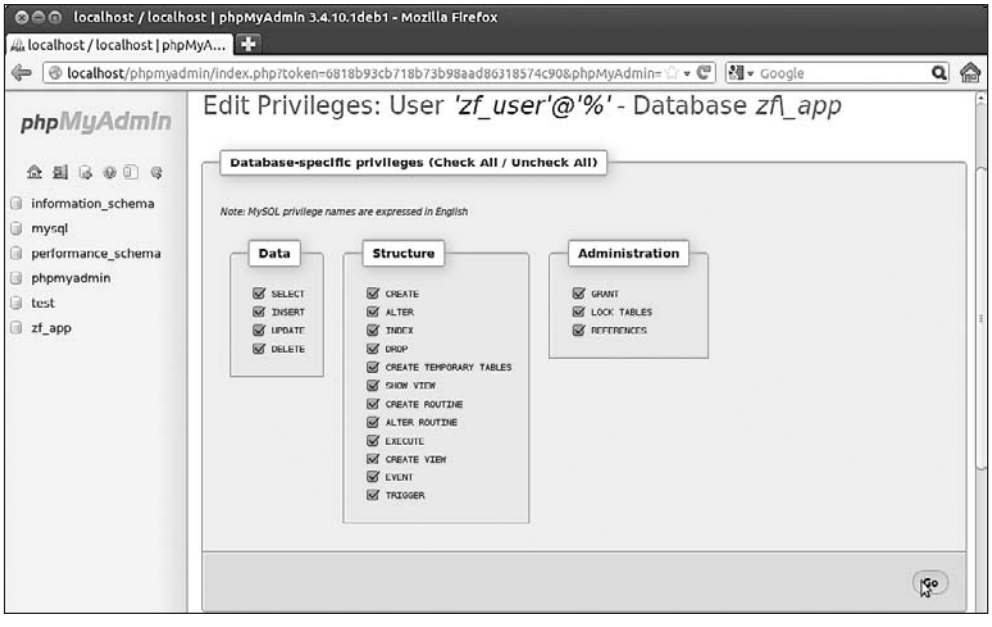


4. После создания пользователя перейдите в раздел Privileges (Привилегии) и выберите команду Edit Privileges (Редактировать привилегии) для пользователя `zf_user`.

5. В разделе Database-specific privileges (Привилегии базы данных) выберите базу данных `zf_app`.

6. Вы будете перенаправлены в раздел привилегий базы данных `zf_app` для пользователя `zf_user`. Выберите вариант Check All (Выбрать все) и щелкните на кнопке Go (Применить) (как показано на рисунке справа).

Теперь вы можете протестировать базу данных, выйдя из программы phpMyAdmin и затем войдя в нее с учетными данными пользователя `zf_user`. Вы должны увидеть единственную базу данных `zf_app`.



Что сейчас произошло?

Мы создали в MySQL нашу первую базу данных. Мы также создали в ней пользователя и назначили ему административные права для этой базы данных. Теперь мы сможем применять учетные данные этого пользователя в приложении, которое будем разрабатывать в следующих главах.

Самостоятельная работа

Теперь, когда у вас работает веб-сервер PHP-приложений и имеется база данных MySQL, создайте простую таблицу с названием Students и добавьте в нее несколько записей с помощью инструмента phpMyAdmin.

Ваша задача — создать простую веб-страницу на языке PHP, которая будет отображать все записи таблицы Students.

Контрольные вопросы

1. Какова минимальная версия PHP, необходимая для работы Zend Framework 2.0?

   1) PHP 4.3 и выше;

   2) PHP 5.2.0 и выше;

   3) PHP 5.3.3 и выше;

   4) PHP 5.4.7 и выше.

2. В каком каталоге по умолчанию расположен файл php.ini в установленной копии программы Zend Server?

   1) /home/<user>/etc/php/php.inc;

   2) /etc/php/php.ini;

   3) /var/www/php.ini;

   4) /usr/local/zend/etc/php.ini.

Заключение

В этой главе мы изучили вопросы установки и конфигурирования стека PHP-приложений программы Zend Server. Затем мы установили MySQL-сервер и создали нашу первую базу данных. В практической части мы научились устанавливать инструменты Git и phpMyAdmin.

В следующей главе мы познакомимся со структурой проекта Zend Framework и его ключевыми компонентами, такими как представления и контроллеры.

Создание первого приложения с помощью Zend Framework

2

В этой главе мы разработаем наш первый проект в Zend Framework 2.0. Мы рассмотрим ключевые аспекты разработки приложения с помощью Zend Framework 2.0, создавая модули, контроллеры и представления. Мы создадим в Zend Framework собственный модуль, который будем расширять в следующих главах этой книги.

Подготовка

Перед тем как приступать к созданию первого проекта в Zend Framework 2.0, удостоверьтесь, что в вашей среде разработки установлены и сконфигурированы следующие программы:

- командно-строковый интерфейс PHP;
- **Git** — программа необходима для работы с исходными кодами из различных репозитариев github.com;
- **Composer** — инструмент управления PHP-зависимостями.

Следующие команды пригодятся для установки инструментов, необходимых для настройки проекта Zend Framework 2.0.

- Для установки командно-строкового интерфейса PHP:

```
$ sudo apt-get install php5-cli
```

- Для установки системы Git:

```
$ sudo apt-get install git
```

- Для установки программы Composer:

```
$ curl -s https://getcomposer.org/installer | php
```

Приложение ZendSkeletonApplication

Проект ZendSkeletonApplication представляет собой приложение-заготовку, с помощью которого разработчики могут начать освоение Zend Framework 2.0. В приложении-заготовке используется поддерживаемый Zend Framework 2 паттерн модель-представление-контроллер (Model-View-Controller, MVC), в том числе новая модульная система. Приложение ZendSkeletonApplication можно загрузить с ресурса GitHub (https://github.com/zendframework/ZendSkeletonApplication).

Время действовать — создание проекта Zend Framework

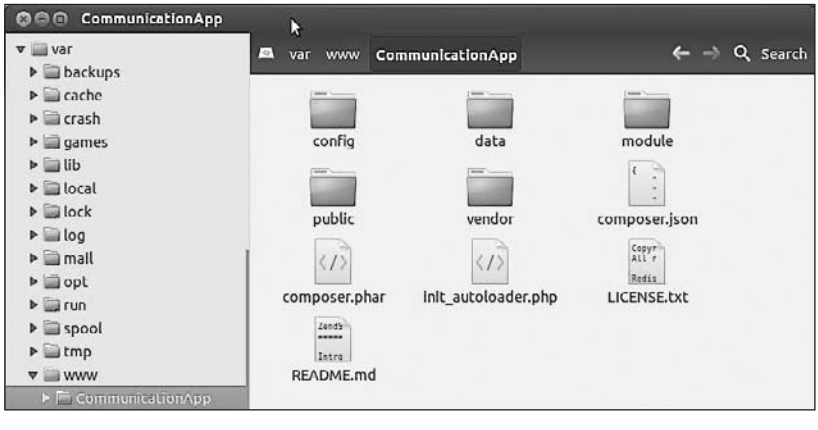
Чтобы настроить новый проект Zend Framework, нам потребуется загрузить последнюю версию приложения ZendSkeletonApplication и настроить виртуальный хост так, чтобы он указывал на новый проект Zend Framework. Выполните следующие шаги.

1. Перейдите в папку, в которой вы хотите разместить проект Zend Framework:

```
$ cd /var/www/
```

2. Выполните клонирование проекта ZendSkeletonApplication из GitHub:

```
$ git clone git://github.com/zendframework/
ZendSkeletonApplication.git CommunicationApp
```



СОВЕТ

В некоторых конфигурациях Linux текущий пользователь может не обладать правами доступа, требуемыми для записи в каталог /var/www. В таких случаях можно использовать любой каталог, доступный для записи, и вносить необходимые изменения в конфигурацию виртуального хоста.

3. Установите зависимости с помощью программы Composer:

```
$ cd CommunicationApp/
$ php composer.phar self-update
$ php composer.phar install
```

На следующем рисунке показано, как программа Composer загружает и устанавливает необходимые зависимости.

```
krishnav@ubuntu:/var/www/CommunicationApp$ php composer.phar self-update
Updating to version 172414a.
    Downloading: 100%
krishnav@ubuntu:/var/www/CommunicationApp$ php composer.phar install
Loading composer repositories with package information
Installing dependencies
  - Installing zendframework/zendframework (2.0.3)
    Downloading: 100%

zendframework/zendframework suggests installing doctrine/common (Doctrine\Common >=2
zendframework/zendframework suggests installing ext-intl (ext/intl for i18n features
zendframework/zendframework suggests installing pecl-weakref (Implementation of weak
zendframework/zendframework suggests installing zendframework/zendpdf (ZendPdf for c
zendframework/zendframework suggests installing zendframework/zendservice-recaptcha
as in Zend\Captcha and/or Zend\Form)
Writing lock file
Generating autoload files
krishnav@ubuntu:/var/www/CommunicationApp$
```

4. Перед тем как создать запись для виртуального хоста, мы должны создать запись для имени хоста в файле хостов, чтобы система всегда указывала на локальный компьютер при использовании нового имени хоста. В операционной системе Linux это можно сделать, добавив запись в файл /etc/hosts:

```
$ sudo vim /etc/hosts
```

ПРИМЕЧАНИЕ

В операционной системе Windows этот файл находится в папке %SystemRoot%\system32\drivers\etc\hosts.

5. Добавьте следующую строку в файл hosts:

```
127.0.0.1 comm-app.local
```

В результате файл hosts должен выглядеть так.

```
127.0.0.1       localhost
127.0.1.1       ubuntu
127.0.0.1       comm-app.local

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

6. На следующем шаге добавим запись виртуального хоста на наш веб-сервер; это можно сделать, создав конфигурационный файл нового виртуального хоста:

```
sudo vim /usr/local/zend/etc/sites.d/vhost_comm-app-80.conf
```

ПРИМЕЧАНИЕ

В вашем случае имя нового виртуального хоста может быть иным в зависимости от того, какой веб-сервер вы используете. Чтобы настроить новый виртуальный хост, обратитесь к документации вашего веб-сервера.

Например, если вы работаете с веб-сервером Apache2 в операционной системе Linux, то вам нужно создать файл нового виртуального хоста в каталоге /etc/apache2/sites-available и включить этот сайт командой a2ensite comm-app.local.

7. Добавьте следующую конфигурацию в файл виртуального хоста:

```
<VirtualHost *:80>
  ServerName comm-app.local
  DocumentRoot /var/www/CommunicationApp/public
  SetEnv APPLICATION_ENV "development"
  <Directory /var/www/CommunicationApp/public>
    DirectoryIndex index.php
    AllowOverride All
    Order allow,deny
    Allow from all
  </Directory>
</VirtualHost>
```

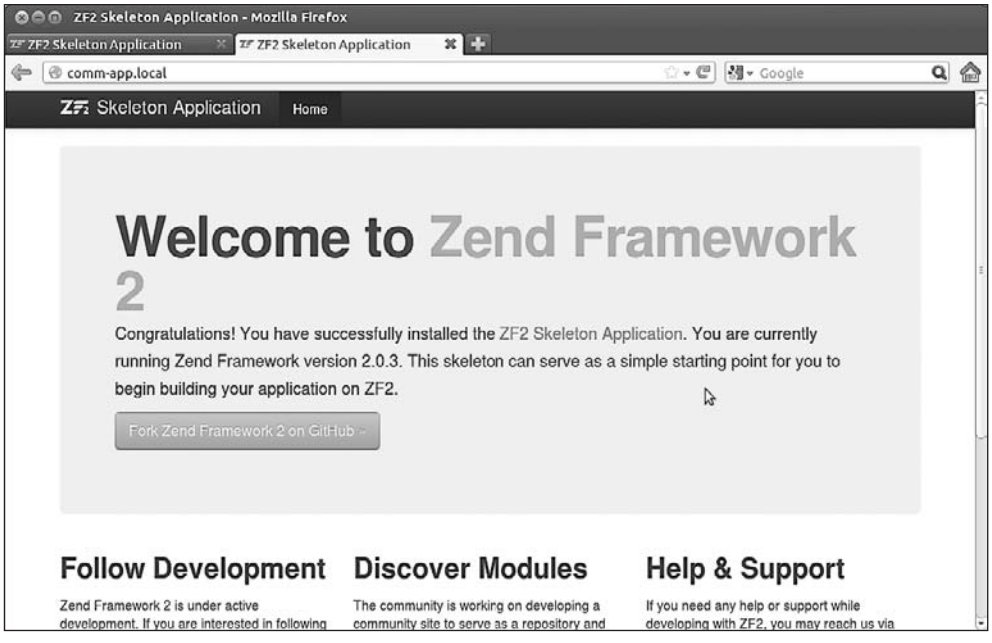
ПРИМЕЧАНИЕ

Если вы используете другой путь для работы с проектом ZendSkeletonApplication, удостоверьтесь в том, что этот путь используется как в директиве DocumentRoot, так и в директиве Directory.

8. После конфигурирования файла виртуального хоста необходимо перезапустить веб-сервер командой

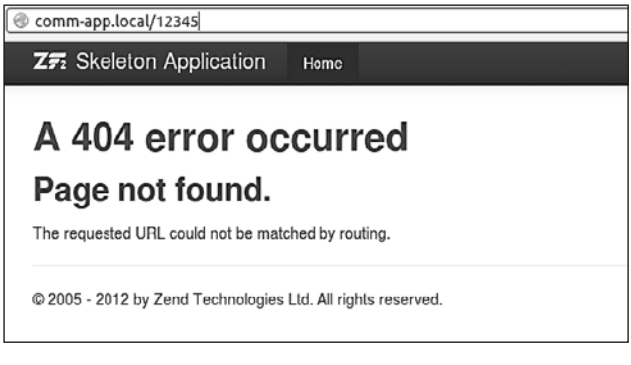
```
$ sudo service zend-server restart
```

9. После завершения установки у вас должна открываться в браузере ссылка http://comm-app.local, ведущая к следующей тестовой странице.



ПРОВЕРЯЙТЕ ПРАВИЛА ПЕРЕЗАПИСИ

Иногда модуль mod_rewrite по умолчанию может не подключаться в веб-браузере. Чтобы проверить корректность работы механизма перенаправления URL-адресов, попробуйте перейти по неправильному URL-адресу, например http://comm-app.local/12345. Если вы получите от Apache «ошибку 404», то правила перезаписи .htaccess не работают — их нужно исправить, в противном случае, получая страницу наподобие представленной на следующем рисунке, вы можете быть уверены, что URL-адрес работает должным образом.



Что сейчас произошло?

Мы успешно создали новый проект Zend Framework 2, воспользовавшись приложением ZendSkeletonApplication из ресурса GitHub, и загрузили с помощью программы Composer необходимые зависимости, в том числе для Zend Framework 2.0. Мы также создали конфигурацию виртуального хоста, указывающую на папку public проекта, и протестировали проект в веб-браузере.

ЗАГРУЗКА ПРИМЕРОВ КОДА

Все файлы с примерами кода для книг издательства Packt, приобретенных с помощью учетной записи на http://www.packtpub.com, доступны для загрузки. Если вы приобрели эту книгу в другом месте, то вы можете зарегистрироваться на сайте http://www.packtpub.com/support и получить файлы напрямую по электронной почте.

АЛЬТЕРНАТИВНЫЕ СПОСОБЫ УСТАНОВКИ

Мы рассмотрели лишь один из способов установки приложения ZendSkeletonApplication, но есть и другие.

Вы можете напрямую загрузить приложение-заготовку с помощью программы Composer и создать проект командой

```
$ php composer.phar create-project –repository-url="http://packages.zendframework.com" zendframework/skeleton-application путь/для/установки
```

Можно также воспользоваться рекурсивным клонированием программы Git для создания такого же проекта:

```
$ git clone git://github.com/zendframework/
ZendSkeletonApplication.git –recursive
```

С дополнительной информацией можно ознакомиться в документе http://framework.zend.com/downloads/skeleton-app.

Модули Zend Framework 2.0

В Zend Framework 2.0 модуль можно определить как единицу программного обеспечения, которую можно переносить, многократно использовать и связывать с другими модулями для создания более крупных и сложных приложений.

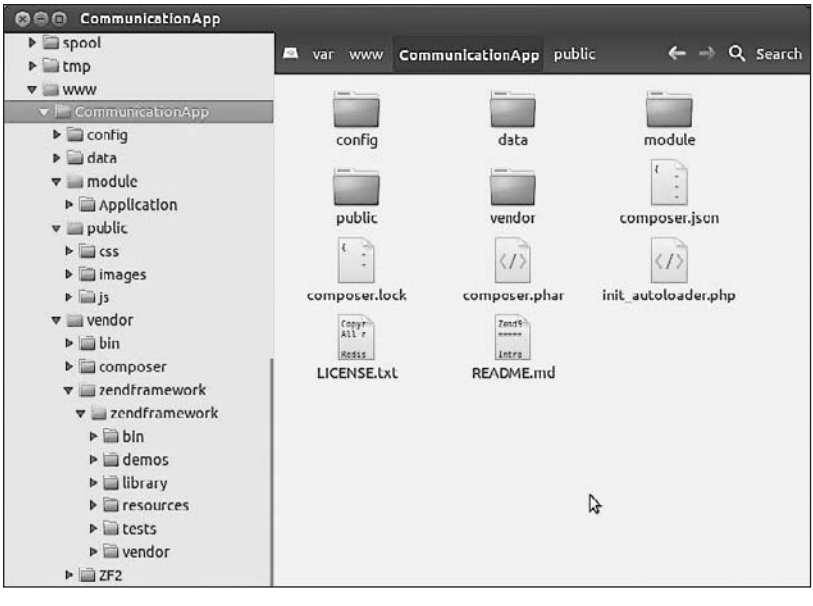
Модули — не новая концепция для Zend Framework, однако в Zend Framework 2 модули применяются совершенно особым образом. В Zend Framework 2 модули могут совместно использоваться несколькими системами, а действия по их переупаковке и распространению относительно просты. Еще одно существенное нововведение в Zend Framework 2 заключается в том, что теперь даже главное приложение преобразуется в модуль — модуль приложения.

Перечислим некоторые ключевые преимущества модулей Zend Framework 2.0:

- автономность, переносимость, возможность многократного использования;
- управление зависимостями;
- легковесность и быстродействие;
- поддержка упаковки с помощью программы Phar и распростанения с помощью инструмента Pyrus.

Структура папок проекта Zend Framework

Структура папок проекта Zend Framework 2 показана на рисунке.



Имя папки	Описание
config	Используется для управления конфигурацией приложения
data	Используется как временное хранилище для данных приложения, в том числе файлов кэша, файлов сеансов, журналов и индексов
module	Служит для управления всем кодом приложения
module/Application	Модуль приложения, предлагаемый по умолчанию; входит в состав ZendSkeletonApplication

Имя папки	Описание
public	Служит точкой входа в приложение и корневой папкой для документов веб-сайта. В ней хранятся все веб-ресурсы, в том числе файлы каскадных таблиц стилей, изображения и Java-сценарии
vendor	Применяется для управления общими библиотеками, которые использует приложение. Zend Framework также устанавливается в эту папку
vendor/zendframework	Zend Framework 2.0 устанавливается в эту папку

Время действовать — создание модуля

Цель наших следующих действий — создать новый модуль `Users` в Zend Framework 2.0. Этот модуль будет управлять пользователями, в том числе осуществлять их регистрацию, аутентификацию и т. д. Мы задействуем модуль `ZendSkeletonModule` из состава Zend Framework.

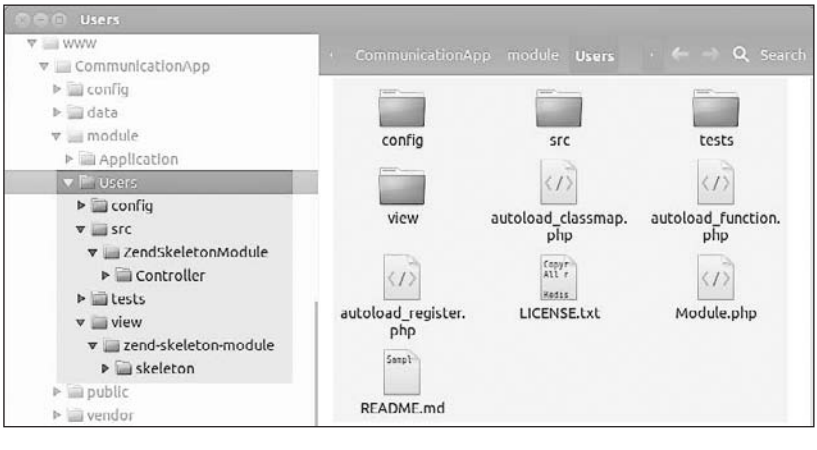
1. Перейдите в папку module приложения:

```
$ cd /var/www/CommunicationApp/
$ cd module/
```

2. Выполните клонирование модуля `ZendSkeletonModule`, назвав клон как сочтете нужным, в данном случае — `Users`:

```
$ git clone git://github.com/zendframework/ZendSkeletonModule.git Users
```

3. По окончании работы структура папок должна выглядеть следующим образом.



4. Отредактируйте файл Module.php, который будет расположен в папке Users раздела модулей (CommunicationApp/module/Users/module.php): измените пространство имен на `Users`. Замените инструкцию `namespace ZendSkeletonModule;` инструкцией `namespace Users;`.

5. Следующие папки можно удалить, поскольку мы не будем использовать их в нашем проекте:

   - Users/src/ZendSkeletonModule;
   - Users/view/zend-skeleton-module.

Что сейчас произошло?

Мы установили модуль-заготовку для Zend Framework; этот модуль пуст, и нам понадобится доработать его, создав собственные контроллеры и представления. Наши следующие действия будет направлены на создание новых контроллеров и представлений для этого модуля.

СОЗДАНИЕ МОДУЛЯ С ПОМОЩЬЮ ПРОГРАММЫ ZFTOOL

ZFTool — это утилита для управления приложениями и проектами Zend Framework; ее также можно использовать для создания новых модулей. Вам потребуется установить программу ZFTool и с ее помощью создать модуль командой create module:

```
$ php composer.phar require zendframework/zftool:dev-master
$ cd vendor/zendframework/zftool/
$ php zf.php create module Users2 /var/www/CommunicationApp
```

Дополнительную информацию об утилите ZFTool можно получить по ссылке http://framework.zend.com/manual/2.0/en/modules/zendtool.introduction.html.

Модель, представление, контроллер

Фундаментальная цель любого фреймворка, поддерживающего паттерн модель-представление-контроллер (Model-View-Controller, MVC), — упростить разделение трех уровней: модели, представления и контроллера. Перед тем как погрузиться в детали создания модулей, попробуем быстро разобраться в том, как эти три уровня работают в MVC-фреймворке.

- **Модель** является отражением данных. Модель также включает в себя бизнес-логику для различных транзакций приложений.
- **Представление** содержит в себе логику вывода на экран, которая используется для визуализации различных элементов пользовательского интерфейса в веб-браузере.

- **Контроллер** управляет прикладной логикой в любом MVC-приложении; все действия и события обрабатываются контроллером. Уровень контроллера служит коммуникационным интерфейсом между моделью и представлением, который управляет состоянием модели и отражает изменения в представлении. Контроллер также предоставляет точку входа для доступа к приложению.

В новой MVC-структуре Zend Framework 2 все модели, представления и контроллеры сгруппированы по модулям. Каждый модуль имеет свой набор моделей, представлений и контроллеров, используя некоторые компоненты совместно с другими модулями.

Структура папок модуля Zend Framework

Структура папок модуля Zend Framework 2.0 содержит три ключевых компонента — варианты конфигурации, логику модуля, а также представления. В следующей таблице описано, как организовано содержимое модуля:

Имя папки	Описание
config	Используется для управления конфигурацией модуля
src	Содержит весь исходный код модуля, в том числе все контроллеры и модели
view	Содержит все представления, используемые в модуле

Время действовать — создание контроллеров и представлений

Теперь, когда мы создали модуль, наш следующий шаг — определить собственные контроллеры и представления. В этом разделе мы создадим два простых представления и напишем контроллер, который будет обеспечивать переключение между ними.

1. Перейдите в папку внутри модуля:

```
$ cd /var/www/CommunicationApp/module/Users
```

2. Создайте папку для контроллеров:

```
$ mkdir -p src/Users/Controller/
```

3. Создайте новый файл IndexController в папке <имя модуля>/src/<имя модуля>/Controller/:

```
$ cd src/Users/Controller/
$ vim IndexController.php
```

4. Добавьте следующий код в файл IndexController:

```
<?php
namespace Users\Controller;
use Zend\Mvc\Controller\AbstractActionController;
use Zend\View\Model\ViewModel;
class IndexController extends AbstractActionController
{
  public function indexAction()
  {
    $view = new ViewModel();
    return $view;
  }
  public function registerAction()
  {
    $view = new ViewModel();
    $view->setTemplate('users/index/new-user');
    return $view;
  }
  public function loginAction()
  {
    $view = new ViewModel();
    $view->setTemplate('users/index/login');
    return $view;
  }
}
```

5. Этот код выполняет следующие действия: если пользователь посещает домашнюю страницу, то ему выводится представление, предлагаемое по умолчанию; если пользователь совершает действие `register`, то ему выводится шаблон `new-user`; если пользователь совершает действие `login`, то выводится шаблон `login`.

6. Теперь, когда мы создали контроллер, нам нужно создать требуемые представления, которые будут выводиться при каждом из действий контроллера.

7. Создайте папки для представлений:

```
$ cd /var/www/CommunicationApp/module/Users
$ mkdir -p view/users/index/
```

8. Перейдите в папку представлений <модуль>/view/<имя модуля>/index:

```
$ cd view/users/index/
```

9. Создайте следующие файлы представлений:

   - index;
   - login;
   - new-user.

   1) для создания файла view/users/index/index.phtml используйте следующий код:

```
<h1>Welcome to Users Module</h1>
<a href="/users/index/login">Login</a> | <a href="/users/
index/register">New User Registration</a>
```

   2) для создания файла /users/index/login.phtml используйте следующий код:

```
<h2> Login </h2>
<p> This page will hold the content for the login form </p>
<a href="/users"><< Back to Home</a>
```

   3) для создания файла view/users/index/new-user.phtml используйте следующий код:

```
<h2> New User Registration </h2>
<p> This page will hold the content for the registration
form </p>
<a href="/users"><< Back to Home</a>
```

Что сейчас произошло?

Сейчас мы создали новый контроллер и представления для нашего модуля в Zend Framework, но этот модуль пока что не готов к тестированию. Чтобы сделать его полностью функциональным, мы должны внести изменения в его конфигурацию и включить модуль в конфигурацию приложения.

Конфигурирование модуля Zend Framework

Конфигурация модуля Zend Framework 2.0 хранится в нескольких файлах, которые можно найти в модуле-заготовке. Вот описания некоторых конфигурационных файлов.

- Файл Module.php: менеджер модулей Zend Framework 2 ищет файл Module.php в корневой папке модуля. Менеджер модулей использует файл Module.php для конфигурирования модуля и вызывает методы `getAutoloaderConfig()` и `getConfig()`.

- Файл autoload_classmap.php: метод `getAutoloaderConfig()` в модуле-заготовке загружает файл autoload_classmap.php, чтобы включить любые нестандартные переопределения, отличные от классов, загруженных с помощью стандартного формата автозагрузчика. Управление этими нестандартными переопределениями можно осуществлять, добавляя и удаляя записи из файла autoload_classmap.php.

- Файл config/module.config.php: метод `getConfig()` загружает файл config/module.config.php; этот файл служит для конфигурирования различных параметров модуля, в том числе маршрутов, контроллеров, макетов и др.

Время действовать — изменение конфигурации модуля

В этом разделе мы внесем изменения в конфигурацию модуля `Users`, чтобы он мог работать с созданными контроллером и представлениями.

1. **Конфигурирование автозагрузчика**. Конфигурация автозагрузчика, предлагаемая по умолчанию модулем `ZendSkeletonModule`, должна быть отключена. Для этого можно отредактировать файл autoload_classmap.php, заменив его содержимое следующим:

```
<?php
return array();
```

2. **Конфигурирование модуля**. Конфигурацию модуля можно найти в файле config/module.config.php. Этот файл необходимо обновить, поместив в него информацию о созданных контроллерах и представлениях следующим образом.

   - **Контроллеры**. По умолчанию привязка контроллеров указывает на модуль `ZendSkeletonModule`; ее нужно заменить следующей:

```
'controllers' => array(
  'invokables' => array(
    'Users\Controller\Index' =>
    'Users\Controller\IndexController',
  ),
),
```

   - **Представления**. Представления модуля необходимо связать с их местоположениями. Обратите внимание на то, что имена представлений записаны в нижнем регистре и разделены дефисом (например, обращение к `ZendSkeleton` будет выглядеть как `zend-skeleton`):

```
'view_manager' => array(
  'template_path_stack' => array(
```

```
      'users' => __DIR__ . '/../view',
    ),
  ),
```

- **Маршруты**. Последнее действие по конфигурированию модуля — определение маршрута доступа к этому модулю из браузера. В данном случае мы определяем маршрут как `/users`, который указывает на действие `index` в контроллере `Index` модуля `Users`:

```
'router' => array(
  'routes' => array(
    'users' => array(
      'type' => 'Literal',
      'options' => array(
        'route' => '/users',
        'defaults' => array(
          '__NAMESPACE__' =>
          'Users\Controller',
          'controller' => 'Index',
          'action' => 'index',
        ),
      ),
```

3. После внесения в конфигурацию всех описанных изменений итоговый конфигурационный файл config/module.config.php должен выглядеть следующим образом:

```
<?php
return array(
  'controllers' => array(
    'invokables' => array(
      'Users\Controller\Index' =>
      'Users\Controller\IndexController',
    ),
  ),
  'router' => array(
    'routes' => array(
      'users' => array(
        'type' => 'Literal',
        'options' => array(
          // Измените в соответствии с вашим модулем
          'route' => '/users',
          'defaults' => array(
            // Задайте это значение согласно пространству
            // имен, в котором находятся ваши контроллеры
            '__NAMESPACE__' => 'Users\Controller',
            'controller' => 'Index',
```

```
                'action' => 'index',
              ),
            ),
            'may_terminate' => true,
            'child_routes' => array(
              // Это маршрут, предлагаемый по умолчанию. Его разумно
              // использовать при разработке модуля;
              // с появлением определенности в отношении
              // маршрутов для модуля, возможно, появится
              // смысл указать здесь более точные пути.
              'default' => array(
                'type' => 'Segment',
                'options' => array(
                  'route' =>
                  '/[:controller[/:action]]',
                  'constraints' => array(
                    'controller' =>
                    '[a-zA-Z][a-zA-Z0-9_-]*',
                    'action' =>
                    '[a-zA-Z][a-zA-Z0-9_-]*',
                  ),
                  'defaults' => array(),
                  ),
                ),
              ),
            ),
          ),
        ),
        'view_manager' => array(
          'template_path_stack' => array(
            'users' => __DIR__ . '/../view',
          ),
        ),
      );
```

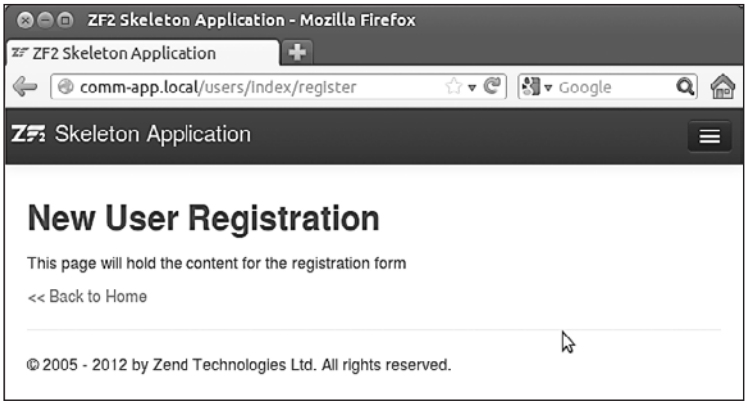
4. **Конфигурирование приложения**. Включите модуль в конфигурацию приложения: это можно сделать, изменив файл config/application.config.php и добавив модуль `Users` в список подключенных модулей:

```
'modules' => array(
  'Application',
  'Users',
),
```

5. Чтобы протестировать модуль, откройте ссылку http://comm-app.local/users/ в веб-браузере; вы должны иметь возможность перемещаться внутри модуля.

Домашняя страница модуля отображается следующим образом.

Страница регистрации выглядит следующим образом.



Что сейчас произошло?

Мы изменили конфигурацию модуля `ZendSkeletonModule` так, чтобы он работал с новым контроллером и представлениями, созданными для модуля `Users`. Теперь у нас есть полнофункциональный и работающий модуль, который использует новую модульную систему Zend Framework.

Самостоятельная работа

Теперь, когда мы знаем, как создавать и конфигурировать собственные модули, ваша следующая задача — настроить новый модуль `CurrentTime`. Предназначение этого модуля заключается в выводе текущих даты и времени в следующем формате:

Время: 14:00:00 GMT Дата: 12-Oct-2012

Контрольные вопросы

1. Какой инструмент используется приложением `ZendSkeletonApplication` для управления PHP-зависимостями?

   1) Git;

   2) Composer;

   3) Командно-строковый интерфейс PHP;

   4) Pyrus.

2. Как называется конфигурационный файл модуля?

   1) <Приложение>/module/<Модуль>/config.inc;

   2) <Приложение>/<Модуль>/config/config.php;

   3) <Приложение>/module/<Модуль>/module.config.php;

   4) <Приложение>/module/<Модуль>/config/module.config.php.

Заключение

Мы узнали о том, как сформировать проект Zend Framework с помощью заготовок приложения и модуля. В следующих главах мы сконцентрируемся на дальнейшей разработке этого модуля и расширении его возможностей до уровня полноценного приложения.

Создание коммуникационного приложения

3

В предыдущей главе мы рассмотрели создание контроллеров и представлений в новом модуле Zend Framework. В этой главе мы создадим нашу первую регистрационную форму и настроим механизм входа и аутентификации для зарегистрированных пользователей с помощью компонентов Zend Framework.

Вот некоторые ключевые компоненты, на которых мы сконцентрируем внимание в этой главе:

- `Zend\Form`;
- `Zend\InputFilter`;
- `Zend\Validator`;
- модели и `Zend\Db`.

Компонент Zend\Form

Обычно форма строится следующим образом: создается HTML-страница, пишутся отдельные процедуры проверки и фильтрации для различных событий формы, и наконец, создаются контроллеры и действия для работы с формой. В Zend Framework все перечисленные функциональные возможности обеспечиваются одним компонентом `Zend\Form`.

Компонент `Zend\Form` позволяет разработчикам создавать формы в приложениях программным путем. Он поддерживает вывод и обработку форм, фильтрацию и проверку ввода, а также конфигурирование форм. Нашей следующей задачей будет настроить нашу первую форму в Zend Framework 2.

Время действовать — создание регистрационной формы

Чтобы создать первую регистрационную форму, мы разработаем контроллер для ее вывода, а также новые формы и представления. Нам необходимо внести следующие изменения в модуль `Users`.

1. **Форма**. Нам понадобится создать регистрационную форму в файле src/Users/Form/RegisterForm.php.

   1) класс `RegisterForm` расширяет класс `Zend\Form\Form`; конфигурация формы добавляется в конструктор:

```
<?php
// имя файла : module/Users/src/Users/Form/RegisterForm.php
namespace Users\Form;
use Zend\Form\Form;
class RegisterForm extends Form
{
    public function __construct($name = null)
    {
        parent::__construct('Register');
        $this->setAttribute('method', 'post');
        $this->setAttribute('enctype','multipart/form-data');
```

   2) все поля добавляются в форму с помощью метода `$this->add()` в конструкторе формы:

```
$this->add(array(
    'name' => 'name',
    'attributes' => array(
        'type' => 'text',
    ),
    'options' => array(
        'label' => 'Full Name',
    ),
));
```

   3) к полям можно добавить дополнительные валидаторы/фильтры при объявлении полей в форме. В данном случае мы добавляем специальную проверку к полю `EmailAddress`:

```
$this->add(array(
    'name' => 'email',
    'attributes' => array(
        'type' => 'email',
    ),
    'options' => array(
```

```
            'label' => 'Email',
        ),
        'attributes' => array(
            'required' => 'required'
        ),
        'filters' => array(
            array('name' => 'StringTrim'),
        ),
        'validators' => array(
            array(
                'name' => 'EmailAddress',
                'options' => array(
                    'messages' => array(
                        \Zend\Validator\
EmailAddress::INVALID_FORMAT => 'Email address format is invalid'
                    )
                )
            )
        )
    ));
```

4) таким же методом добавьте поля `password`, `confirm_password` и `submit`; поля `password` и `confirm_password` будут иметь тип `password`, а поле `submit` — тип `button`.

2. **Представления**. Для поддержки процесса регистрации будет нужно создать следующие представления.

1) **регистрационная страница**. Представление для регистрационной страницы создается в файле src/view/users/register/index.phtml.

2) представление состоит из трех основных разделов: раздела для вывода сообщений об ошибках, раздела для логики представления, которая используется для генерации тега формы, и раздела помощников представления, генерирующих фактические элементы формы. Следующая логика служит для вывода сообщений об ошибках:

```
<section class="register">
<h2>Register</h2>
<?php if ($this->error): ?>
<p class="error">
    There were one or more issues with your submission.
    Please correct them as indicated below.
</p>
<?php endif ?>
```

3) следующий блок предназначен для генерации HTML-тега `<form>` при помощи объекта `form`, присвоенного представлению в контроллере:

```
<?php
$form = $this->form;
$form->prepare();
$form->setAttribute('action', $this->url(NULL,
array('controller'=>'Register', 'action' => 'process')));
$form->setAttribute('method', 'post');
echo $this->form()->openTag($form);
?>
```

4) следующий раздел служит для генерации отдельных элементов формы, соответствующих полям Name (Имя), Email (Электронная почта), Password (Пароль), Confirm Password (Подтвердить пароль) и Submit (Отправить):

```
<dl class="zend_form">
<dt><?php echo $this->formLabel($form->get('name')); ?></dt>
<dd><?php
    echo $this->formElement($form->get('name'));
    echo $this->formElementErrors($form->get('name'));
?></dd>
<dt><?php echo $this->formLabel($form->get('email')); ?>
</dt>
<dd><?php
    echo $this->formElement($form->get('email'));
    echo $this->formElementErrors($form->get('email'));
?></dd>
<dt><?php echo $this->formLabel($form->get('password'));
?></dt>
<dd><?php
    echo $this->formElement($form->get('password'));
    echo $this->formElementErrors($form->get('password'));
?></dd>
<dt><?php echo $this->formLabel($form->get('confirm_
password')); ?></dt>
<dd><?php
    echo $this->formElement($form->get('confirm_password'));
    echo $this->formElementErrors($form->get('confirm_
password'));
?></dd>
<dd><?php
    echo $this->formElement($form->get('submit'));
    echo $this->formElementErrors($form->get('submit'));
?></dd>
</dl>
```

5) наконец, HTML-тег `form` должен быть закрыт:

```
<?php echo $this->form()->closeTag() ?>
</section>
```

6) **страница подтверждения**. Представление для страницы подтверждения не отличается сложностью; оно создается в файле src/view/users/register/confirm.phtml:

```
<section class="register-confirm">
<h2>Register Sucessfull</h2>
<p> Thank you for your registration. </p>
</section>
```

3. **Контроллер**. Теперь, когда форма и представление готовы, наш следующий шаг заключается в создании контроллера, который поможет нам получить доступ к форме. Мы создадим новый класс `RegisterController` и загрузим новую форму в его методе `indexAction`. Новый контроллер создается в файле src/Users/Controller/RegisterController.php:

```
<?php
namespace Users\Controller;
use Zend\Mvc\Controller\AbstractActionController;
use Zend\View\Model\ViewModel;
use Users\Form\RegisterForm;
class RegisterController extends AbstractActionController
{
    public function indexAction()
    {
        $form = new RegisterForm();
        $viewModel = new ViewModel(array('form' =>$form));
        return $viewModel;
}
    public function confirmAction()
    {
        $viewModel = new ViewModel();
        return $viewModel;
    }
}
```

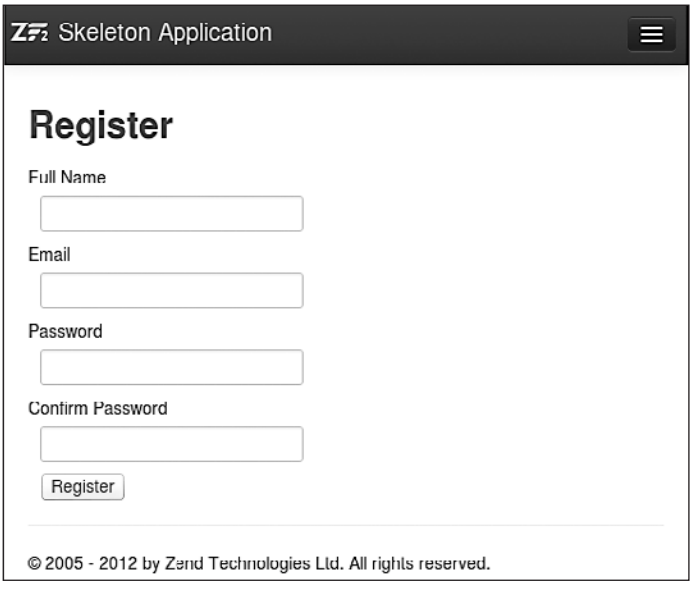
4. **Конфигурация**. Мы создали все компоненты, необходимые для вывода нашей формы на экран, теперь нам нужно добавить контроллер в список `invokables` в конфигурационном файле модуля (config/module.config.php):

```
'controllers' => array(
    'invokables' => array(
        'Users\Controller\Index' =>
        'Users\Controller\IndexController',
        'Users\Controller\Register' =>
        'Users\Controller\RegisterController',
),
```

5. Чтобы протестировать вывод регистрационной формы, откройте любой веб-браузер и попробуйте получить доступ к следующему URL-адресу:

```
http://comm-app.local/users/register
```

Регистрационная форма должна выглядеть следующим образом.



Что сейчас произошло?

К настоящему моменту мы создали форму со всеми полями, необходимыми для регистрации. Давайте разберемся, как происходит вывод этой формы. Когда мы вызываем страницу http://comm-app.local/users/register, контроллер создает новый экземпляр класса `RegisterForm` и показывает его в веб-браузере. Мы добавили следующие элементы управления к классу `RegisterForm` с помощью его конструктора:

- поле Name;
- поле Email;
- поле Password;
- поле Confirm Password;
- кнопку Submit.

Эти элементы управления были добавлены к новому объекту `Form`. Шаблон `ViewModel` выводит форму, объект `form` передается представлению для вывода, и каждый элемент управления показывается в соответствии с логикой представления помощником `FormElement`.

ПРИМЕЧАНИЕ

Метод FormElement действует как «волшебный помощник», выводящий любое поле формы с учетом передаваемого ему типа тега Zend\Form\Element. Для вывода конкретных полей формы существуют отдельные помощники. Полный список помощников представления формы можно найти в статье Form View Helpers в документации Zend Framework по адресу http://framework.zend.com/manual/2.0/en/modules/zend.form.view.helpers.html.

Самостоятельная работа

Перед тем как мы перейдем к следующему разделу, создайте форму входа в систему по аналогии с тем, как мы создавали форму регистрации. Форма должна содержать следующие элементы управления:

- поле Email;
- поле Password;
- кнопку Submit.

Мы воспользуемся этой формой для выполнения аутентификации ближе к концу текущей главы.

Валидация формы

Если вы внимательно изучили код формы, то обратили внимание на то, что мы добавили действия по проверке поля Email Address в следующем фрагменте кода:

```
'attributes' => array(
    'required' => 'required'
),
'filters' => array(
    array('name' => 'StringTrim'),
),
'validators' => array(
    array(
        'name' => 'EmailAddress',
```

```
        'options' => array(
            'messages' => array(
                \Zend\Validator\EmailAddress::INVALID_FORMAT =>
                'Email address format is invalid'
            )
```

Вот что именно мы добавили:

- атрибут, делающий поле обязательным (`required`);
- фильтр, обрезающий передаваемую строку;
- валидатор, проверяющий, что адрес электронной почты имеет корректный формат.

С появлением фильтра ввода в Zend Framework мы имеем возможность проверять форму целиком, а не каждое ее поле в отдельности. Это делает код значительно понятней и улучшает масштабируемость форм Zend Framework. Фактически мы можем использовать одну форму в нескольких разделах веб-сайта, каждый из которых имеет свой набор правил проверки, не зависящих от правил проверки формы. В следующем разделе мы настроим новый валидатор для формы регистрации.

Компонент Zend\InputFilter

Проверка форм и других средств ввода может выполняться с помощью компонента `Zend\InputFilter`. Этот компонент позволяет фильтровать и проверять произвольные наборы входных данных. Можно осуществлять проверку и фильтрацию конкретных элементов форм специально предназначенными для этого методами, но если требуется фильтровать данные, полученные с помощью запросов `$_GET` или `$_POST`, класс `InputFilter` позволит решить эту задачу.

В следующем упражнении мы добавим класс `InputFilter` в нашу регистрационную форму.

Время действовать — добавление в регистрационную форму механизма проверки

Чтобы добавить класс `InputFilter` в существующую форму, нам необходимо создать новый класс `InputFilter` и воспользоваться им для выполнения проверки при отправке формы.

1. Создайте новый класс `InputFilter` в файле src/Users/Form/RegisterFilter.php. Класс `RegisterFilter` расширит класс `Zend\InputFilter\InputFilter` и добавит все необходимые валидаторы в своем конструкторе:

```
<?php
namespace Users\Form;
use Zend\InputFilter\InputFilter;
class RegisterFilter extends InputFilter
{
    public function __construct()
    {
```

2. С помощью метода `$this->add()` мы можем включить в регистрационную форму различные средства фильтрации.

   1) для поля Email Address мы добавим валидатор, который проверяет, является ли введенное значение корректным адресом электронной почты:

```
$this->add(array(
    'name' => 'email',
    'required' => true,
    'validators' => array(
        array(
            'name' => 'EmailAddress',
            'options' => array(
                'domain' => true,
            ),
        ),
    ),
));
```

   2) для поля Name мы добавим валидатор, который ограничивает размер вводимых данных от 2 до 140 символов, и фильтр, вырезающий HTML-теги:

```
$this->add(array(
    'name' => 'name',
    'required' => true,
    'filters' => array(
    array(
            'name' => 'StripTags',
        ),
    ),
    'validators' => array(
        array(
            'name' => 'StringLength',
```

```
                'options' => array(
                    'encoding' => 'UTF-8',
                    'min' => 2,
                    'max' => 140,
                ),
            ),
        ),
    ));
```

3) для полей Password и Confirm Password мы не будем использовать какие-либо валидаторы, однако сделаем эти поля обязательными:

```
    'password' ));
            $this->add(array(
                'name' => 'confirm_password',
                'required' => true,
            ));
```

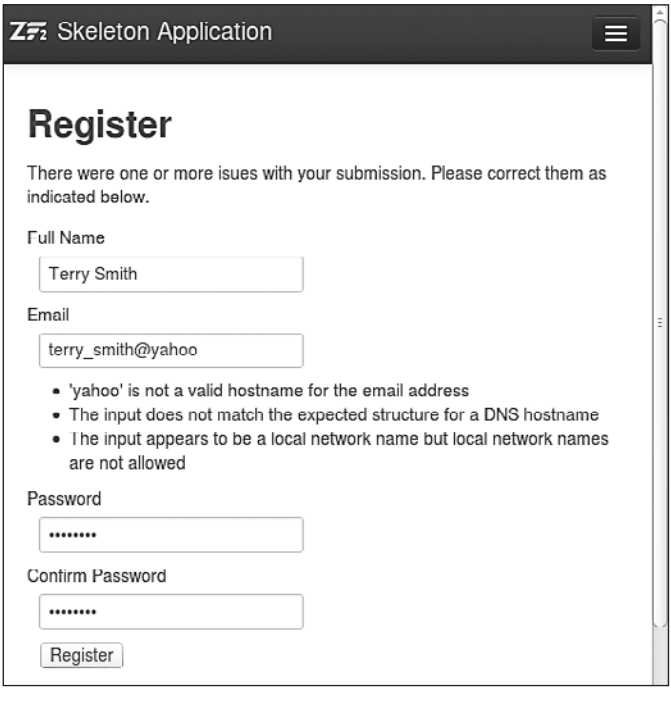
3. Класс `InputFilter` еще не отображен на класс `RegisterForm`; мы будем выполнять проверку при отправке формы. Нам нужно изменить класс `RegisterController`, включив в него метод `processAction` и выполнив проверку формы после ее отправки.

4. Измените класс `RegisterController`, включив в него метод `processAction`:

```
public function processAction()
{
    if (!$this->request->isPost()) {
        return $this->redirect()->toRoute(NULL ,
            array( 'controller' => 'register',
                'action' => 'index'
            ));
    }
    $post = $this->request->getPost();
    $form = new RegisterForm();
    $inputFilter = new RegisterFilter();
    $form->setInputFilter($inputFilter);
    $form->setData($post);
    if (!$form->isValid()) {
        $model = new ViewModel(array(
            'error' => true,
            'form' => $form,
        ));
        $model->setTemplate('users/register/index');
        return $model;
    }
```

```
        return $this->redirect()->toRoute(NULL , array(
            'controller' => 'register',
            'action' => 'confirm'
        ));
    }
```

5. Теперь откройте регистрационную страницу в веб-браузере и удостоверьтесь, что проверка выполняется.



Что сейчас произошло?

Мы только что реализовали проверку регистрационной формы. В функции `processAction()` класса `RegisterController` происходит создание нового экземпляра класса `RegisterForm` и применение фильтра `RegisterFilter` к форме с помощью метода `$form->setInputFilter()`. Данные, введенные в форму, добавляются еще раз, и выполняется их проверка методом `isValid()`. Сообщения об ошибках выводятся в форме с использованием помощника представления `FormElementErrors`.

Нам необходимо, чтобы при включении механизма проверки в класс `InputFilter` имена в этом классе правильно отображались на имена в форме.

Самостоятельная работа

В последнем упражнении вы познакомились с добавлением собственного класса `InputFilter` в форму Zend Framework. Перед тем как приступить к следующему разделу, реализуйте проверку `InputFilter` для формы `Login`, которую вы создали в предыдущем упражнении.

Модели и доступ к базам данных

Модели обеспечивают представление данных в MVC-приложении. В Zend Framework нет компонента `Zend\Model`, поэтому разработчикам приходится реализовывать модели самостоятельно. Сами по себе модели не могут обращаться к базам данных, извлекать или обрабатывать данные, поэтому они, как правило, связаны с объектами-преобразователями или используют объектно-реляционное отображение для подключения к базам данных. В этом примере мы будем использовать паттерн `TableGateway` для хранения данных в базе.

ПРИМЕЧАНИЕ

TableGateway — это паттерн баз данных, встроенный в Zend Framework и играющий роль шлюза для таблицы базы данных. Он имеет доступ ко всем строкам таблицы для выполнения различных SQL-операций, в том числе select, insert, update и delete.

Паттерн TableGateway

Паттерн `TableGateway` служит для создания объекта, представляющего таблицу базы данных. В этом примере объект `TableGateway` понадобится нам для таблицы `User`.

ВНИМАНИЕ

Если модель использует паттерн TableGateway для работы с содержимым базы данных, то в ней необходимо объявить метод exchangeArray().

Время действовать — создание моделей и сохранение формы

В этом упражнении мы сформируем модель для новых пользователей, создав таблицу в базе данных MySQL для хранения регистрационных данных с помощью паттерна `TableGateway`. В конце мы подключим нашу регистрационную форму к таблице `UserTable`, чтобы новые регистрационные данные сохранялись в базе.

1. Для того чтобы хранить регистрационную информацию в базе данных MySQL, необходимо создать новую таблицу:

```
CREATE TABLE user (
    id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    name TEXT NOT NULL,
    email VARCHAR(255) NOT NULL,
    password TEXT NOT NULL,
    PRIMARY KEY (id),
    UNIQUE INDEX idx_email(email)
);
```

2. Чтобы добавить ссылки на подключение к базе данных, требуется изменить глобальную конфигурацию приложения, как показано в следующем фрагменте кода в файле <домашний_каталог_приложения>/config/autoload/global.php:

```
return array(
    'db' => array(
        'driver' => 'Pdo',
        'dsn' => 'mysql:dbname=test;host=localhost',
        'username' => 'db_user',
        'password' => '',
        'driver_options' => array(
            PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES \'UTF8\''
        ),
    ),
    'service_manager' => array(
        'factories' => array(
            'Zend\Db\Adapter\Adapter'
                => 'Zend\Db\Adapter\AdapterServiceFactory',
        ),
    ),
);
```

3. Создайте новую модель для класса `User` в файле src/Users/Model/User.php:

```
<?php
namespace Users\Model;
class User
{
    public $id;
    public $name;
    public $email;
    public $password;
}
```

4. В модели `User` определяются методы `setPassword()` и `exchangeArray()`.

1) реализуйте метод `setPassword()`, который будет назначать пароль версии MD5 для сущности `UserTable`:

```
public function setPassword($clear_password)
{
    $this->password = md5($clear_password);
}
```

2) реализуйте метод `exchangeArray()`, предназначенный для отображения сущности `User` на сущность `UserTable`:

```
function exchangeArray($data)
{
    $this->name = (isset($data['name'])) ?
    $data['name'] : null;
    $this->email = (isset($data['email'])) ?
    $data['email'] : null;
    if (isset($data["password"]))
    {
        $this->setPassword($data["password"]);
    }
}
```

5. Создайте новую ссылку на таблицу для `User` в файле src/Users/Model/UserTable.php:

```
<?php
namespace Users\Model;
use Zend\Db\Adapter\Adapter;
use Zend\Db\ResultSet\ResultSet;
use Zend\Db\TableGateway\TableGateway;
class UserTable
{
    protected $tableGateway;
    public function __construct(TableGateway $tableGateway)
    {
        $this->tableGateway = $tableGateway;
    }
    public function saveUser(User $user)
    {
        $data = array(
            'email' => $user->email,
            'name' => $user->name,
            'password' => $user->password,
        );
        $id = (int)$user->id;
        if ($id == 0) {
            $this->tableGateway->insert($data);
```

```
            } else {
                if ($this->getUser($id)) {
                    $this->tableGateway->update($data, array('id' => $id));
                } else {
                    throw new \Exception('User ID does not exist');
                }
            }
        }
        public function getUser($id)
        {
            $id = (int) $id;
            $rowset = $this->tableGateway->select(array('id' => $id));
            $row = $rowset->current();
            if (!$row) {
                throw new \Exception("Could not find row $id");
            }
            return $row;
        }
    }
```

6. Теперь можно использовать таблицу `UserTable` для сохранения новых регистрационных данных в базе. Чтобы регистрационные данные сохранялись, мы должны внести изменения в класс `RegisterController`. Сначала мы создадим новую функцию для сохранения регистрационных данных пользователя:

```
protected function createUser(array $data)
{
    $sm = $this->getServiceLocator();
    $dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');
    $resultSetPrototype = new \Zend\Db\ResultSet\ResultSet();
    $resultSetPrototype->setArrayObjectPrototype(new
        \Users\Model\User);
    $tableGateway = new \Zend\Db\TableGateway\TableGateway('user',
        $dbAdapter, null, $resultSetPrototype);
    $user = new User();
    $user->exchangeArray($data);
    $userTable = new UserTable($tableGateway);
    $userTable->saveUser($user);
    return true;
}
```

ПРИМЕЧАНИЕ

Конструктор TableGateway принимает следующие параметры и генерирует объект типа TableGateway:

$table — используется для указания имени таблицы для объекта TableGateway;

Adapter $adapter — служит для указания имени адаптера базы данных;

$features (необязательный) — это API паттерна TableGateway, позволяющий расширять его функциональные возможности, не расширяя при этом базовый класс (здесь можно перечислить дополнительные функциональные возможности);

ResultSet $resultSetPrototype (необязательный) — служит для указания типа ResultSet;

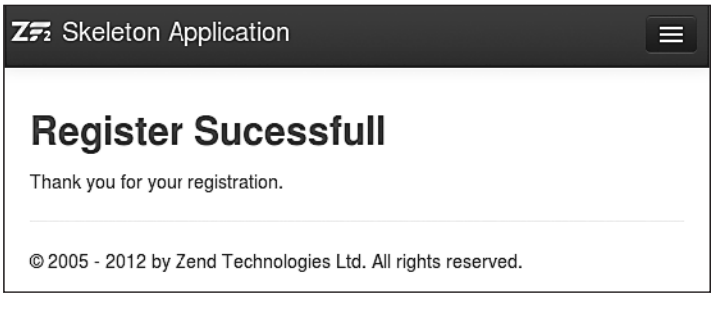
Sql $sql (необязательный) — предназначен для указания дополнительных SQL-критериев; проверьте, что SQL-объект связан с той же таблицей, что и в параметре $table.

Дополнительную информацию можно получить в документе http://framework.zend.com/manual/2.0/en/modules/zend.db.table-gateway.html#zend-dbtablegateway.

7. Далее мы должны вызвать эту функцию в методе `processAction()` до перенаправления на страницу подтверждения:

```
// Создание пользователя
$this->createUser($form->getData());
```

8. Откройте регистрационную страницу в браузере по вашему выбору и воспользуйтесь базой данных MySQL, чтобы проверить, что регистрационные данные корректно сохранены в базе данных. Страница подтверждения регистрации должна выглядеть так, как показано на рисунке.



Вы можете проверить правильность вставки записей через базу данных MySQL.

```
mysql> SELECT id, name, email, password FROM user;
+----+-------------+----------------------+----------------------------------+
| id | name        | email                | password                         |
+----+-------------+----------------------+----------------------------------+
|  1 | Terry Smith | terry.smith@email.com | 5f4dcc3b5aa765d61d8327deb882cf99 |
|  2 | John Smith  | john.smith@email.com  | 5f4dcc3b5aa765d61d8327deb882cf99 |
|  3 | Mani Raj    | mani.raj@email.com    | 5f4dcc3b5aa765d61d8327deb882cf99 |
|  4 | Test User   | test.user@email.com   | 5f4dcc3b5aa765d61d8327deb882cf99 |
+----+-------------+----------------------+----------------------------------+
4 rows in set (0.00 sec)

mysql>
```

Что сейчас произошло?

Мы изменили форму так, чтобы регистрационные данные новых пользователей сохранялись в базе данных. Наш следующий шаг будет заключаться в настройке аутентификации на основе информации, содержащейся в базе данных.

Компонент Zend\Authentication

Компонент `Zend\Authentication` входит в состав Zend Framework и может быть использован для аутентификации с помощью разнообразных механизмов, в том числе таблицы базы данных, а также инструментов HTTP- и LDAP-аутентификации. Этот компонент также позволяет помещать информацию о сеансах в разнообразные хранилища.

В этом примере мы воспользуемся компонентом `Zend\Authentication` для проверки учетных данных пользователя, отправленных через форму входа в систему.

Время действовать — аутентификация пользователя

В этом задании мы проведем аутентификацию формы входа в систему с помощью компонента `Zend\Authentication`.

1. Добавьте в контроллер входа в систему (файл Users/Controller/LoginController.php) функцию, которая возвращает службу аутентификации:

```
// Ссылки
use Zend\Authentication\AuthenticationService;
use Zend\Authentication\Adapter\DbTable as DbTableAuthAdapter;
// Определение класса
public function getAuthService()
{
    if (! $this->authservice) {
        $dbAdapter = $this->getServiceLocator()->get(
            'Zend\Db\Adapter\Adapter');
        $dbTableAuthAdapter = new DbTableAuthAdapter(
            $dbAdapter,'user','email','password', 'MD5(?)');
        $authService = new AuthenticationService();
        $authService->setAdapter($dbTableAuthAdapter);
        $this->authservice = $authService;
    }
    return $this->authservice;
}
```

2. В методе `processAction()` контроллера `LoginController` проверьте корректность данных формы и воспользуйтесь методом `AuthService`, чтобы проверить учетные данные методом `authenticate`:

```
public function processAction()
//
$this->getAuthService()->getAdapter()->setIdentity(
    $this->request->getPost('email'))->setCredential(
    $this->request->getPost('password'));
$result = $this->getAuthService()->authenticate();
if ($result->isValid()) {
    $this->getAuthService()->getStorage()->write(
        $this->request->getPost('email'));
    return $this->redirect()->toRoute(NULL , array(
        'controller' => 'login',
        'action' => 'confirm'
    ));
}
```

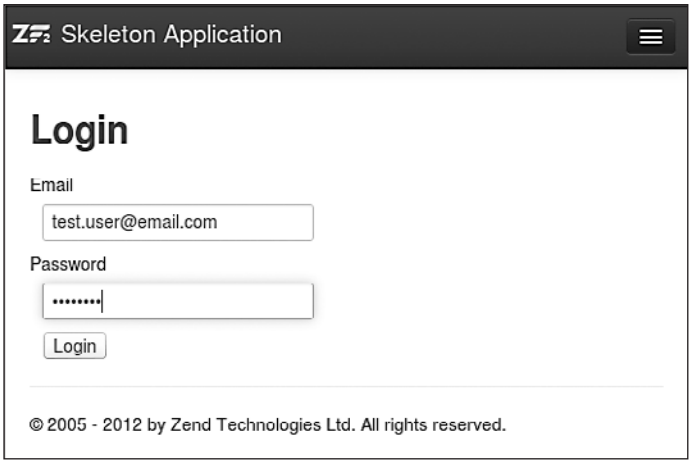
3. Функция `ConfirmAction` выводит на экран приветствие для вошедшего пользователя:

```
public function confirmAction()
{
    $user_email = $this->getAuthService()->getStorage()->read();
    $viewModel = new ViewModel(array(
        'user_email' => $user_email
    ));
    return $viewModel;
}
```

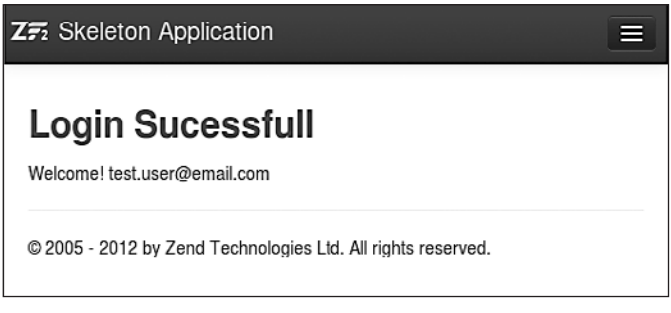
4. Представление домашней страницы пользователя в файле /view/users/login/confirm.phtml будет выглядеть следующим образом:

```
<section class="login-confirm">
<h2>Login Successful</h2>
<p> Welcome! <?php echo $this->user_email; ?> </p>
</section>
```

5. Откройте в браузере страницу входа в систему и попытайтесь войти с теми учетными данными, которые вы указали при регистрации. Форма входа должна выглядеть так.



После успешного входа в систему вы будете перенаправлены на следующую страницу.



Что сейчас произошло?

Мы создали новый адаптер аутентификации с использованием таблицы `user` базы данных, который проверяет поля `email` и `password`. С помощью этого адаптера мы смогли провести аутентификацию зарегистрировавшихся пользователей.

Контрольные вопросы

1. В каком файле необходимо сохранить учетную информацию для подключения к базе данных, чтобы она была доступна во всем приложении?

   1) <Приложение>/module/<Модуль>/config.inc;

   2) <Приложение>/config/autoload/global.php;

3) <Приложение>/module/<Модуль>/module.config.php;

4) <Приложение>/module/<Модуль>/config/module.config.php.

2. Какой метод связывает фильтр ввода с формой?

1) `$form->setInputFilter($inputFilter)`;

2) `$form->useInputFilter($inputFilter)`;

3) `$form->assignInputFilter($inputFilter)`;

4) `$form->mapInputFilter($inputFilter)`.

Заключение

В этой главе мы изучили вопросы создания форм, выполнения простых проверок, сохранения данных форм в базе данных, использования моделей и аутентификации с помощью базы данных. В следующей главе мы познакомимся с более сложными операциями над базами данных с применением паттерна `TableGateway`, рассмотренного в этой главе.

Управление данными и совместное использование документов

4

Теперь, когда вы готовы создавать собственные базовые модели, основываясь на знаниях из предыдущих глав, в этой главе вы можете научиться с максимальной отдачей использовать концепции управления данными и файлами в Zend Framework.

Вот основные темы этой главы:

- менеджер служб Zend Framework 2;
- паттерн `TableGateway`;
- выгрузка и совместное использование файлов в Zend Framework.

Менеджер служб Zend Framework 2

Менеджер служб Zend Framework 2 реализует паттерн локатора служб. Локатор служб представляет собой локатор «служба/объект», который используется для получения других объектов.

Варианты конфигурации менеджера служб охватывают 6 основных категорий; конфигурация вашего приложения/модуля может соответствовать одной или нескольким категориям, перечисленным в следующей таблице.

Тип конфигурации	**Описание**
abstract_factories	Определение массива абстрактных классов
aliases	Определение массива связанных пар «псевдоним/целевое имя»
factories	Определения массива пар «имя службы/класс фабрики». Классы фабрики, определенные здесь, должны реализовывать интерфейс Zend/ServiceManager/FactoryInterface либо вызываемые классы

Тип конфигурации	Описание
invokables	Определение массива пар «имя службы/имя класса». Экземпляры перечисленных здесь классов могут создаваться непосредственно, без каких-либо аргументов конструктора
services	Определение массива пар «имя службы/имя объекта». Служба по сути является экземпляром класса. Службы могут использоваться для регистрации классов, экземпляры которых уже созданы
shared	Определение массива пар «имя службы/логическое значение», указывающих на то, должна ли служба быть общедоступной. По умолчанию все службы являются общедоступными, однако в менеджере служб имеется параметр, с помощью которого можно отключить режим совместного использования указанных служб

Конфигурация менеджера служб может храниться с конфигурационными данными приложения или модуля; вы можете выбрать один из этих двух вариантов, исходя из конкретных потребностей и особенностей приложения или модуля. Как правило, конфигурация, являющаяся статической в масштабах приложения, хранится с конфигурационными данными уровня приложения; вся остальная информация хранится на уровне модуля.

Слияние частей, на которые разделены конфигурационные данные менеджера служб, осуществляется в следующем порядке.

1. Конфигурационные данные модуля, предоставляемые классом `Module`, с помощью метода `getServiceConfig()`. Они обрабатываются в том же порядке, что и модули:

```
public function getServiceConfig()
{
  return array(
    'abstract_factories' => array(),
    'aliases' => array(),
    'factories' => array(),
    'invokables' => array(),
    'services' => array(),
    'shared' => array(),
  );
}
```

2. Конфигурационные данные модуля в ключе `service_manager`; его обработка происходит в том же порядке, в котором обрабатываются модули.

3. Конфигурационные данные приложения находятся в различных конфигурационных файлах каталога config/autoload/ в порядке их обработки:

```
<?php
return array(
  'service_manager' => array(
    'abstract_factories' => array(),
    'aliases' => array(),
    'factories' => array(),
    'invokables' => array(),
    'services' => array(),
    'shared' => array(),
  ),
);
```

Время действовать — перенос существующего кода в менеджер служб

Наш следующий шаг — изменить блоки существующего кода так, чтобы они использовали менеджер служб. Вот некоторые ключевые фабрики, которые можно перенести в менеджер служб:

- подключения к базам данных;
- шлюзы к моделям и таблицам;
- формы и фильтры;
- служба аутентификации.

Если вы изучите существующий код, то увидите, что все подключения к базам данных уже используют модель менеджера служб Zend Framework 2 для хранения учетных данных. Мы сделаем шаг вперед и перенесем остальные фабрики в менеджер служб с помощью следующих действий.

1. Измените файл Module.php и добавьте новую функцию, загружающую конфигурационные данные менеджера служб:

```
public function getServiceConfig()
{
  return array(
    'abstract_factories' => array(),
    'aliases' => array(),
    'factories' => array(

      // база данных
      'UserTable' => function($sm) {
        $tableGateway = $sm->get('UserTableGateway');
        $table = new UserTable($tableGateway);
```

```
            return $table;
          },
          'UserTableGateway' => function ($sm) {
            $dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');
            $resultSetPrototype = new ResultSet();
            $resultSetPrototype->setArrayObjectPrototype(new User());
            return new TableGateway('user', $dbAdapter, null,
            $resultSetPrototype);
          },

          // Формы
          'LoginForm' => function ($sm) {
            $form = new \Users\Form\LoginForm();
            $form->setInputFilter($sm->get('LoginFilter'));
            return $form;
          },
          'RegisterForm' => function ($sm) {
            $form = new \Users\Form\RegisterForm();
            $form->setInputFilter($sm->get('RegisterFilter'));
            return $form;
          },

          // Фильтры
          'LoginFilter' => function ($sm) {
            return new \Users\Form\LoginFilter();
          },
          'RegisterFilter' => function ($sm) {
            return new \Users\Form\RegisterFilter();
          },
        ),
        'invokables' => array(),
        'services' => array(),
        'shared' => array(),
      );
    }
```

2. Убедитесь, что файл Module.php включает в себя все необходимые пространства имен:

```
use Users\Model\User;
use Users\Model\UserTable;

use Zend\Db\ResultSet\ResultSet;
use Zend\Db\TableGateway\TableGateway;
```

3. Внесите необходимые изменения в контроллеры, чтобы получить экземпляры из менеджера служб:

```
// получение формы входа в систему
$form = $this->getServiceLocator()->get('LoginForm');
// получение таблицы пользователей
$userTable = $this->getServiceLocator()->get('UserTable');
```

ИСПОЛЬЗОВАНИЕ ПРОСТРАНСТВ ИМЕН

Пространства имен можно использовать с помощью ключевых слов namespace и use, принятых в языке PHP 5.3. Все классы Zend Framework 2 имеют пространство имен, которое в точности соответствует структуре папок внутри папки, содержащей класс. Все классы, хранимые внутри этой папки, определяются непосредственно по их пространству имен.

По умолчанию ключевое слово use создает псевдоним последнему сегменту пространства имен, но этот режим можно изменить, используя параметр as вместе с ключевым словом. Пример:

```
use Zend\Form\Element as Element;

use Zend\Form\Element; // то же самое, что в предыдущей строке
```

4. Чтобы проверить, что изменения функционируют надлежащим образом, попробуйте зарегистрироваться и войти в систему с новыми учетными данными.

Что сейчас произошло?

Мы изменили написанный нами код так, чтобы он использовал фреймворк менеджера служб. Менеджер служб обладает огромным преимуществом с точки зрения чистоты кода, обеспечивает высокоэффективное масштабирование и служит централизованным реестром для ключевых компонентов приложения.

Самостоятельная работа

Теперь, когда вы понимаете, как функционирует фреймворк менеджера служб, вам предстоит решить несложную задачу. Контроллер входа в систему (CommunicationApp/module/Users/src/Users/Controller/LoginController.php) использует функцию `getAuthService()` для определения службы аутентификации. Измените эту функцию так, чтобы она получала службу аутентификации от менеджера служб.

Операции с базами данных

В предыдущей главе мы узнали, как реализовать простую операцию с базой данных, а именно — вставку таблицы (`table insert`). В этом разделе вы узнаете обо

всех основных действиях с базами данных, необходимых для создания простого CRUD[1]-интерфейса.

Еще немного о классе TableGateway

Класс `TableGateway` расширяет класс `AbstractTableGateway`, который реализует интерфейс `TableGatewayInterface`. Определение интерфейса `TableGatewayInterface` приведено в следующем фрагменте кода; все основные операции с таблицей определены в интерфейсе.

```
interface Zend\Db\TableGateway\TableGatewayInterface
{
  public function getTable();
  public function select($where = null);
  public function insert($set);
  public function update($set, $where = null);
  public function delete($where);
}
```

Класс `TableGateway` предоставляет широкий круг методов для выполнения основных операций с базами данных; пояснения о некоторых наиболее применяемых методах приведены далее.

- `getTable()`: возвращает строку, которая содержит имя таблицы, связанное с объектом `TableGateway`. Пример:

```
$myTableName = $myTableGateway->getTable();
```

- `select($where = null)`: используется для выбора множества строк, соответствующих критерию, указанному в параметре `$where`; таким критерием может быть условие `where` на основе `Zend\Db\Sql\Where` или массив критериев. Пример:

```
$rowset = $myTableGateway->select( array('id' => 2));
```

- `insert($set)`: служит для вставки данных, определенных в параметре `$set`, в таблицу в виде новой записи. Пример:

```
$myTableGateway->insert( array('id' => 2, 'name'=>'Ravi'));
```

- `update($set, $where = null)`: позволяет обновить множество строк, соответствующих критерию, который указан в параметре `$where`; таким критерием

[1] Сокращение от Create, Read, Update, Delete (создание, чтение, обновление, удаление). — *Примеч. перев.*

может быть условие `where` на основе `Zend\Db\Sql\Where` или массив критериев. Параметр `$set` содержит данные, которые будут обновлены во всех записях, соответствующих параметру `$where`. Пример:

```
$rowset = $myTableGateway->update(array('name' => 'Jerry'),
  array('id' => 2));
```

- `delete($where)`: применяется для удаления множества строк, соответствующих критерию, который указан в параметре `$where`; таким критерием может быть условие `where` на основе `Zend\Db\Sql\Where` или массив критериев. Пример:

```
$myTableGateway->delete( array('id' => 2));
```

- `getLastInsertValue()`: возвращает последнее вставленное значение для первичного ключа таблицы. Тип возвращаемого значения — целочисленный. Пример:

```
$myTableGateway->insert( array('name'=>'Ravi'));
$insertId = $myTableGateway-> getLastInsertValue ();
```

Время действовать — реализация административного интерфейса для управления пользователями

В этом упражнении мы создадим в нашем приложении административный пользовательский интерфейс для управления пользователями. Действия, которые мы реализуем, включают в себя перечисление всех пользователей, редактирование данных существующих пользователей, удаление и добавление пользователей.

1. Измените файл CommunicationApp/module/Users/src/Users/Model/UserTable.php с помощью приведенного далее кода, добавив следующие функции:

   - `fetchAll()`;
   - `getUser($id)`;
   - `getUserByEmail($userEmail)`;
   - `deleteUser($id)`.

```
public function fetchAll()
{
  $resultSet = $this->tableGateway->select();
  return $resultSet;
```

```
}
public function getUser($id)
{
  $id = (int) $id;
  $rowset = $this->tableGateway->select(array('id' => $id));
  $row = $rowset->current();
  if (!$row) {
    throw new \Exception("Could not find row $id");
  }
  return $row;
}
public function getUserByEmail($userEmail)
{
  $rowset = $this->tableGateway->select(array('email' =>
  $userEmail));
  $row = $rowset->current();
  if (!$row) {
    throw new \Exception("Could not find row $ userEmail");
  }
  return $row;
}
public function deleteUser($id)
{
  $this->tableGateway->delete(array('id' => $id));
}
```

2. Создайте новый контроллер для управления пользователями в файле CommunicationApp/module/Users/src/Users/Controller/UserManagerController.php.

3. Контроллер `UserManagerController` будет содержать в себе следующие действия:

   - `indexAction()`. Это действие выводит на экран всех пользователей системы; мы также выведем ссылки для добавления/редактирования и удаления пользователей, как показано в следующем коде:

     ```
     $userTable = $this->getServiceLocator()->get('UserTable');
     $viewModel = new ViewModel(array(
     'users' => $userTable->fetchAll()));
     return $viewModel;
     ```

   - `editAction()`. Это действие служит для вывода формы `edit`, которая позволяет изменить информацию, относящуюся к пользователю:

     ```
     $userTable = $this->getServiceLocator()->get('UserTable');
     $user = $userTable->getUser($this->params()->fromRoute('id'));
     $form = $this->getServiceLocator()->get('UserEditForm');
     $form->bind($user);
     $viewModel = new ViewModel(array(
     ```

```
        'form' => $form,
        'user_id' => $this->params()->fromRoute('id')
      ));
      return $viewModel;
```

МЕТОД BIND

Метод bind, используемый в функции Form, позволяет связать модель с формой. Эта функция работает в двух направлениях: она обновляет форму в представлении данными из модели, а модель — данными, получаемыми из формы, если форма проверяется ($form->isValid()).

Дополнительную информацию можно получить в статье http://framework.zend.com/manual/2.2/en/modules/zend.form.quick-start.html#binding-an-object.

- `processAction()`. Действие `processAction` применяется при отправке пользовательской формы `edit`; `processAction` сохраняет обновленную запись и возвращает управление методу `indexAction`.

```
// Получение идентификатора пользователя из POST
$post = $this->request->getPost();
$userTable = $this->getServiceLocator()->get('UserTable');
// Загрузка сущности User
$user = $userTable->getUser($post->id);
// Привязка сущности User к Form
$form = $this->getServiceLocator()->get('UserEditForm');
$form->bind($user);
$form->setData($post);
// Сохранение пользователя
$this->getServiceLocator()->get('UserTable')->saveUser($user);
```

- `deleteAction()`. Это действие служит для удаления записи о пользователе:

```
$this->getServiceLocator()->get('UserTable')
  ->deleteUser($this->params()->fromRoute('id'));
```

4. Создайте необходимые представления и измените файл модуля config/module.config.php, указав уникальный дочерний путь для доступа к контроллеру:

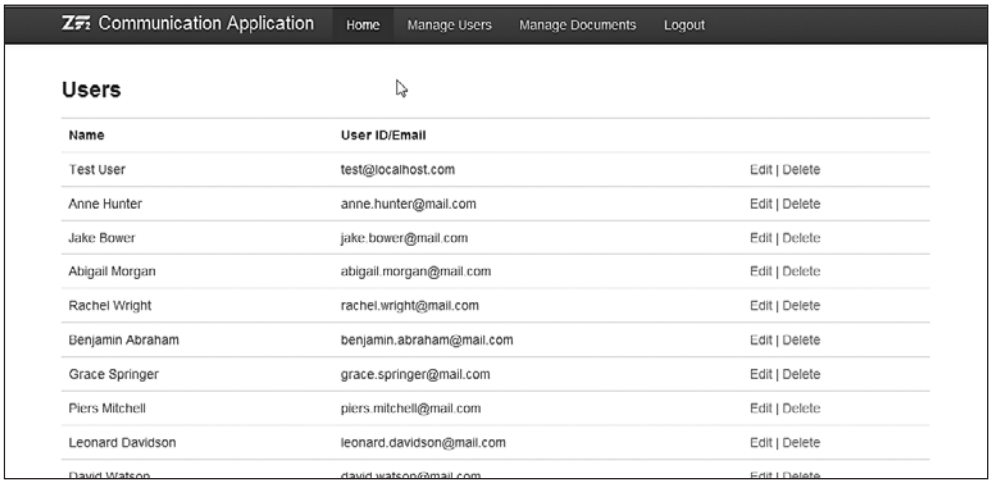
```
'user-manager' => array(
  'type' => 'Segment',
  'options' => array(
    'route' => '/user-manager[/:action[/:id]]',
    'constraints' => array(
      'action' => '[a-zA-Z][a-zA-Z0-9_-]*',
      'id' => '[a-zA-Z0-9_-]*',
    ),
    'defaults' => array(
      'controller' => 'Users\Controller\UserManager',
```

```
            'action' => 'index',
        ),
    ),
),
```

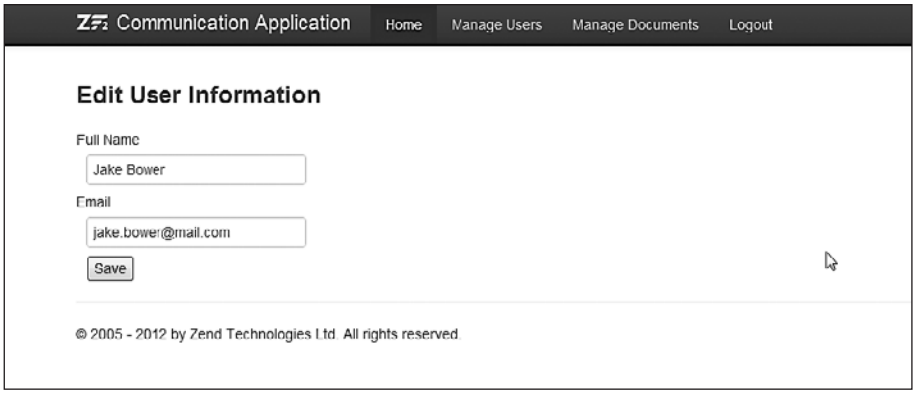
5. Наконец, добавьте новый контроллер в массив `invokables`:

```
'Users\Controller\UserManager'
  => 'Users\Controller\UserManagerController',
```

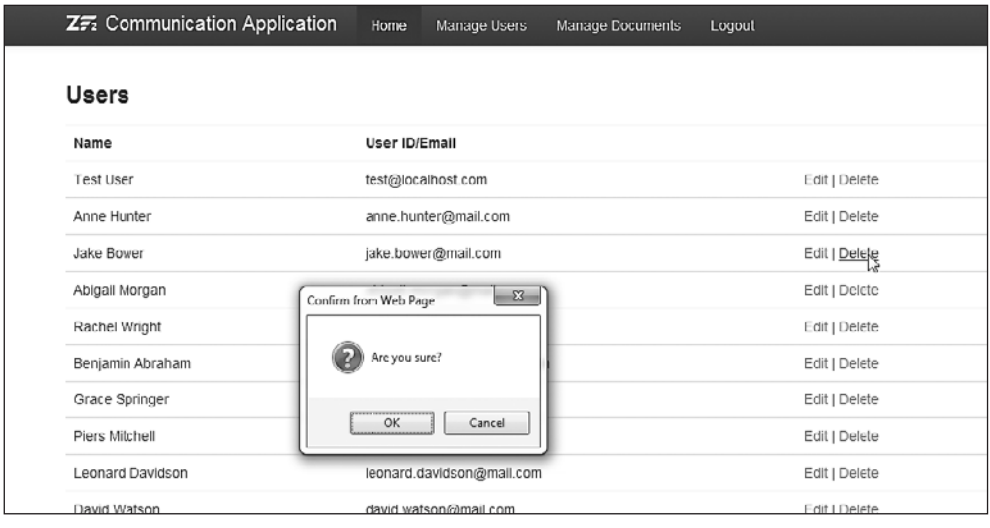
6. Теперь откройте веб-браузер и обратитесь к контроллеру, войдите в приложение и перейдите по ссылке http://comm.-app.local/users/user-manager. Вы должны увидеть страницу, схожую с представленной на рисунке.



Ссылка Edit (Редактировать) должна перенаправить вас к форме редактирования пользователя.



Ссылкой Delete (Удалить) можно воспользоваться для удаления пользователя из списка.



Что сейчас произошло?

Мы создали административный интерфейс для добавления, изменения и удаления пользователей в нашем коммуникационном приложении. Мы задействовали все ключевые функции модели `TableGateway` и реализовали функции для выполнения CRUD-операций с объектами таблицы.

Далее мы займемся более сложными вариантами применения класса `TableGateway`.

Самостоятельная работа

Перед тем как перейти к следующему разделу, попрактикуйтесь на небольшой задаче. Создайте новую форму, открывающуюся по ссылке Add User (Добавить пользователя).

Эта форма похожа на регистрационную форму, которую мы создали в предыдущей главе. Сразу после отправки формы приложение возвращает пользователя на страницу со списком пользователей системы. Ссылку на эту форму будет нужно добавить на страницу со списком пользователей.

Управление документами

В этом разделе мы создадим новый интерфейс управления документами. Этот интерфейс позволит пользователям выгружать документы, управлять выгрузками и делиться выгруженными документами с другими пользователями. Кроме того, интерфейс даст пользователям возможность управлять общим доступом к документам, а также создавать и удалять общедоступные ресурсы.

Здесь мы сконцентрируемся на обеспечении пользователей средствами выгрузки файлов и управления ими. Выгруженные файлы разместятся в файловой системе, а относительный путь выгруженного файла будет храниться в базе данных, связанной с пользователем, который осуществил выгрузку.

Далее перечислено несколько важных компонентов Zend Framework, предназначенных для выгрузки файлов.

- Элемент формы для выгрузки файла (`Zend\Form\Element\File`). Элемент выгрузки `File` используется в форме выгрузки для отображения поля ввода файла. Этот элемент эквивалентен стилевому HTML-элементу `<input type='file'../>`, который дает пользователям возможность выгружать файлы. Элемент ввода файла можно вывести на экран с помощью конструкции `'type' => 'file'` в определении формы.

- Адаптер передачи файлов (`Zend\File\Transfer\Adapter\Http`). Адаптер передачи файлов управляет выгрузкой файлов после отправки формы. Метод `setDestination()` этого адаптера позволяет пользователю задать местоположение для принимаемого файла. Метод `receive()` инициирует передачу файла.

Время действовать — создание формы выгрузки файла

В этом упражнении мы создадим новую форму выгрузки документа; выгруженные файлы будут храниться в файловой системе, а информация о выгруженном файле — в таблице `uploads` базы данных. Папка, в которой будут храниться выгруженные файлы, определена в конфигурации модуля.

1. Наше первое действие — задать в конфигурации модуля (файл config/module.config.php) местоположение, в которое можно выгружать файлы:

```
<?php
return array(
// Другие конфигурации
// ...
// ...
// КОНФИГУРАЦИИ МОДУЛЯ
'module_config' => array(
  'upload_location' => __DIR__ . '/../data/uploads',
  ),
);
```

2. Далее нам необходимо создать таблицу, в которой будет храниться информация о выгруженных файлах:

```
CREATE TABLE IF NOT EXISTS uploads (
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  filename VARCHAR( 255 ) NOT NULL,
  label VARCHAR( 255 ) NOT NULL,
  user_id INT NOT NULL,
  UNIQUE KEY (filename)
);
```

3. Создайте классы `Upload` и `UploadTable` для взаимодействия с таблицей `uploads`. Добавьте методы, вызываемые по умолчанию: `saveUpload()`, `fetchAll()`, `getUpload()` и `deleteUpload()`. Также добавьте метод `getUploadsByUserId($userId)`, считывающий выгрузки указанного пользователя:

```
public function getUploadsByUserId($userId)
{
  $userId = (int) $userId;
```

```
    $rowset = $this->tableGateway->select(
      array('user_id' => $userId));
    return $rowset;
  }
```

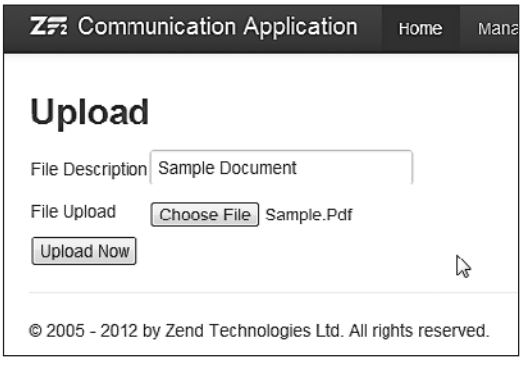
4. Создайте контроллер `UploadManagerController` для управления выгрузками файлов. Добавьте действие `indexAction()` для вывода списка выгрузок, выполненных пользователем:

```
$uploadTable = $this->getServiceLocator()->get('UploadTable');
$userTable = $this->getServiceLocator()->get('UserTable');
// Получение информации о пользователе от сеанса
$userEmail = $this->getAuthService()->getStorage()->read();
$user = $userTable->getUserByEmail($userEmail);

$viewModel = new ViewModel( array(
  'myUploads' => $uploadTable->getUploadsByUserId($user->id),
));
return $viewModel;
```

5. Создайте форму выгрузки с полем ввода файла, как описано в следующем фрагменте кода:

```
$this->add(array(
  'name' => 'fileupload',
  'attributes' => array(
    'type' => 'file',
  ),
  'options' => array(
    'label' => 'File Upload',
  ),
));
```

Форма выгрузки должна выглядеть следующим образом.



6. Создайте представления для формы выгрузки файла и действия `index`. Теперь у нас есть все элементы, необходимые для управления выгрузкой файлов. Нам нужно считать путь выгрузки файла из конфигурации и воспользоваться штатным адаптером Zend Framework для передачи файлов по протоколу HTTP, чтобы принять файл в это место. Для получения конфигурации используется метод `get('config')` локатора служб. Следующий код предназначен для считывания из конфигурации местоположения для выгрузки файла:

```
public function getFileUploadLocation()
{
  // Получение конфигурации из конфигурационных данных модуля
  $config = $this->getServiceLocator()->get('config');
  return $config['module_config']['upload_location'];
}
```

7. Последний шаг — реализовать процесс выгрузки файла. Имеются два действия, которые необходимо выполнить после успешной отправки формы.

   1) файл должен быть перемещен в место, предназначенное для выгруженных файлов.

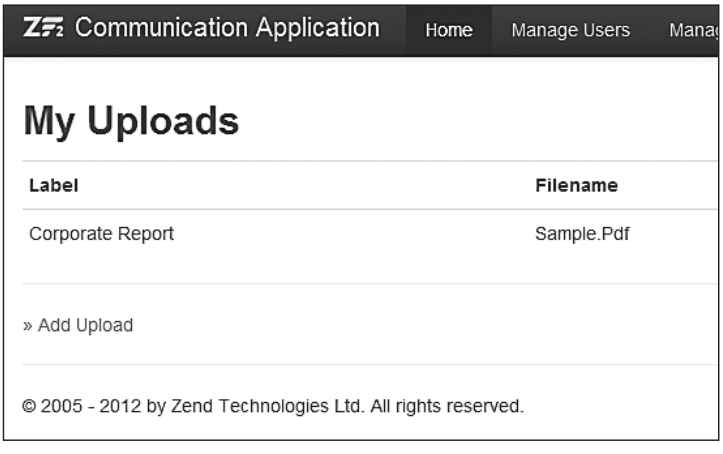
   2) в таблицу `'uploads'` нужно добавить запись, описывающую выгруженный файл, с помощью следующего кода:

```
$uploadFile = $this->params()->fromFiles('fileupload');
$form->setData($request->getPost());
if ($form->isValid()) {
  // Получение конфигурации из конфигурационных данных модуля
  $uploadPath = $this->getFileUploadLocation();

  // Сохранение выгруженного файла
  $adapter = new \Zend\File\Transfer\Adapter\Http();
  $adapter->setDestination($uploadPath);
  if ($adapter->receive($uploadFile['name'])) {
    // Успешная выгрузка файла
    $exchange_data = array();
    $exchange_data['label'] = $request->getPost()->get('label');
    $exchange_data['filename'] = $uploadFile['name'];
    $exchange_data['user_id'] = $user->id;

    $upload->exchangeArray($exchange_data);
    $uploadTable = $this->getServiceLocator()->get('UploadTable');
    $uploadTable->saveUpload($upload);

    return $this->redirect()->
      toRoute('users/upload-manager', array('action' => 'index'
    ));
  }
}
```

8. Добавьте дочерний путь (менеджер выгрузок) для контроллера `UploadManager`, а контроллер добавьте в список `invokables`.

9. Откройте веб-браузер и протестируйте форму выгрузки файлов.

   Итоговая форма должна выглядеть следующим образом.

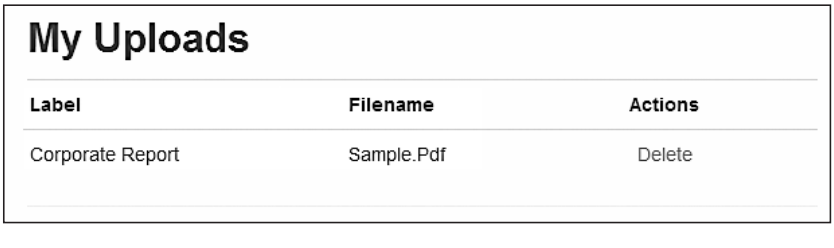


Что сейчас произошло?

Мы реализовали процесс, который позволяет пользователям выгружать файлы в приложение и просматривать выгруженные файлы. Для управления выгрузками мы воспользовались компонентами Zend Framework. В следующем разделе мы настроим механизм общего доступа к файлам так, чтобы разные пользователи могли совместно работать с документами. Но перед тем как приступить к реализации общего доступа к файлам, выполните задание.

Самостоятельная работа

Ваше следующее задание — добавить ссылку Delete (Удаление), которая позволила бы пользователям удалять выгруженные файлы, как показано на рисунке. Команда пользователя на удаление файла обязательно должна приводить к уничтожению файла в файловой системе.



Управление общим доступом к файлам

Теперь, когда у нас есть полнофункциональное звено управления документами, наша следующая задача — наделить его механизмом общего доступа пользователей к файлам. Наиболее важная часть реализации механизма общего доступа к файлам — хранение информации о совместно используемых выгрузках. Для этого мы свяжем документы с идентификаторами пользователей в таблице `upload_sharing`.

Время действовать — реализация системы общего доступа к файлам

Для того чтобы реализовать общий доступ к файлам, нам понадобится создать новую таблицу под названием `upload_sharing` и сохранить в ней всю информацию, связанную с общим доступом. Следующие шаги поясняют, как реализовать эту систему в нашем приложении.

1. Создайте новую таблицу с названием `upload_sharing`, которая будет содержать информацию о взаимоотношениях общих для пользователей выгрузок:

```
CREATE TABLE IF NOT EXISTS uploads_sharing (
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  upload_id INT NOT NULL,
  user_id INT NOT NULL,
  UNIQUE KEY (upload_id, user_id)
);
```

2. В определении модуля `Module.php` добавьте простой объект `TableGateway` для таблицы `uploads_sharing`:

```
'UploadSharingTableGateway' => function ($sm) {
  $dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');
  return new TableGateway('uploads_sharing', $dbAdapter);
},
```

3. Измените конструктор класса `UploadTable` так, чтобы он принимал дополнительный параметр объекта общедоступной выгрузки `TableGateway`:

```
public function __construct(TableGateway $tableGateway,
TableGateway $uploadSharingTableGateway)
{
  $this->tableGateway = $tableGateway;
  $this->uploadSharingTableGateway = $uploadSharingTableGateway;
}
```

4. Измените конфигурацию модуля (Module.php) для фабрики `UploadTable`, включив в нее поддержку `UploadSharingTableGateway`:

```
'UploadTable' => function($sm) {
  $tableGateway = $sm->get('UploadTableGateway');
  $uploadSharingTableGateway = $sm->get('UploadSharingTableGateway');
  $table = new UploadTable($tableGateway,
  $uploadSharingTableGateway);
  return $table;
},
```

5. Включите в класс `UploadTable` поддержку следующих функций общего доступа к файлам:

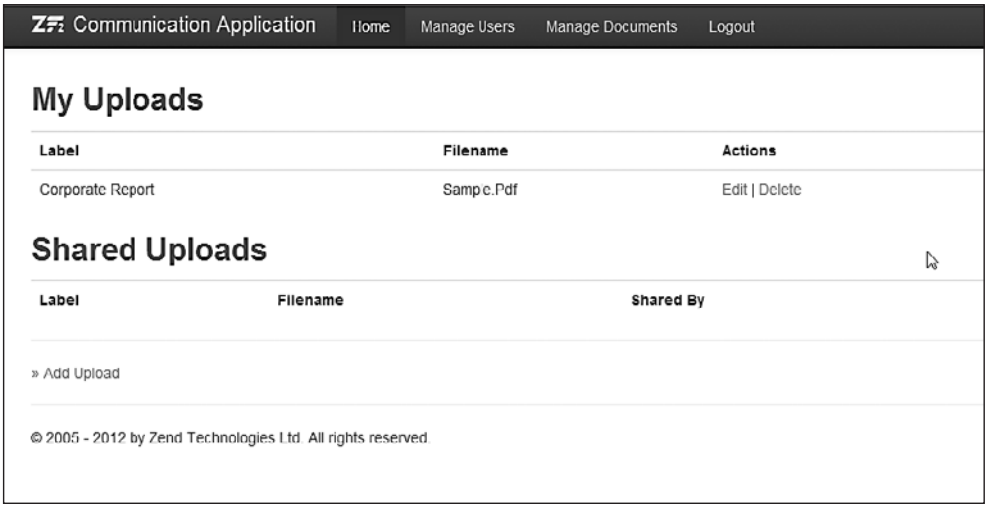
   - `addSharing()` — добавляет новое разрешение на доступ к выгруженному файлу для пользователя;
   - `removeSharing()` — удаляет разрешение на доступ к выгруженному файлу для пользователя;
   - `getSharedUsers()` — получает список пользователей, которые имеют доступ к выгруженному файлу;
   - `getSharedUploadsForUserId()` — получает список выгруженных файлов, доступных данному пользователю.

   Это можно сделать с помощью следующего кода:

```
public function addSharing($uploadId, $userId)
{
  $data = array(
    'upload_id' => (int)$uploadId,
    'user_id' => (int)$userId,
  );
  $this->uploadSharingTableGateway->insert($data);
}

public function removeSharing($uploadId, $userId)
{
  $data = array(
    'upload_id' => (int)$uploadId,
    'user_id' => (int)$userId,
  );
  $this->uploadSharingTableGateway->delete($data);
}

public function getSharedUsers($uploadId)
{
  $uploadId = (int) $uploadId;
```

```
    $rowset = $this->uploadSharingTableGateway->select(
    array('upload_id' => $uploadId));
    return $rowset;
  }

  public function getSharedUploadsForUserId($userId)
  {
    $userId = (int) $userId;
    $rowset = $this->uploadSharingTableGateway->select(
    function (Select $select) use ($userId){
      $select->columns(array())
      ->where(array('uploads_sharing.user_id'=>$userId))
      ->join('uploads', 'uploads_sharing.upload_id = uploads.id');
    });
    return $rowset;
  }
```

В разделе Manage Documents (Управление документами) перечислены все файлы, выгруженные конкретным пользователем, а также доступные ему файлы, выгруженные другими пользователями.



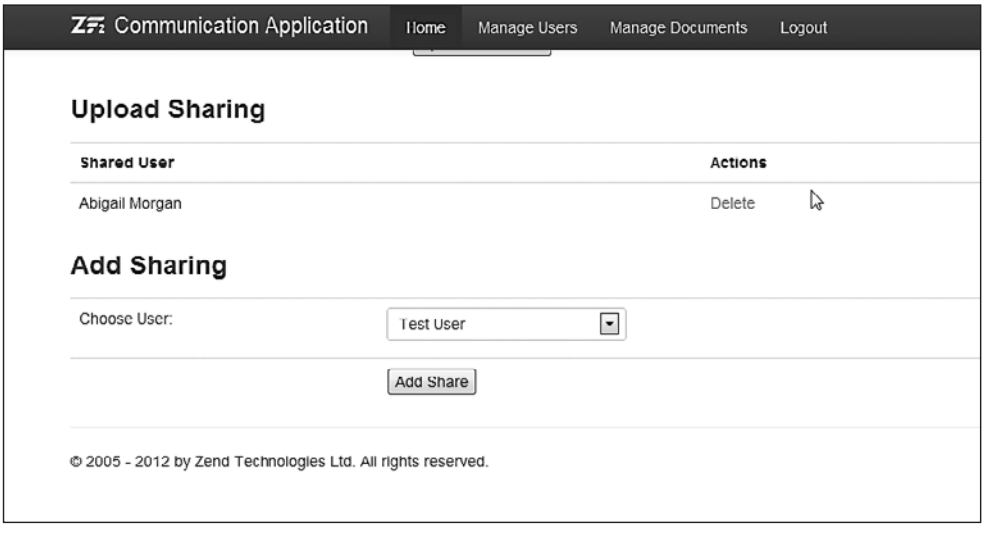
6. Измените форму редактирования выгрузок так, чтобы она отображала список пользователей, которым предоставлен общий доступ к выгруженному файлу. Для этого можно передать идентификатор выгрузки методу `getSharedUsers()` объекта `UploadTable`.

7. Добавьте в форму редактирования выгрузок новый раздел, который позволил бы предоставить общий доступ к выгрузкам; для этого можно вывести список всех пользователей системы в раскрывающемся списке. Когда пользователь

щелкает на кнопке Add Share (Добавить общий доступ), в таблицу upload_sharing добавляется новая запись.

```
$userTable = $this->getServiceLocator()->get('UserTable');
$uploadTable = $this->getServiceLocator()->get('UploadTable');
$form = $this->getServiceLocator()->get('UploadForm');
$request = $this->getRequest();
if ($request->isPost()) {
  $userId = $request->getPost()->get('user_id');
  $uploadId = $request->getPost()->get('upload_id');
  $uploadTable->addSharing($uploadId, $userId);
}
```

На следующем рисунке показана страница Upload Sharing (Общий доступ к выгруженным файлам) с раскрывающимся списком для предоставления общего доступа.



8. Последний раздел интерфейса управления общим доступом к файлам должен давать пользователям возможность загружать общедоступные файлы. Эта возможность обеспечивается в нашем приложении, управляющем совместным использованием файлов, функцией fileDownloadAction():

```
public function fileDownloadAction()
{
  $uploadId = $this->params()->fromRoute('id');
  $uploadTable = $this->getServiceLocator()->get('UploadTable');
  $upload = $uploadTable->getUpload($uploadId);
```

```
        // Считывание конфигурации из модуля
        $uploadPath = $this->getFileUploadLocation();
        $file = file_get_contents($uploadPath ."/" . $upload->filename);

        // Непосредственное возвращение ответа
        $response = $this->getEvent()->getResponse();
        $response->getHeaders()->addHeaders(array(
          'Content-Type' => 'application/octet-stream',
          'Content-Disposition' => 'attachment;filename=
            "'.$upload->filename . '"',
        ));
        $response->setContent($file);
        return $response;
    }
```

ЗАГРУЗКА ФАЙЛОВ

Чтобы реализовать функцию загрузки файлов, мы должны отключить макет. Это можно сделать, прямо указав объект HTTP-ответа как результат этого действия (см. предыдущий код). То же самое можно выполнить с помощью метода setTerminal(), как показано в следующем коде:

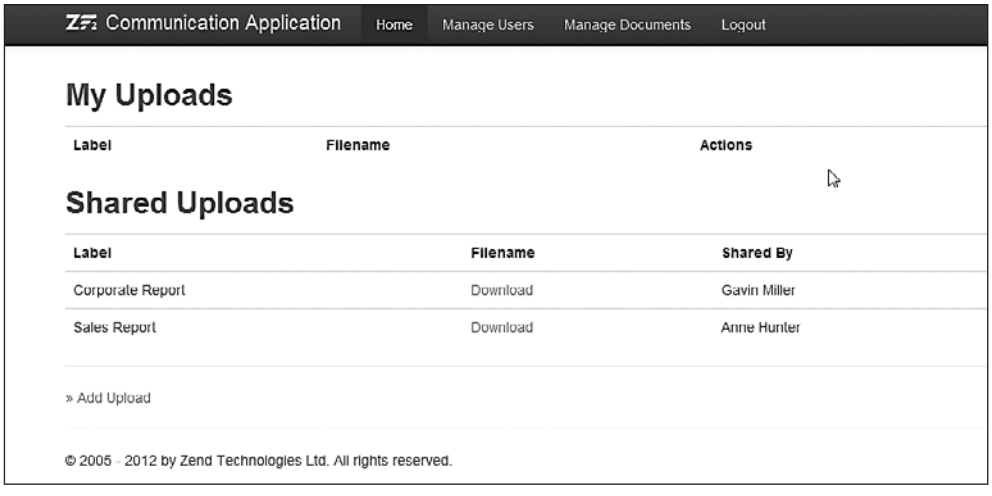
```
$result = new ViewModel();

$result->setTerminal(true);

return $result;
```

ЗАГРУЗКА БОЛЬШИХ ФАЙЛОВ

Метод file_get_contents() рассчитан на выгрузку небольших файлов и расходует большой объем памяти при обработке файлов большого размера. Для повышения производительности можно создать потоковый объект HTTP-ответа Zend\Http\Response\Stream() и выполнить загрузку файла через поток.

9. Теперь у нас есть полнофункциональная система, обеспечивающая совместный доступ пользователей к файлам. Протестируйте ее: начните с предоставления доступа к файлу различным пользователям и выполните от имени этих пользователей вход в систему и выход из системы.

Окончательная форма должна выглядеть следующим образом.



Что сейчас произошло?

Мы создали таблицу с информацией о взаимоотношениях пользователей и выгруженных файлов. Мы изменили класс `UploadTable`, добавив в него механизм поддержки дополнительных функций для совместного использования файлов. Мы создали контроллеры и представления, позволяющие открывать общий доступ к файлам, и дали пользователям возможность загружать общие файлы с помощью сценария загрузки. Благодаря этим действиям мы успешно реализовали систему совместного использования файлов, в которой пользователи могут выгружать и редактировать документы, а также делиться ими с другими пользователями.

Контрольные вопросы

1. Какая функция класса `TableGateway` служит для определения идентификатора последней вставленной записи?

   1) `getLastId();`

   2) `getLastInsertId();`

   3) `get('last_insert_id');`

   4) `getLastInsertValue()`.

2. Какой метод можно использовать для отключения макета в модели представления?

   1) `$viewModel->setNoLayouts(true);`

   2) `$ viewModel->Layouts(false);`

   3) `$viewModel->setTerminal(true);`

   4) `$viewModel->setLayouts(false)`.

Заключение

В этой главе мы рассмотрели несколько тем, касающихся управления данными и файлами. Сначала мы подробно изучили, как использовать паттерн `TableGateway` при работе с базами данных. Затем мы реализовали простую службу выгрузки файлов с помощью компонентов Zend Framework, обеспечивающих передачу файлов. Наконец, мы создали простую службу совместного использования файлов, применяя как компоненты Zend Framework для передачи файлов, так и паттерн `TableGateway`. В следующей главе мы будем работать над созданием внешних интерфейсов, в основном на базе технологий JavaScript и AJAX.

Чат и электронная почта

5

Разработка любых веб-приложений в значительной степени опирается на сценарии, выполняемые на стороне клиента, в первую очередь — на JavaScript- и CSS-сценарии. Zend Framework поддерживает паттерн модель-представление-контроллер (Model-View-Controller, MVC), который предоставляет базовые функции управления информацией, передаваемой в браузер. Классы помощников представлений в Zend Framework 2.0 обеспечивают максимальный контроль над контентом, выводимым в браузере клиента.

В этой главе мы сконцентрируемся на создании компонента с функциями простого группового чата и электронной почты, который будет использовать различные возможности Zend Framework 2.0 по поддержке внешних интерфейсов. Перечислим важные темы, рассматриваемые в этой главе:

- использование внешних JavaScript-библиотек в приложении Zend Framework 2.0;
- реализация простого приложения группового чата с помощью Zend Framework 2.0 и JavaScript;
- использование библиотеки `Zend\Mail` для отправки сообщений электронной почты;
- основы работы с менеджером событий Zend Framework.

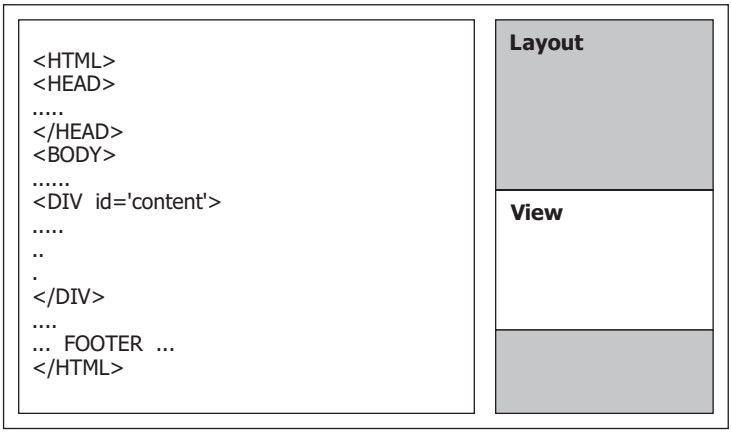
Макеты и представления

Паттерн MVC в Zend Framework использует макеты и представления для вывода страниц в веб-браузере; спецификация макета управляет контентом страницы в целом, а информация уровня представления содержится в самих представлениях. Главная идея такого подхода — свести к минимуму количество избыточного HTML-кода, который необходимо сгенерировать для каждого из представлений.

Благодаря макетам можно создавать приложения с целостными и простыми в настройке пользовательскими интерфейсами; представления позволяют гибко изме-

нять содержимое интерфейса и осуществлять максимально тонкую его настройку. Эта концепция известна как *двухэтапное представление* (two-step view).

При генерации нового представления осуществляется поиск соответствующего макета среди макетов, определенных в конфигурации `view_manager`, и представление визуализируется с этим макетом.



Представленная схема объясняет, как сочетание макета и представления приводит к формированию HTML-страницы: каким бы ни было представление, часть, соответствующая представлению, изменяется, а часть, соответствующая макету, остается неизменной.

Помощники представлений

Zend Framework 2 включает в себя широкий набор помощников представлений, которые помогают выполнять сложные операции с представлениями. Если штатных помощников недостаточно, то вы можете определить собственного нестандартного помощника, реализовав интерфейс `Zend\View\HelperInterface`.

В этом разделе мы кратко рассмотрим некоторых помощников, входящих в состав Zend Framework 2.

Помощник URL

Помощник `URL` имеет следующий синтаксис:

```
url($name, $urlParams, $routeOptions = array(),
    $reuseMatchedParams = array())
```

Этот помощник генерирует URL-адрес для заданного маршрута. Чтобы получить URL-адрес для маршрута с параметрами, эти параметры можно передать помощнику. Пример:

```
<a href="<?php $this->url('users/upload-manager', array('action'=>'edit',
  'id' => 10));">Edit</a>
```

Этот код сгенерирует тег `<a href="/users/upload-manager/edit/10">Edit</a>`, если определение маршрута выглядит так:

```
'route' => '/user-manager[/:action[/:id]]'
```

Помощник BasePath

Синтаксис этого помощника:

```
basePath()
```

Помощник `Basepath` возвращает базовый URL-адрес представления, который разработчики могут добавлять к началу собственных URL-адресов и формировать ссылки на различные ресурсы.

Помощник JSON

Синтаксис этого помощника:

```
json($jsonData = array())
```

Помощник `JSON` служит для вывода PHP-массивов и зашифрованных данных в формате JSON. Большинство AJAX-библиотек классифицирует JSON-контент по заголовкам, и этот помощник также устанавливает значение заголовка типа контента `application/json`.

Реализации заполнителей

Помощники-заполнители используются в Zend Framework для выполнения с HTML-разделами `head` некоторых стандартных операций, в том числе добавления/удаления ссылок на новые JavaScript-библиотеки, привязки к новым стилям, добавления сценариев и создания перекрестных ссылок, а также добавления/удаления контента `meta`.

Перечисленные действия осуществляются помощниками-заполнителями (placeholder helpers). Такое название обусловлено тем, что сами помощники не вносят каких-либо изменений в визуализацию контента. Например, если вы добавите в HTML-код конструкцию `<?php echo $this->headLink();?>`, то она не будет ничего делать до тех пор, пока вы не добавите что-либо в помощник `headLink` с помощью функции `appendStylesheet` или другой функции.

Помощник HeadLink

Помощник `HeadLink` изменяет тег `<link>` в HTML-разделе `head` и используется для присоединения внешних CSS-стилей и управления ими.

Вот некоторые наиболее часто применяемые функции помощника `HeadLink`:

```
appendStylesheet($href, $media, $conditionalStylesheet, $extras)
offsetSetStylesheet($index, $href, $media, $conditionalStylesheet,
$extras)
prependStylesheet($href, $media, $conditionalStylesheet, $extras)
setStylesheet($href, $media, $conditionalStylesheet, $extras)
```

СОВЕТ

Для визуализации тегов Link в HTML-макете/представлении воспользуйтесь сценарием <?php echo $this->headLink(); ?>.

Помощник HeadMeta

Помощник `HeadMeta` используется для модификации тега `<meta>` в HTML-разделе `head`; он работает с данными тега `meta`.

Вот некоторые наиболее часто применяемые функции помощника `HeadMeta`:

```
appendName($keyValue, $content, $conditionalName)
offsetSetName($index, $keyValue, $content, $conditionalName)
prependName($keyValue, $content, $conditionalName)
setName($keyValue, $content, $modifiers)
appendHttpEquiv($keyValue, $content, $conditionalHttpEquiv)
offsetSetHttpEquiv($index, $keyValue, $content, $conditionalHttpEquiv)
prependHttpEquiv($keyValue, $content, $conditionalHttpEquiv)
setHttpEquiv($keyValue, $content, $modifiers)
setCharset($charset)
```

СОВЕТ

Для визуализации тегов meta в HTML-макете/представлении воспользуйтесь сценарием <?php echo $this->headMeta(); ?>.

Помощник HeadScript

Помощник `HeadScript` используется для модификации тега `<script>` в HTML-разделе `head`; он подключает внешний JavaScript-сценарий и добавляет теги `<script>` в HTML-раздел `head`.

Вот некоторые наиболее часто применяемые функции помощника `HeadScript`:

```
appendFile($src, $type = 'text/javascript', $attrs = array())
offsetSetFile($index, $src, $type = 'text/javascript', $attrs =array())
prependFile($src, $type = 'text/javascript', $attrs = array())
setFile($src, $type = 'text/javascript', $attrs = array())
appendScript($script, $type = 'text/javascript', $attrs =array())
offsetSetScript($index, $script, $type
  = 'text/javascript', $attrs = array())
prependScript($script, $type = 'text/javascript', $attrs = array())
setScript($script, $type = 'text/javascript', $attrs = array())
```

СОВЕТ

Для визуализации тегов Script в HTML-макете/представлении воспользуйтесь сценарием <?php echo $this->headScript(); ?>.

Помощник HeadStyle

Помощник `HeadStyle` используется для модификации тега `<style>` в HTML-разделе `head`; он добавляет внутренние стили путем включения тегов `<style>` в HTML-раздел `head`.

Вот некоторые наиболее часто применяемые функции помощника `HeadStyle`:

```
appendStyle($content, $attributes = array())
offsetSetStyle($index, $content, $attributes = array())
prependStyle($content, $attributes = array())
setStyle($content, $attributes = array())
```

СОВЕТ

Для визуализации тегов Style в HTML-макете/представлении воспользуйтесь сценарием <?php echo $this->headStyle(); ?>.

Помощник HeadTitle

Помощник `HeadTitle` используется для визуализации заголовка в теге `<title>` в HTML-разделе `head`; несколько вызовов помощника `headTitle()` создают список заголовков, которые визуализируются при выводе тега в макет/представление.

Можно задать необязательный параметр `$setType`, чтобы проигнорировать существующий массив заголовков; по умолчанию он имеет значение `APPEND`, которое можно заменить на `PREPEND` или `SET` (перезаписать). Синтаксис помощника имеет вид:

```
headTitle($title, $setType = null);
```

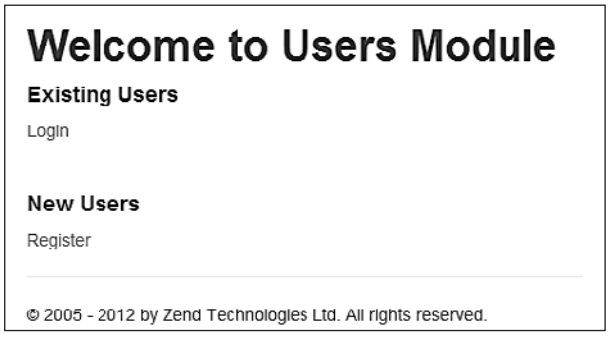
СОВЕТ

Для визуализации тегов Title в HTML-макете/представлении воспользуйтесь сценарием <?php echo $this->headTitle(); ?>.

Время действовать — использование библиотеки jQuery UI в простой странице

В этом упражнении мы изменим некоторые из созданных нами страниц так, чтобы в них использовалась библиотека jQuery UI, и обеспечим визуализацию элементов интерфейса на этих страницах средствами jQuery UI.

1. Посмотрите на текущую домашнюю страницу приложения, показанную на рисунке; наша следующая задача — преобразовать ссылки Login (Войти в систему) и Register (Зарегистрироваться) так, чтобы они визуализировались как кнопки из библиотеки jQuery UI.



2. Удалите ссылки Login и Register из представления `index` (файл module/Users/view/users/index/index.html) и добавьте класс `ui-button` к ссылкам, как показано в следующем фрагменте кода:

```
<a href="/users/login" class='ui-button'>Login</a>
<a href="/users/register" class='ui-button'>Register</a>
```

3. Добавьте внешние ссылки на jQuery UI в начало представления:

```
// Подключенные сценарии из библиотеки jQuery UI
$this->headScript()->appendFile(
  'http://code.jquery.com/jquery-1.8.3.js','text/javascript');

$this->headScript()->appendFile(
  'http://code.jquery.com/ui/1.10.0/jquery-ui.js','text/javascript');

// Подключение стилей из библиотеки jQuery UI
$this->headLink()->appendStylesheet(
  'http://code.jquery.com/ui/1.10.0/themes/base/jquery-ui.css');
```

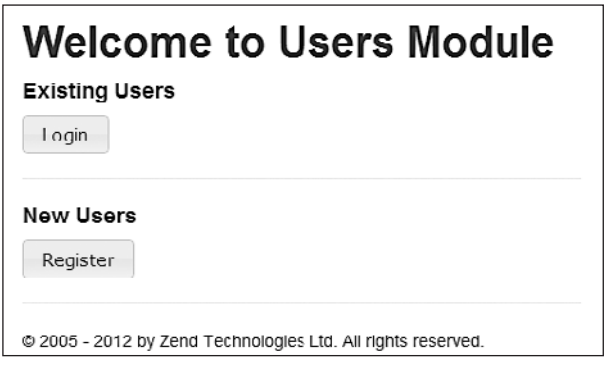
ССЫЛКИ НА СОБСТВЕННЫЕ JAVASCRIPT-БИБЛИОТЕКИ

Вместо использования прямых ссылок на внешние сценарии можно загружать эти сценарии в папку /public вашего приложения и передавать относительные ссылки в качестве параметров в функции appendFile и appendStylesheet. Можно также воспользоваться помощником basePath() для добавления базового URL-адреса в начало ссылки.

4. Добавьте сценарий инициализации пользовательского интерфейса, чтобы обе ссылки приняли вид кнопок:

```
// Инициализация кнопок
$this->headScript()->appendScript(
  '$(function() {
    $("a.ui-button").button();
  });', 'text/javascript');
```

5. Теперь посмотрите на домашнюю страницу в браузере, и вы увидите кнопки Login и Register, визуализированные в стиле jQuery UI.



Ссылка View Source (Просмотреть исходный код) на индексной странице демонстрирует применение функции `headScript()`, как показано в следующем коде:

```
<!DOCTYPE html>
<html lang="en">
```

```
    ...
    ...
    <script type="text/javascript" src="http://code.jquery.com/
      jquery-1.8.3.js"></script>
    <script type="text/javascript" src="http://code.jquery.com/
      ui/1.10.0/
jquery-ui.js"></script>
    <script type="text/javascript">
      //<!--
        $(function() {
      $("a.ui-button").button();
    });
      //-->
    </script>
    ...
    ...
    </html>
```

Что сейчас произошло?

Мы воспользовались помощниками представления Zend Framework для подключения к внешней JavaScript-библиотеке; затем мы добавили собственный JavaScript-сценарий в HTML-раздел `head` с помощью помощника `headScript()`.

Теперь наше приложение интегрировано с внешним JavaScript-сценарием; в следующем упражнении мы более подробно познакомимся с тем, как можно добавлять сценарии в HTML-раздел `head`.

Самостоятельная работа

Перед тем как приступить к разработке интерфейса для группового чата, выполним небольшое задание. Теперь, когда вы знаете, как подключать внешние JavaScript-библиотеки, вы можете загрузить библиотеку jQuery UI с ее веб-сайта, извлечь ее в папку public/ и изменить показанную чуть раньше страницу так, чтобы в ней использовалась загруженная версия jQuery UI.

Библиотеку jQuery UI можно загрузить с сайта http://jqueryui.com/.

Создание простого группового чата

Наша следующая задача — создать простое приложение для группового чата, позволяющее нескольким пользователям войти в систему и общаться друг с другом. Внутренние интерфейсы такого инструмента предельно просты: нам нужно создать таблицу, которая будет хранить все сообщения пользователей и визуализировать

их в отдельном представлении. Мы создадим форму, которая даст пользователям возможность отправлять сообщения.

Время действовать — создание простого приложения для группового чата

1. Создайте новую таблицу `chat_messages` для хранения всех пользовательских сообщений:

```
CREATE TABLE IF NOT EXISTS chat_messages (
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  message VARCHAR( 255 ) NOT NULL,
  stamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
```

2. Создайте контроллер для группового чала в файле CommunicationApp/module/Users/src/Users/Controller/GroupChatController.php.

3. Внесите необходимые изменения в файл CommunicationApp/module/Users/config/module.config.php и добавьте новый контроллер в исполняемые классы и маршруты:

```
// Исполняемый класс
'Users\Controller\GroupChat' => 'Users\Controller\
                                GroupChatController',

// Маршрут
'group-chat' => array(
  'type' => 'Segment',
  'options' => array(
    'route' => '/group-chat[/:action[/:id]]',
    'constraints' => array(
      'action' => '[a-zA-Z][a-zA-Z0-9_-]*',
      'id' => '[a-zA-Z0-9_-]*',
    ),
    'defaults' => array(
      'controller' => 'Users\Controller\GroupChat',
      'action' => 'index',
    ),
  ),
),
```

4. Создайте новое представление в файле CommunicationApp/module/Users/view/users/groupchat/index.phtml:

```
<?php
$this->headScript()->appendScript(
'$(function() {
  $( "#btnRefresh" )
    .click(function( event ) {
  document.getElementById(
    "messageListFrame").contentWindow.location.reload(true);
  })
});', 'text/javascript');

$this->headStyle()->appendStyle('
  #userName { width:100px; margin-top:10px; display: inline}
  #messageText { width:700px; margin-top:10px;}
');
?>
<h3>Group Chat</h3>
<iframe src="<?php echo $this->url('users/group-chat', array(
  'action' => 'messageList'
  )) ?>" width="80%" height="400px"
id="messageListFrame"></iframe>

<?php
// Визуализация открывающего тега
echo $this->form()->openTag($form);

// ... цикл визуализации элементов формы...
echo '<label id="userName">'. $userName .': </label>';
foreach ($form as $element) {
  echo $this->formElement($element); // <-- Magic!
  echo $this->formElementErrors($element);
}

// Визуализация закрывающего тега
echo $this->form()->closeTag();
?>
```

5. Добавьте действие `messageList` в контроллер `GroupChatController` (файл CommunicationApp/module/Users/src/Users/Controller/GroupChatController.php); это действие выполнит запрос к таблице `chat_messages`, считает из нее все записи и передаст их в представление:

```
public function messageListAction()
{
  $userTable = $this->getServiceLocator()->get('UserTable');
  $chatMessageTG = $this->getServiceLocator()->get(
    'ChatMessagesTableGateway');
  $chatMessages = $chatMessageTG->select();

  $messageList = array();
```

```
    foreach($chatMessages as $chatMessage) {
      $fromUser = $userTable->getUser($chatMessage->user_id);
      $messageData = array();
      $messageData['user'] = $fromUser->name;
      $messageData['time'] = $chatMessage->stamp;
      $messageData['data'] = $chatMessage->message;
      $messageList[] = $messageData;
    }

    $viewModel = new ViewModel(array('messageList' =>$messageList));
    $viewModel->setTemplate('users/group-chat/message-list');
    $viewModel->setTerminal(true);
    return $viewModel;
  }
```

6. Создайте простое представление для вывода сообщений в файле CommunicationApp/module/Users/view/users/group-chat/message-list.phtml, которое будет выводить сообщения из массива `$messageList`:

```
<!DOCTYPE html>
<html lang="en">
<body>
<section id="messages" >
  <?php foreach ($messageList as $mesg) : ?>
  <div class="message" style="clear:both;">
    <span class='msg-time'>
  [<?php echo $this->escapeHtml($mesg['time']);?>]
    </span>
    <span class='msg-user'>
  <?php echo $this->escapeHtml($mesg['user']);?>:
    </span>
    <span class='msg-data'>
  <?php echo $this->escapeHtml($mesg['data']);?>
    </span>
  </div>
  <?php endforeach; ?>
</section>
</body>
</html>
```

7. Создайте метод с названием `sendMessage()`, который вызывается, когда пользователь посылает сообщение, и сохраняет это сообщение в базе данных, как показано в следующем коде. Этот метод нужно поместить в контроллер группового чата в файле CommunicationApp/module/Users/src/Users/Controller/GroupChatController.php.

```
protected function sendMessage($messageTest $fromUserId)
{
```

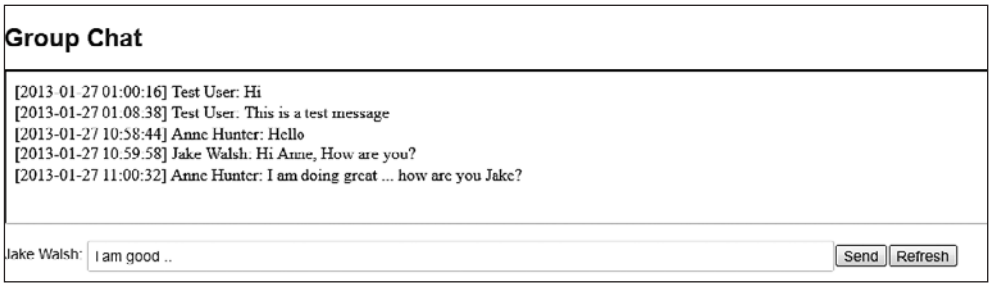
```
    $chatMessageTG = $this->getServiceLocator()
                          ->get('ChatMessagesTableGateway');
    $data = array(
      'user_id' => $fromUserId,
      'message' => $messageTest,
      'stamp' => NULL
    );
    $chatMessageTG->insert($data);
    return true;
  }
```

8. Измените функцию `indexAction` так, чтобы она выводила форму Send Message (Отправить сообщение) и вызывала функцию `sendMessage()` при отправке формы. Функцию `indexAction` нужно поместить в контроллер группового чата (файл CommunicationApp/module/Users/src/Users/Controller/GroupChatController.php).

```
public function indexAction(
{
  $user = $this->getLoggedInUser();
  $request = $this->getRequest();
  if ($request->isPost()) {
    $messageTest = $request->getPost()->get('message');
    $fromUserId = $user->id;
    $this->sendMessage($messageTest, $fromUserId);
    // Для предотвращения дублирования записей при обновлении
    return $this->redirect()->toRoute('users/group-chat');
  }

  // Подготовка формы отправки сообщения
  $form = new \Zend\Form\Form();

  $form->add(array(
    'name' => 'message',
    'attributes' => array(
      'type' => 'text',
      'id' => 'messageText',
      'required' => 'required'
    ),
    'options' => array(
    'label' => 'Message',
    ),
  ));

  $form->add(array(
    'name' => 'submit',
    'attributes' => array(
      'type' => 'submit',
```

```
        'value' => 'Send'
      ),
    ));

    $form->add(array(
      'name' => 'refresh',
      'attributes' => array(
        'type' => 'button',
        'id' => 'btnRefresh',
        'value' => 'Refresh'
      ),
    ));

    $viewModel = new ViewModel(
      array('form' => $form, 'userName' => $user->name));
    return $viewModel;
  }
```

9. Для проверки изменений войдите в браузер с двух разных компьютеров или в два различных браузера с разными учетными данными и протестируйте интерфейс группового чата.



Что сейчас произошло?

Мы успешно реализовали интерфейс группового чата с помощью Zend Framework. Этот интерфейс позволяет нескольким людям беседовать друг с другом в группе. Наша следующая задача — создать механизм отправки сообщений электронной почты другим пользователям системы; для этого мы в полной мере прибегнем к возможностям Zend Framework по работе с электронной почтой.

Самостоятельная работа

Перед тем как перейти к следующему разделу, сделайте простое упражнение. В интерфейсе группового чата мы создали кнопку Refresh (Обновить), которая перезагружает тег `iframe`. Создайте JavaScript-сценарий, перезагружающий этот тег каждые 5 секунд, и свяжите его с представлением.

Отправка почты

Zend Framework включает в себя библиотеку `Zend\Mail`, предназначенную для отправки и приема сообщений электронной почты. В этом разделе мы рассмотрим основные возможности Zend Framework по работе с электронной почтой и реализуем простой почтовый сценарий.

Библиотека `Zend\Mail` поддерживает как обычный текстовый формат, так и составные MIME-совместимые сообщения электронной почты. По умолчанию фреймворк поддерживает транспортные протоколы Sendmail, SMTP и File; новые транспортные протоколы можно реализовать с помощью интерфейса `Zend\Mail\Transport\TransportInterface`.

Объект Zend\Mail\Transport

Транспортный протокол `Mail` служит для отправки сообщений электронной почты получателям; библиотека `Zend\Mail` поддерживает следующие транспортные протоколы:

- Sendmail с помощью класса `Zend\Mail\Transport\Sendmail`;
- SMTP с помощью класса `Zend\Mail\Transport\Smtp`;
- File Transport с помощью класса `Zend\Mail\Transport\File`.

Транспортный протокол `Mail` реализует метод `send()`; этот метод принимает в качестве параметра объект типа `Zend\Mail\Message`, который содержит всю необходимую информацию для сообщения электронной почты. Отправка сообщения происходит с использованием этого протокола.

Объект Zend\Mail\Message

Объект `Zend\Mail\Message` используется для написания электронного письма в Zend Framework. Он принимает набор параметров, в том числе адрес отправителя, адрес получателя, тему и тело письма. Если сообщение имеет MIME-совместимый составной тип, то тело сообщения можно поместить в объект почтового сообщения `Zend\Mime\Message` с помощью метода `setBody()`, после чего сообщение можно отправлять. Вот некоторые наиболее часто применяемые методы объекта `Zend\Mail\Message`:

- `setFrom()`;
- `setHeaders`;

- `setTo()`;
- `addCc()` и `addBcc()`;
- `setSubject()`;
- `setBody()`.

Объекты Zend\Mime\Message и Zend\Mime\Part

Для отправки HTML-сообщений и составных сообщений каждая часть сообщения определяется как объект `Zend\Mime\Part` вместе со своим типом и связывается с объектом `Zend\Mime\Message` при помощи метода `setParts()`. Объект `Zend\Mime\Message` присваивается объекту `Zend\Mail\Message` с помощью метода `setBody()`.

Время действовать — создание простой формы электронной почты

В этом разделе мы создадим форму электронного сообщения, используя средства Zend Framework для работы с электронной почтой.

1. Создайте простую форму электронного сообщения с полями для ввода темы письма, его содержимого и адресата.
2. Настройте новый контроллер для вывода формы и напишите требуемые представления.
3. Измените контроллер так, чтобы он ссылался на пространство имен `Zend\Mail`:

```
use Zend\Mail;
```

4. Создайте в контроллере новый метод, который осуществляет отправку электронного сообщения; этот метод можно поместить в контроллер группового чата (файл CommunicationApp/module/Users/src/Users/Controller/GroupChatController.php) с помощью следующего кода:

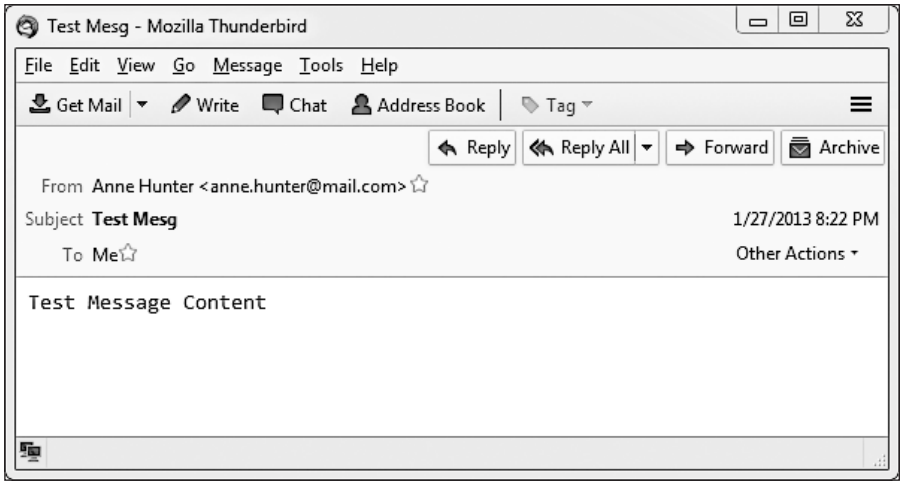
```
protected function sendOfflineMessage(
  $msgSubj, $msgText, $fromUserId, $toUserId)
{
  $userTable = $this->getServiceLocator()->get('UserTable');

  $fromUser = $userTable->getUser($fromUserId);
  $toUser = $userTable->getUser($toUserId);
```

```
        $mail = new Mail\Message();
        $mail->setFrom($fromUser->email, $fromUser->name);
        $mail->addTo($toUser->email, $toUser->name);
        $mail->setSubject($msgSubj);
        $mail->setBody($msgText);

        $transport = new Mail\Transport\Sendmail();
        $transport->send($mail);

        return true;
    }
```

ПРИМЕЧАНИЕ

В операционной системе Linux транспорт Sendmail (Zend\Mail\Transport\Sendmail) доступен по умолчанию и может использоваться для отправки сообщений электронной почты. Пользователи Windows могут воспользоваться транспортным протоколом SMTP (Zend\Mail\Transport\Smtp), чтобы подключиться к SMTP-серверу для отправки электронных сообщений. Следующая ссылка демонстрирует простой пример использования SMTP: https://packages.zendframework.com/docs/latest/manual/en/modules/zend.mail.transport.html#zend-mail-transport-quick-start-smtp-usage.

5. Просмотрите форму в веб-браузере и проверьте, доходит ли сообщение до адресата. Адресат должен получить сообщение, подобное представленному на рисунке.



Что сейчас произошло?

Мы воспользовались объектом `Zend\Mail` для передачи сообщений электронной почты внутри системы с помощью почтового транспортного протокола Sendmail. Мы также познакомились с отправкой HTML-сообщений и составных сообщений.

Самостоятельная работа

Перед тем как переходить к следующему разделу, попробуйте реализовать форму для отправки электронных сообщений в формате HTML.

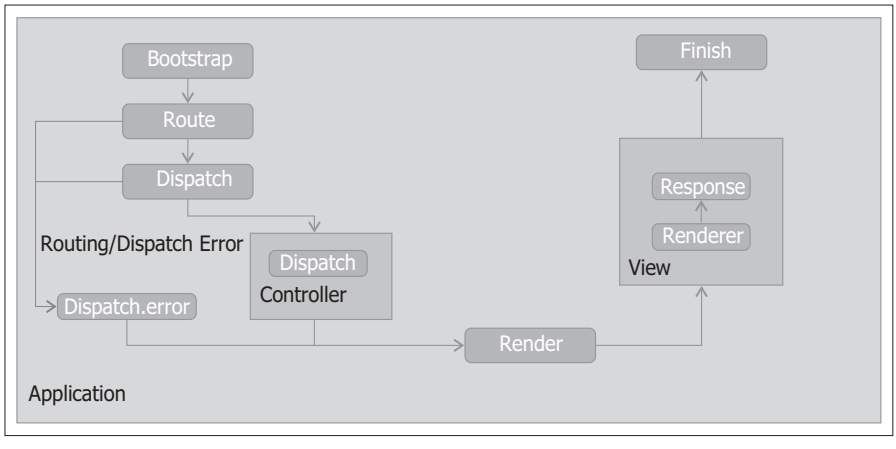
Класс Zend\EventManager

Zend Framework 2 является системой, управляемой событиями; менеджер событий позволяет подключать события почти к любой функциональности. В управлении событиями Zend Framework существуют три ключевых понятия.

- **Менеджер событий** — объект `EventManager`, который содержит набор слушателей и соответствующих им событий.
- **Слушатель** — функция обратного вызова, которая реагирует на возникновение события.
- **Событие** — действие, которое активизируется менеджером событий.

Менеджер событий предоставляет функции `attach()` и `trigger()` для создания и запуска событий соответственно. В основном мы будем пользоваться MVC-

событиями для выполнения различных действий; последовательность обработки событий в MVC-приложении представлена на следующей диаграмме.



ПРИМЕЧАНИЕ

Последовательность событий в приложении Zend Framework 2 объясняется в статье http://akrabat.com/zend-framework-2/a-list-of-zf2-events/.

При успешном выполнении приложения последовательность событий выглядит следующим образом.

1. `Zend\Mvc\Application`: `Bootstrap`.
2. `Zend\Mvc\Application`: `Route`.
3. `Zend\Mvc\Application`: `Dispatch`.
4. `Zend\Mvc\Controller\ActionController`: `Dispatch` (если контроллер расширяет этот класс).
5. `Zend\Mvc\Application`: `Render`.
6. `Zend\View\View`: `Renderer`.
7. `Zend\View\View`: `Response`.
8. `Zend\Mvc\Application`: `Finish`.

При возникновении ошибок в процессе диспетчеризации или маршрутизации поток событий будет следующим.

1. `Zend\Mvc\Application`: `Dispatch.error`.

2. `Zend\Mvc\Application`: `Render`.
3. `Zend\View\View`: `Renderer`.
4. `Zend\View\View`: `Response`.
5. `Zend\Mvc\Application`: `Finish`.

В следующем упражнении мы попробуем задать новый макет для нескольких контроллеров с помощью общего менеджера событий Zend Framework.

Время действовать — задание макета модуля с помощью менеджера событий Zend Framework

1. Создайте новый макет для страницы My Account (Моя учетная запись) и сохраните его в файле CommunicationApp/module/Users/view/layout/myaccount-layout.phtml.
2. Добавьте макет в `view_manager -> template_map` (файл CommunicationApp/module/Users/config/module.config.php):

```
'layout/myaccount' =>
                __DIR__ . '/../view/layout/myaccount-layout.phtml',
```

3. Откройте файл CommunicationApp/module/Users/module.php и добавьте ссылки на событие `MvcEvent`:

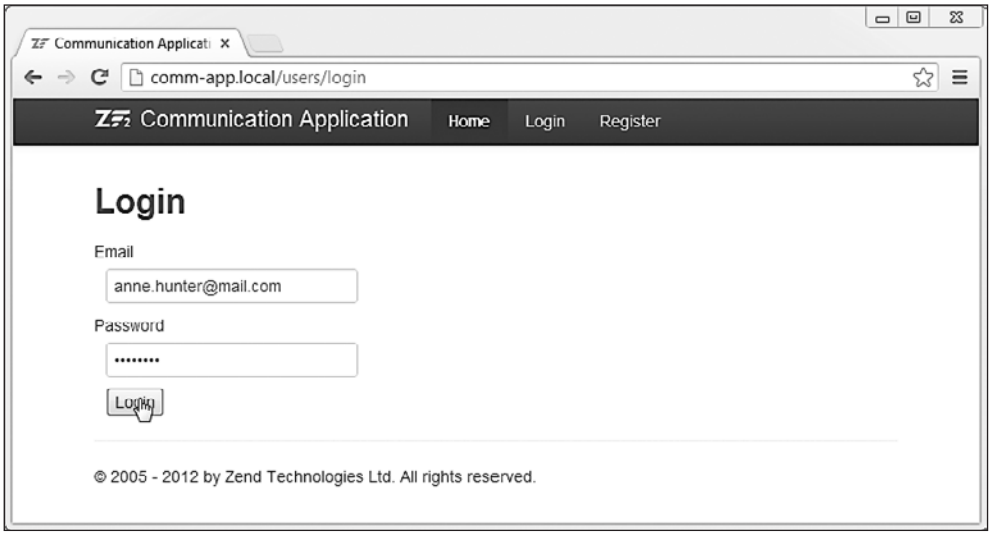
```
use Zend\Mvc\MvcEvent;
```

4. Перепишите метод `onBootStrap()`, поместив в него следующий код:

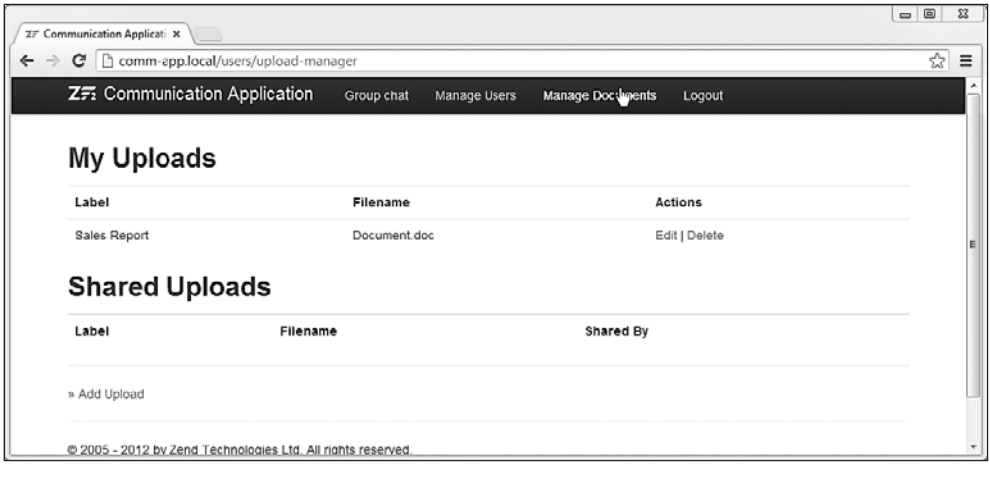
```
public function onBootstrap($e)
{
  $eventManager = $e->getApplication()->getEventManager();
  $moduleRouteListener = new ModuleRouteListener();
  $moduleRouteListener->attach($eventManager);

  $sharedEventManager = $eventManager->getSharedManager(); // общий
                                                  // менеджер событий
  $sharedEventManager->attach(
    __NAMESPACE__, MvcEvent::EVENT_DISPATCH, function($e) {
    $controller = $e->getTarget(); // обслуживаемый контроллер
    $controllerName = $controller->getEvent()
     ->getRouteMatch()->getParam('controller');
    if (!in_array($controllerName, array(
        'Users\Controller\Index', 'Users\Controller\Register',
        'Users\Controller\Login'))) {
```

```
            $controller->layout('layout/myaccount');
        }
    });
}
```

5. Откройте страницу Communication Application (Коммуникационное приложение) в любом веб-браузере. Обратите внимание на макет.



6. Войдите в приложение и понаблюдайте за применением нового макета.



Что сейчас произошло?

Мы воспользовались менеджером событий Zend Framework для подключения слушателя к событию `Dispatch` модуля. Каждый раз при диспетчеризации контроллера возникает это событие. Функция обратного вызова проверяет, действителен ли контроллер, и если он отсутствует в списке контроллеров, имеющих макет `default`, то к нему применяется макет `layout`.

Контрольные вопросы

1. Какой из перечисленных помощников можно использовать для определения/подключения CSS-стилей внутри HTML-секции `head`?

   1) `HeadLink`;

   2) `HeadScript`;

   3) `HeadCss`;

   4) `HeadStyle`.

2. Какой из перечисленных транспортных протоколов подходит для передачи сообщений электронной почты и поддерживается в Zend Framework 2?

   1) `Zend\Mail\Transport\Pop`;

   2) `Zend\Mail\Transport\Smtp`;

   3) `Zend\Mail\Transport\Imap`;

   4) `Zend\Mail\Transport\File`.

Заключение

В этой главе мы рассмотрели широкий круг тем: сначала мы познакомились с использованием внешних JavaScript-сценариев, затем создали простое приложение для группового чата, изучили библиотеку `Zend\Mail` и реализовали простую форму для отправки сообщений электронной почты. В конце главы мы познакомились с событиями и их использованием в Zend Framework. В следующей главе мы будем работать над обменом мультимедийной информацией с помощью Zend Framework, используя для этого различные специализированные прикладные программные интерфейсы.

Совместный доступ к мультимедиа

6

С появлением социальных сетей выгрузка изображений и видеороликов и управление ими в Интернете получили очень широкое распространение. Все больше и больше приложений позволяют получать и совместно использовать мультимедийную информацию с помощью внешних медиахостов и таких служб, как Google, Flickr и YouTube. В Zend Framework 1.0 интеграция с многочисленными внешними службами обеспечивалась пакетом `Zend_Service`; в Zend Framework 2 с новой модульной системой интеграция реализована иначе.

В этой главе мы используем различные внешние модули Zend Framework 2.0 для управления изображениями и видеороликами. Вот краткий список рассматриваемых тем:

- установка внешних модулей в приложение Zend Framework;
- настройка простой фотогалереи;
- изменение размера изображений и другие действия с изображениями, выполняемые с помощью модуля `WebinoImageThumb`;
- знакомство прикладным программным интерфейсом Zend GData;
- использование интерфейса GData для доступа к альбомам из служб Google Photos и YouTube.

Внешние модули

Одна из самых важных функциональных возможностей Zend Framework 2.0 — интеграция внешних модулей в PHP-приложения, и такая интеграция полностью контролируется с помощью инструмента управления зависимостями (в нашем случае — программы `Composer`).

Эта возможность позволяет разрабатывать PHP-приложения, не заботясь о поддержке внешних библиотек средствами самого приложения. Библиотеки и приложения можно отделить друг от друга и поддерживать их по отдельности.

В этой главе мы воспользуемся внешним модулем для изменения размеров изображений, а также внешними библиотеками для подключения к службам сервиса Google.

ПРОГРАММА COMPOSER

Composer — это одно из решений по управлению зависимостями, используемое в Zend Framework. Composer позволяет разработчикам объявлять зависимости приложения от необходимых ему библиотек и управляет установкой этих библиотек. Конфигурация зависимостей хранится в файле composer.json.

Изменение размера изображений

В составе Zend Framework 1.0 имелся фильтр, который позволял изменять размер изображений при их выгрузке; в Zend Framework 2.0 такого инструмента больше нет. Наша следующая задача — найти несложный модуль для изменения размера изображений и установить его в наше приложение. Давайте начнем.

Время действовать — изменение размера изображений с помощью модулей

1. Посетите сайт модулей Zend Framework 2: http://modules.zendframework.com/.
2. Выполните поиск по ключевому слову `WebinoImageThumb`.
3. Чтобы установить этот модуль, вам понадобится обновить файл composer.json в корне приложения и подключить модуль как обязательный.
4. Чтобы сделать это, отредактируйте файл CommunicationApp/composer.json, изменив раздел обязательных модулей:

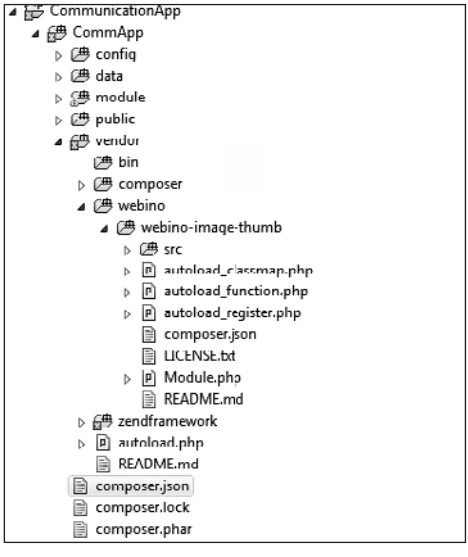
```
"require": {
  "php": ">=5.3.3",
  "zendframework/zendframework": "2.0.*",
  "webino/webino-image-thumb": "1.*",
}
```

5. Теперь выполните команду `composer.phar update`, чтобы установить добавленную зависимость:

```
$ php composer.phar update
Loading composer repositories with package information
Updating dependencies
- Installing webino/webino-image-thumb (1.0.0)
```

```
Downloading: 100%
Writing lock file
Generating autoload files
```

6. Вы увидите установленные модули в папке vendor, как показано на рисунке.



7. Теперь, когда модуль загружен, нам необходимо активировать его в файле CommunicationApp/config/application.config.php, добавив элемент `'WebinoImageThumb'` в массив `modules`:

```
return array(
  'modules' => array(
    'Application',
    'WebinoImageThumb',
    'Users',
),
```

Что сейчас произошло?

Мы установили в наше приложение внешний модуль с помощью инструмента управления зависимостями `Composer`. Мы также активировали этот модуль, чтобы сделать его доступным в любом месте нашего приложения.

Самостоятельная работа

Теперь, когда вы знаете, как устанавливать новые модули в приложение Zend Framework 2, выполните простое задание: установите в наше приложение пакет

ZendGData. Инструкции по его установке можно найти по адресу https://packages.zendframework.com/. Мы будем пользоваться этим модулем в следующих разделах этой главы.

Приложение для работы с фотогалереей

Мы начнем с того, что создадим собственную фотогалерею с помощью Zend Framework 2. Поскольку мы уже реализовали интерфейс управления файлами, воспользуемся схожим интерфейсом и для создания фотогалереи.

Схема фотогалереи будет похожа на сущность `Upload`; мы дополнительно включим в нее поле thumbnail для хранения имени файла эскиза, который генерируется при выгрузке файла. Изображения и их эскизы будут храниться в папке <Модуль>\data\images. Для вывода изображений в браузере мы будем использовать собственное действие.

Перед тем как приступить к работе, давайте кратко ознакомимся с несколькими важными методами, поддерживаемыми модулем `WebinoImageThumb`:

```
resize($maxWidth = 0, $maxHeight = 0)
```

Изменяет размер изображения в соответствии с заданной высотой и шириной; если значение какого-либо из параметров равно 0, то соответствующее измерение не рассматривается как ограничитель.

```
adaptiveResize($width, $height)
```

Пытается с максимальной точностью преобразовать изображение в соответствии с указанными размерами, а затем обрезает остатки (от центра), чтобы размеры изображения соответствовали заданным.

```
crop($startX, $startY, $cropWidth, $cropHeight)
```

Обрезает изображения, отсчитывая заданные размеры от указанных начальных координат.

```
rotateImage($direction = 'CW')
```

Поворачивает изображения на 90 градусов по часовой стрелке или против часовой стрелки.

```
rotateImageNDegrees($degrees)
```

Поворачивает изображение на угол, заданный в градусах.

```
save($fileName, $format = null)
```

Сохраняет изображение под заданным именем.

Время действовать — реализация простой фотогалереи

1. Создайте новую сущность с названием `ImageUpload` со следующей структурой таблицы:

```
CREATE TABLE IF NOT EXISTS image_uploads (
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  filename VARCHAR( 255 ) NOT NULL,
  thumbnail VARCHAR( 255 ) NOT NULL,
  label VARCHAR( 255 ) NOT NULL,
  user_id INT NOT NULL,
  UNIQUE KEY (filename)
);
```

2. Создайте соответствующую сущность `ImageUpload` в файле src/Users/Model/ImageUpload.php, объект `TableGateway` — в файле src/Users/Model/ImageUploadTable.php и `Controller` (`MediaManagerController`) модуля `CommunicationApp/module/Users` — в файле src/Users/Controller/MediaManagerController.php.

3. В процессе `Submit` формы `Upload` сгенерируйте эскиз с помощью нового метода под названием `generateThumbnail()`; этот метод принимает в качестве параметра имя файла существующего изображения. Метод `resize` изменяет размер изображения, доводя его до 75×75 пикселов, и сохраняет в каталоге выгрузки изображений с префиксом tn_.

   Этот метод необходимо поместить в файл контроллера `MediaManagerController` (src/Users/Controller/MediaManagerController.php).

```
public function generateThumbnail($imageFileName)
{
  $path = $this->getFileUploadLocation();
  $sourceImageFileName = $path . '/' . $imageFileName;
  $thumbnailFileName = 'tn_' . $imageFileName;

  $imageThumb = $this->getServiceLocator()->get('WebinoImageThumb');
  $thumb = $imageThumb->create($sourceImageFileName,
           $options = array());
  $thumb->resize(75, 75);
  $thumb->save($path . '/' . $thumbnailFileName);

  return $thumbnailFileName;
}
```

4. Наш следующий шаг — написать действие, которое выводит изображение в режимах `Full` (в натуральную величину) и `Thumbnail` (эскиз); для этого нам нужно создать собственный маршрут, который будет принимать параметры `action`, `id` и `subaction`. Поместим следующее определение маршрута в конфигурационный файл модуля (CommunicationApp/module/Users/config/module.config.php):

```
'media' => array(
  'type' => 'Segment',
  'options' => array(
    'route' => '/media[/:action[/:id[/:subaction]]]',
    'constraints' => array(
      'action' => '[a-zA-Z][a-zA-Z0-9_-]*',
      'id' => '[a-zA-Z0-9_-]*',
      'subaction' => '[a-zA-Z][a-zA-Z0-9_-]*',

    ),
    'defaults' => array(
      'controller' => 'Users\Controller\MediaManager',
      'action' => 'index',
    ),
  ),
),
```

5. Следующий шаг — написать действие, которое будет реагировать на различные запросы, относящиеся к изображениям. Это действие нужно поместить в файл контроллера `MediaManagerController` (src/Users/Controller/MediaManagerController.php):

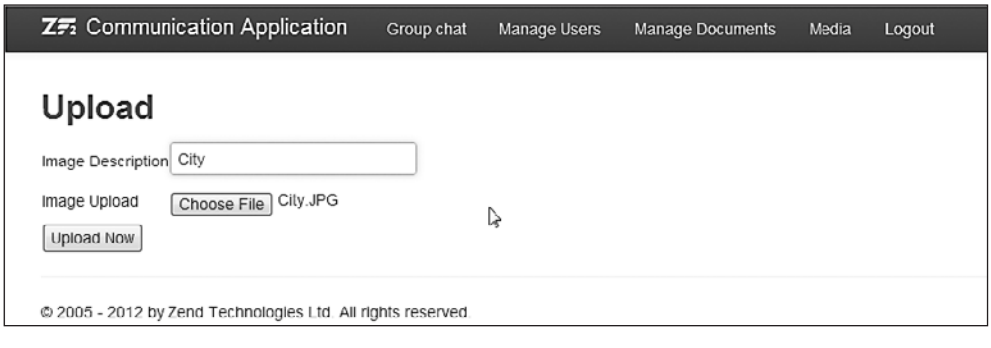
```
public function showImageAction()
{
  $uploadId = $this->params()->fromRoute('id');
  $uploadTable = $this->getServiceLocator()->get('ImageUploadTable');
  $upload = $uploadTable->getUpload($uploadId);

  // Выборка конфигурации из модуля
  $uploadPath = $this->getFileUploadLocation();
  if ($this->params()->fromRoute('subaction') == 'thumb')
  {
    $filename = $uploadPath ."/" . $upload->thumbnail;
  } else {
    $filename = $uploadPath ."/" . $upload->filename;
  }
  $file = file_get_contents($filename);

  // Прямой возврат ответа
  $response = $this->getEvent()->getResponse();
  $response->getHeaders()->addHeaders(array(
    'Content-Type' => 'application/octet-stream',
```

```
        'Content-Disposition' => 'attachment;filename="'
        .$upload->filename . '"',

      ));
      $response->setContent($file);
      return $response;
    }
```

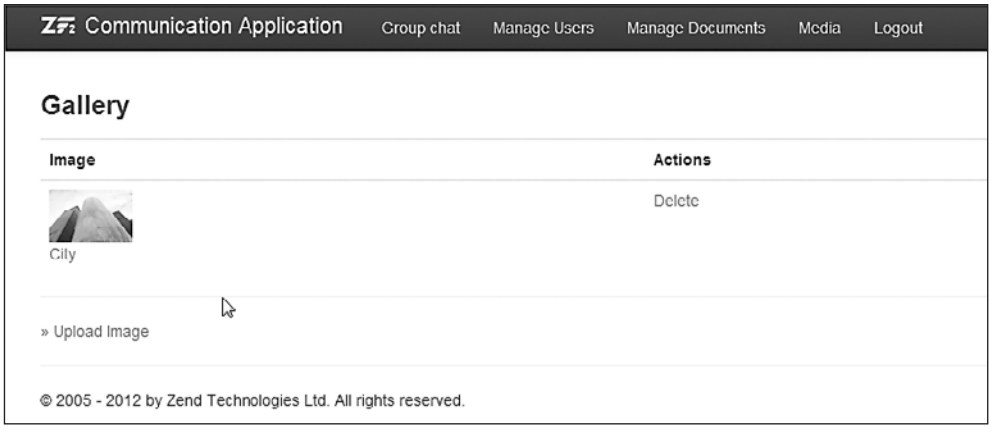
6. Проверьте, что процесс работает от начала до конца — с момента выгрузки изображения в галерею до его вывода на странице с фотографиями. Использование действия `showImageAction()` в представлении `upload` в медиаменеджере демонстрирует следующий код (файл CommunicationApp/module/Users/view/users/media-manager/view.phtml):

```
<section class="upload">
<h2><?php echo $this->escapeHtml($upload->label);?></h2>
<h4><?php echo $this->escapeHtml($upload->filename);?></h4>
<img src="<?php echo $this->url('users/media',
  array('action'=>'showImage',
    'id' => $upload->id,
    'subaction' => 'full'));?>" />
</section>
<a href="<?php echo $this->url('users/media');?>">
&raquo; Show Gallery</a>
```

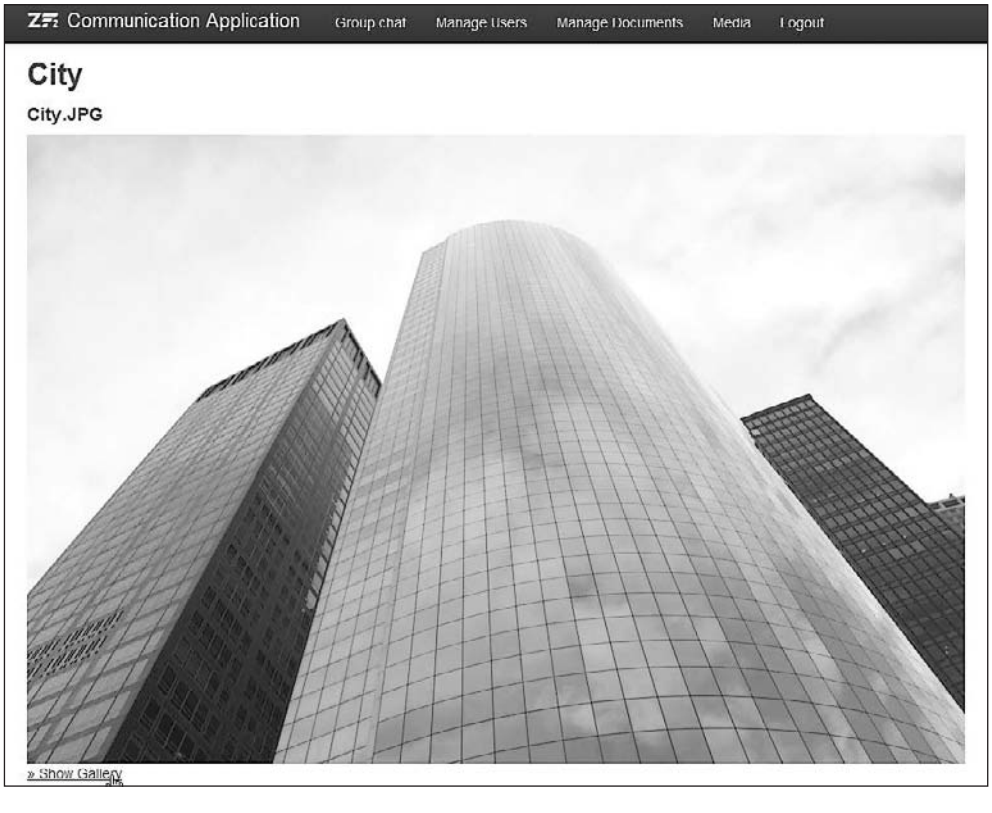
7. Теперь протестируйте приложение в любом браузере по вашему желанию. Страница выгрузки изображения должна выглядеть так, как показано на рисунке.



После успешной отправки формы выгрузки размер изображения изменится, и уменьшенное изображение появится в галерее, как показано на следующем рисунке.



Ссылка View Image (Просмотреть изображение) в верхней части уменьшенного изображения ведет на страницу с полноразмерным изображением.



Что сейчас произошло?

Мы реализовали простую фотогалерею, воспользовавшись внешней библиотекой обработки изображений. Мы создали эскизы изображений с помощью функции `resize` и собственное действие, осуществляющее вывод изображения в веб-браузере.

Самостоятельная работа

Теперь, когда вы понимаете, как работать с модулем `WebinoImageThumb`, ваше следующее задание будет заключаться в расширении возможностей фотогалереи с помощью функции вращения `rotate`. Добавьте функцию `rotate` на страницу View Image и дайте пользователю возможность вращать изображение по часовой стрелке и против часовой стрелки.

Google Data API

Прикладные программные интерфейсы (API) Google Data API предназначены для работы с различными службами Google, позволяя приложениям считывать и записывать данные. Для передачи данных интерфейсы Google Data используют протокол, схожий с протоколом публикации Atom (Atom publishing protocol). Все службы реализованы в пакете под названием `ZendGdata`.

Перечислим некоторые наиболее популярные службы Google, которые поддерживаются API из пакета `ZendGdata`:

- Picasa Web Albums;
- YouTube;
- Google Calendar;
- Google Spreadsheets;
- Google Documents;
- Google Provisioning;
- Google Analytics;
- Google Blogger;

- Google CodeSearch;
- Google Notebook.

Поскольку пакет `ZendGdata` по умолчанию не входит в Zend Framework, его необходимо установить вручную. Это можно сделать с помощью программы Composer или загрузив пакет `"zendframework/zendgdata": "2.*"`.

Google Photos API

Прикладной программный интерфейс Google Photos API позволяет выполнять выборку и редактирование фотографий, а также управлять фотографиями и альбомами в учетных записях Picasa и Google+. Интерфейс Google Data API предлагает все доступные виды услуг; перечислим некоторые ключевые функции:

- `getUserFeed()` — получение всех альбомов, связанных с пользователем;
- `insertAlbumEntry()` — создание нового альбома;
- `getAlbumFeed()` — получение указанного альбома;
- `insertPhotoEntry()` — создание нового фото;
- `getPhotoFeed()` — получение указанного фото;
- `insertCommentEntry()` — создание нового комментария;
- `getCommentEntry()` — получение указанного комментария;
- `insertTagEntry()` — создание нового тега;
- `getTagEntry()` — получение указанного тега;
- `deleteAlbumEntry()` — удаление альбома;
- `deletePhotoEntry()` — удаление фото;
- `deleteCommentEntry()` — удаление комментария;
- `deleteTagEntry()` — удаление тега.

В следующем примере мы выполним выборку существующих альбомов пользователя и фотографий, хранящихся в этих альбомах.

ВНИМАНИЕ

Чтобы двигаться дальше, нужно установить библиотеку ZendGdata в ваше приложение с помощью программы Composer. При установке, во-первых, добавьте в раздел requires файла CommunicationApp/composer.json строку «zendframework/zendgdata»: «2.*». Во-вторых, с помощью программы Composer обновите зависимости приложения командой $ php composer.phar update.

Перед тем как начать действовать, удостоверьтесь, что в вашей учетной записи в Google Photos есть несколько загруженных фотографий.

Время действовать — выборка фотографий из Google Photos

Чтобы получить доступ к фотографиям из учетной записи Google Photos, выполните следующие шаги.

1. Создайте в своем контроллере метод `getGooglePhotos()`, который будет подключаться к службе Google Photos и получать из нее все альбомы. Этот метод нужно поместить в контроллер `MediaManagerController` (файл src/Users/Controller/MediaManagerController.php).

2. Настройте API-клиента на использование запроса `Curl` с параметром, отключающим `sslverifypeer`:

```
$adapter = new \Zend\Http\Client\Adapter\Curl();
$adapter->setOptions(array(
  'curloptions' => array(
    CURLOPT_SSL_VERIFYPEER => false,
  )
));

$httpClient = new \ZendGData\HttpClient();
$httpClient->setAdapter($adapter);

$client = \ZendGData\ClientLogin::getHttpClient(
  self::GOOGLE_USER_ID,
  self::GOOGLE_PASSWORD,
  \ZendGData\Photos::AUTH_SERVICE_NAME,
  $httpClient);
```

3. Теперь создайте нового клиента Google Photos с помощью API-клиента:

```
$gp = new \ZendGData\Photos($client);
```

4. Получите список альбомов с помощью функции `getUserFeed()` и список изображений в альбоме с помощью функции `getAlbumFeed()`:

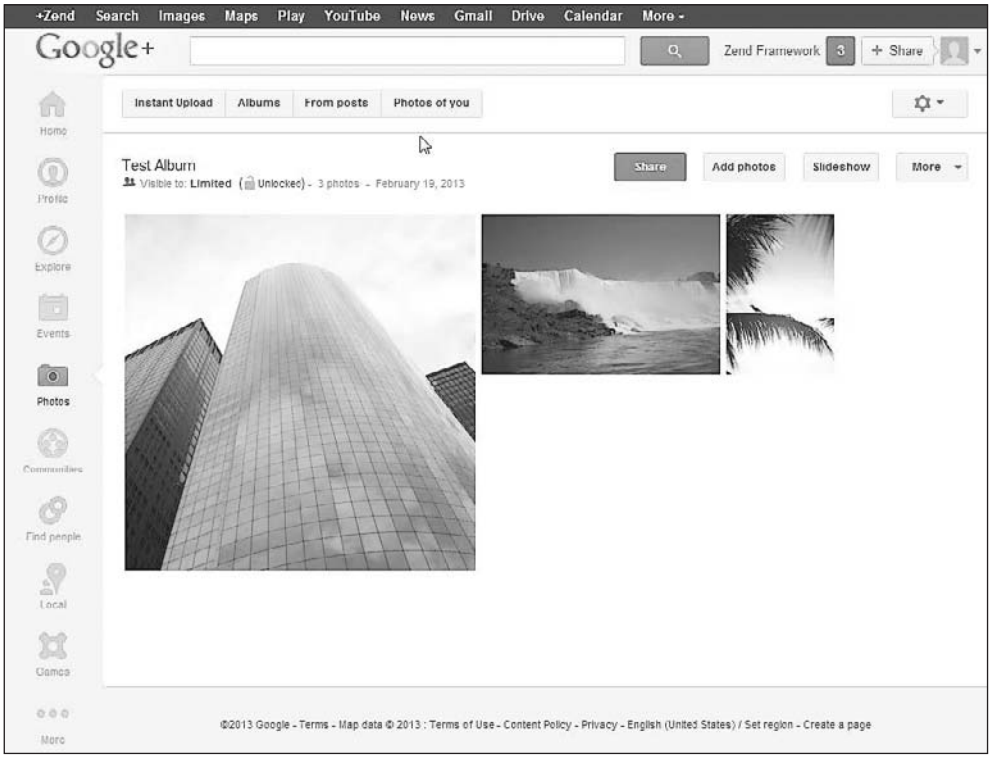
```
$userFeed = $gp->getUserFeed( self::GOOGLE_USER_ID );

foreach ($userFeed as $userEntry) {
  $albumId = $userEntry->getGphotoId()->getText();
  $gAlbums[$albumId]['label'] = $userEntry->getTitle()->getText();

  $query = $gp->newAlbumQuery();
  $query->setUser( self::GOOGLE_USER_ID );
  $query->setAlbumId( $albumId );

  $albumFeed = $gp->getAlbumFeed($query);

  foreach ($albumFeed as $photoEntry) {
    $photoId = $photoEntry->getGphotoId()->getText();
    if ($photoEntry->getMediaGroup()->getContent() != null) {
      $mediaContentArray = $photoEntry->getMediaGroup()->getContent();
      $photoUrl = $mediaContentArray[0]->getUrl();
    }

    if ($photoEntry->getMediaGroup()->getThumbnail() != null)
    {
      $mediaThumbnailArray = $photoEntry->getMediaGroup()
        ->getThumbnail();
      $thumbUrl = $mediaThumbnailArray[0]->getUrl();
    }

    $albumPhoto = array();
    $albumPhoto['id'] = $photoId;
    $albumPhoto['photoUrl'] = $photoUrl;
    $albumPhoto['thumbUrl'] = $thumbUrl;

    $gAlbums[$albumId]['photos'][] =$albumPhoto;
  }
}
// Возвращение объединенного массива в представление для визуализации
return $gAlbums;
```

5. Следующий блок кода в представлении `album` используется для визуализации альбомов; его можно поместить в представление `index` медиаменеджера CommunicationApp/module/Users/view/users/media-manager/index.phtml:

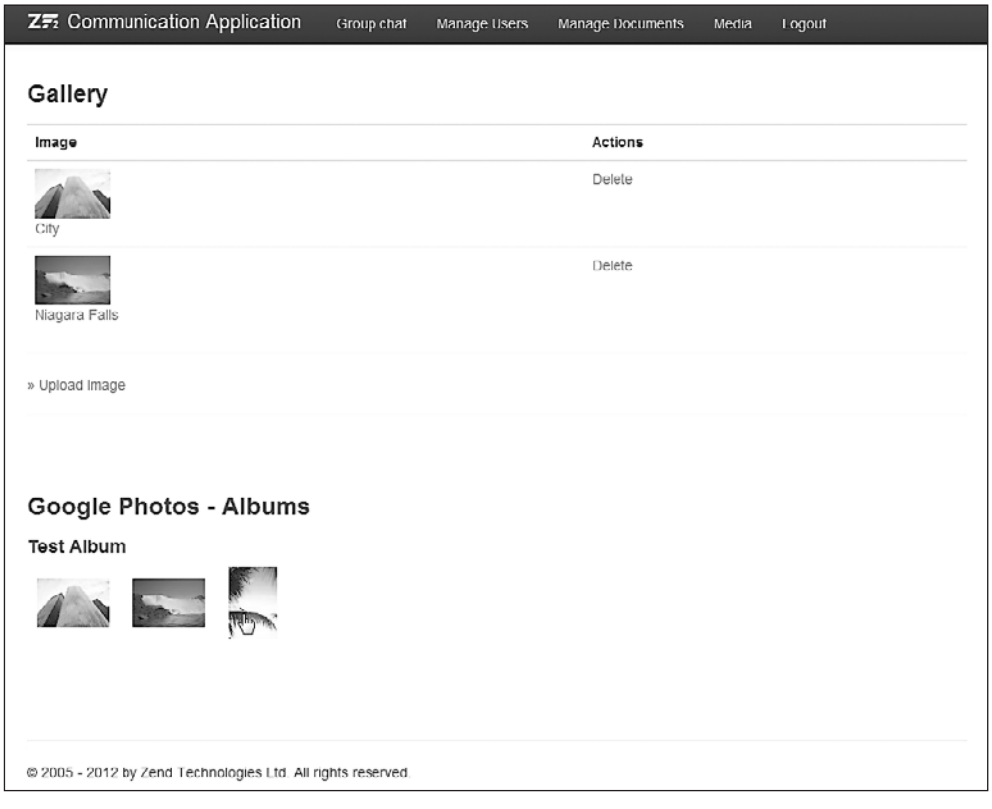
```
<?php foreach ($googleAlbums as $googleAlbum) : ?>
<h4> <?php echo $this->escapeHtml($googleAlbum['label']);?>
</h4>
```

```
<?php foreach ($googleAlbum['photos'] as $googleAlbumPhoto) : ?>
  <div class = "googleAlbumPhoto" style="padding:10px;
               display:inline">
    <a href="<?php echo $this->escapeHtml(
      $googleAlbumPhoto['photoUrl']);?>">
    <img src="<?php echo $this->escapeHtml(
      $googleAlbumPhoto['thumbUrl']);?>" />
    </a>
  </div>
  <?php endforeach; ?>
<?php endforeach; ?>
<hr />
```

6. Выгрузите картинки в альбом Google Photos.



7. Откройте страницу в окне веб-браузера; вы должны увидеть все доступные альбомы и фотографии в выбранном альбоме.



Что сейчас произошло?

Мы успешно воспользовались прикладными программными интерфейсами Google Data для получения от Google информации о файлах, выгруженных в Picasa, и с помощью этой информации визуализировали галереи в нашем приложении.

Самостоятельная работа

Ваше следующее задание будет состоять в том, чтобы при просмотре фотографии в галерее реализовать выгрузку фотографий с помощью Google Data API; у вас будет кнопка, позволяющая выгрузить фотографию в Google Photos.

YouTube Data API

Прикладной программный интерфейс YouTube Data API обеспечивает доступ к контенту YouTube; с его помощью можно выбирать видеоролики, списки воспроизведения, каналы, отправлять комментарии, выгружать видеоролики и управлять ими. Пользователям разрешено без аутентификации получать доступ к популярным видеороликам, отправлять комментарии и т. д.

Вот несколько наиболее применяемых методов YouTube API:

- `getVideoFeed()` — получение видеороликов из запроса видео;
- `getTopRatedVideoFeed()` — получение видеороликов с максимальным рейтингом из определенного запроса видео;
- `getUserUploads()` — получение видеороликов, выгруженных пользователем;
- `getUserFavorites()` — получение избранных видеороликов пользователя;
- `getVideoResponseFeed()` — получение откликов на видеоролик;
- `getVideoCommentFeed()` — получение комментариев на видеоролик;
- `getPlaylistListFeed()` — получение списка воспроизведения пользователя;
- `getSubscriptionFeed()` — получение подписки пользователя;
- `insertEntry()` — загрузка видеоролика на YouTube.

В следующем примере мы получим видеоролики, соответствующие заданному ключевому слову, а затем выведем их на веб-странице.

Время действовать — перечисление видеороликов на YouTube по ключевому слову

Чтобы вывести список видеороликов, которые размещены на YouTube и соответствуют ключевому слову, выполните следующие шаги.

1. Создайте функцию, которая будет получать видеоролики с YouTube по ключевому слову `Zend Framework`.
2. Установите соединение аналогично тому, как вы делали это для Google Photos. Это нужно сделать в новом методе `getYoutubeVideos()` контроллера `MediaManagerController` (файл src/Users/Controller/MediaManagerController.php):

```
$adapter = new \Zend\Http\Client\Adapter\Curl();
```

```
$adapter->setOptions(array(
  'curloptions' => array(
    CURLOPT_SSL_VERIFYPEER => false,
  )
));

$httpClient = new \ZendGData\HttpClient();
$httpClient->setAdapter($adapter);

$client = \ZendGData\ClientLogin::getHttpClient(
  self::GOOGLE_USER_ID,
  self::GOOGLE_PASSWORD,
  \ZendGData\YouTube::AUTH_SERVICE_NAME,
  $httpClient);
```

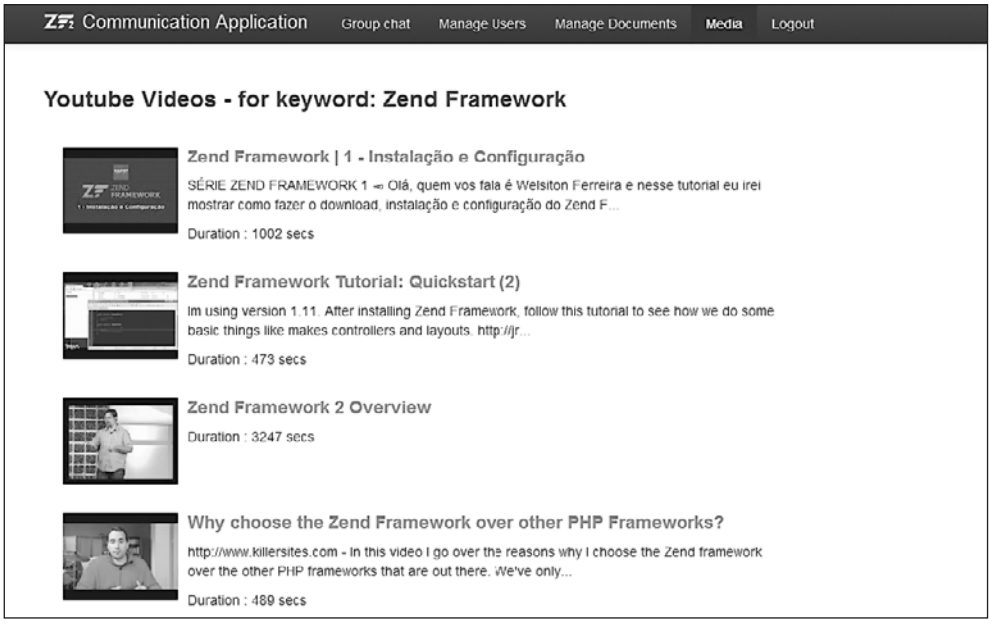
3. Инициализируйте клиента YouTube и выполните запрос видеоролика по ключевому слову `Zend Framework`:

```
$yt = new \ZendGData\YouTube($client);
$yt->setMajorProtocolVersion(2);
$query = $yt->newVideoQuery();
$query->setOrderBy('relevance');
$query->setSafeSearch('none');
$query->setVideoQuery('Zend Framework');
```

4. Выполните синтаксический разбор результатов и сохраните их в массиве:

```
$videoFeed = $yt->getVideoFeed($query->getQueryUrl(2));

$yVideos = array();
foreach ($videoFeed as $videoEntry) {
  $yVideo = array();
  $yVideo['videoTitle'] = $videoEntry->getVideoTitle();
  $yVideo['videoDescription'] =
  $videoEntry->getVideoDescription();
  $yVideo['watchPage'] = $videoEntry->getVideoWatchPageUrl();
  $yVideo['duration'] = $videoEntry->getVideoDuration();
  $videoThumbnails = $videoEntry->getVideoThumbnails();

  $yVideo['thumbnailUrl'] = $videoThumbnails[0]['url'];
  $yVideos[] = $yVideo;
}
return $yVideos;
```

5. Полученный в результате контент визуализируется в представлении, и список видеороликов выглядит так, как показано на рисунке.



Что сейчас произошло?

Мы воспользовались API из пакета `ZendGData` для получения с YouTube простого списка видеороликов, соответствующих заданному ключевому слову.

Контрольные вопросы

1. Какая команда используется в Composer для установки новой сконфигурированной зависимости?

   1) `php composer.phar setup`;

   2) `php composer.phar self-update`;

   3) `php composer.phar show`;

   4) `php composer.phar update`.

2. Какой из перечисленных методов подходит для выгрузки нового фото в службу Google Photos?

   1) `uploadPhoto()`;

   2) `insertPhoto()`;

3) `uploadNewPhoto()`;

4) `insertPhotoEntry()`.

Заключение

В этой главе мы рассмотрели различные приемы управления мультимедийной информацией. Мы начали с реализации собственной фотогалереи, а затем перешли к использованию прикладных программных интерфейсов Google GData для получения из Интернета и сохранения в Интернете этой информации.

В следующей главе мы будем работать над реализацией простого поискового интерфейса.

Поиск с помощью библиотеки Lucene

7

Большинству веб-приложений требуется поддержка встроенных поисковых механизмов. Иногда поиск может быть нацелен на простое поле в MySQL-таблице, иногда — на документ или обычный текстовый файл; есть множество способов реализовать поиск с помощью различных поисковых библиотек. Одна из таких библиотек под названием Lucene обеспечивает прекрасные поисковые возможности, позволяющие создать систему полнотекстового поиска.

В этой главе мы воспользуемся реализацией библиотеки Lucene, входящей в состав Zend Framework. В Zend Framework 1.0 имелась встроенная библиотека `Zend_Search_Lucene`, которая поддерживала индексирование и поиск средствами Lucene; в Zend Framework 2.0 эта библиотека доступна в виде компонента `ZendSearch\Lucene`, который можно загрузить и установить в веб-приложение. В этой главе мы изучим следующие темы, посвященные основам реализации полнотекстового поиска с помощью библиотеки Lucene:

- установка библиотеки `ZendSearch` в приложение;
- создание индекса для простых MySQL-данных;
- запрос Lucene-индекса;
- добавление новых файлов документов в индекс.

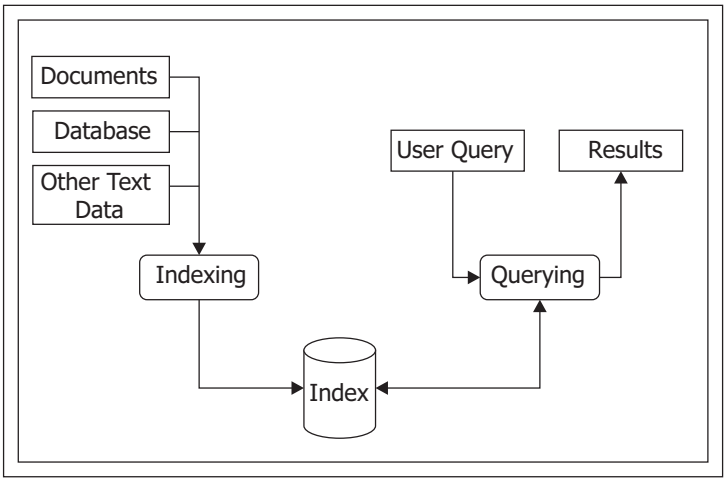
Знакомство с библиотекой Lucene

Lucene — это разработанная организацией Apache Foundation высокопроизводительная масштабируемая библиотека поиска информации, которую можно использовать в веб-приложениях для реализации механизма поиска произвольного текста. Lucene предлагает простой в использовании прикладной программный интерфейс, наделяющий веб-приложения мощными средствами индексирования и поиска. Дополнительную информацию о библиотеке Lucene можно получить по адресу http://lucene.apache.org/.

Рассмотрим ключевые компоненты библиотеки Lucene:

- **Индекс.** Lucene-индекс — это хранилище данных, которое содержит в себе все индексированные документы; при выполнении поисковых запросов искомые документы извлекаются из индекса.
- **Документ.** Lucene-документ по умолчанию является структурным элементом Lucene-индекса; документы можно сравнить с записями в таблице. Каждый документ содержит в себе набор полей, которые обрабатываются в процессе поиска.
- **Поле.** Каждый Lucene-документ состоит из одного или нескольких полей, причем необязательно, чтобы все поля были проиндексированы — хранение полей возможно и без индексирования.

Библиотека Lucene осуществляет поиск на основе индекса, поэтому для получения наилучших результатов необходимо, чтобы индекс обновлялся самым новым контентом. Процесс поиска в Lucene иллюстрирует следующая диаграмма/



Время действовать — установка библиотеки ZendSearch\Lucene

Чтобы установить библиотеку `ZendSearch\Lucene`, выполните следующие шаги.

1. На момент написания этой книги библиотека `ZendSearch\Lucene` была недоступна в виде пакета, с которым работает программа Composer. По этой причине мы получим исходный код библиотеки из репозитария GitHub по адресу https://github.com/zendframework/ZendSearch.

2. Далее переходим в папку vendor:

```
$ cd /var/www/CommunicationApp/vendor/
```

3. Выполняем клонирование поискового репозитария Zend Framework в папку vendor:

```
$ git clone https://github.com/zendframework/ZendSearch.git ZendSearch
```

4. Затем конфигурируем библиотеку ZendSearch с помощью программы Composer:

```
$ cd ZendSearch/
$ curl -s https://getcomposer.org/installer | php
$ php composer.phar install
```

5. После конфигурирования библиотеки нам нужно определить конфигурацию уровня модуля для хранения расположения индекса. Для этого мы должны изменить файл CommunicationApp/module/Users/config/module.config.php, добавив новую конфигурацию для папки search_index:

```
// КОНФИГУРАЦИИ МОДУЛЯ
'module_config' => array(
  'upload_location' => __DIR__ . '/../data/uploads',
  'image_upload_location' => __DIR__ . '/../data/images',
  'search_index' => __DIR__ . '/../data/search_index'
),
```

Что сейчас произошло?

Мы загрузили и сконфигурировали библиотеку `ZendSearch` для ZendFramework 2.0; приведенная пошаговая процедура также содержит указания на то, как загружать и устанавливать пакеты, которые невозможно получить непосредственно с помощью программы Composer.

Теперь, когда мы установили библиотеку `ZendSearch\Lucene`, наша следующая задача — создать с помощью Lucene индекс для некоторых данных, которые уже хранятся в нашем коммуникационном приложении.

Индексирование

Индексирование в библиотеке `ZendSearch\Lucene` не представляет никакой сложности. От нас требуется лишь создавать документы с полями и значениями

и добавлять их в индекс. Можно также удалять и обновлять документы, очищать индекс. Следующие классы используются при генерации индекса.

- `Field`. Класс `ZendSearch\Lucene\Document\Field` позволяет пользователям определять новые поля документов. Поле может иметь один из следующих типов:
  - `Field::keyword($name, $value, $encoding = 'UTF-8')` — тип `keyword` используется для определения строковых полей, которые не требуется маркировать, но требуется индексировать и сохранять, например даты и URL-адреса;
  - `Field::unIndexed($name, $value, $encoding = 'UTF-8')` — тип `unIndexed` служит для хранения полей в индексе без необходимости индексировать/маркировать их; примером являются поля идентификаторов;
  - `Field::binary($name, $value)` — тип `binary` позволяет хранить бинарные значения в индексе;
  - `Field::text($name, $value, $encoding = 'UTF-8')` — тип `text` самый распространенный, он используется для описания коротких строк, которые маркируются и сохраняются в индексе;
  - `Field::unStored($name, $value, $encoding = 'UTF-8')` — тип `unStored` служит для определения полей, которые маркируются и индексируются, но не сохраняются в индексе.
- `Document`. Класс `ZendSearch\Lucene\Document` позволяет определить новый документ индекса. Перечислим некоторые наиболее часто применяемые методы этого класса:
  - `addField(Document\Field $field)` — добавление нового поля в документ;
  - `getFieldNames()` — получение имен всех полей документа;
  - `getField($fieldName)` — получение заданного поля документа;
  - `getFieldValue($fieldName)` — получение значения заданного поля документа.
- `Index`. Индекс можно получить с помощью методов `create()` и `open()` класса `ZendSearch\Lucene`. Оба метода принимают путь индекса в качестве параметра и возвращают индекс типа `ZendSearch\Lucene\SearchIndexInterface`. Интерфейс `SearchIndexInterface` предоставляет следующие методы работы с документами внутри индекса:
  - `addDocument(Document $document)` — добавление нового документа в индекс;

- `delete($id)` — удаление индексированного документа с указанным идентификатором;
- `optimize()` — помогает оптимизировать индекс, объединяя все сегменты в один и тем самым повышая производительность;
- `commit()` — используется для принятия транзакций с поисковым индексом.

Теперь, когда мы ознакомились с методами, используемыми для генерации индекса, давайте приступим к действиям и сгенерируем индекс для таблицы `uploads` нашего коммуникационного приложения.

Время действовать — генерация индекса

1. Создайте новый поисковый контроллер в файле CommunicationApp/module/Users/src/Users/Controller/SearchController.php, который будет использоваться для поиска и генерации индексов.
2. Добавьте ссылки на библиотеку `ZendSearch\Lucene`:

```
use ZendSearch\Lucene;
use ZendSearch\Lucene\Document;
use ZendSearch\Lucene\Index;
```

3. Добавьте метод для выборки местоположения индекса из конфигурационных данных модуля:

```
public function getIndexLocation()
{
// выборка конфигурации из конфигурационных данных модуля
  $config = $this->getServiceLocator()->get('config');
  if ($config instanceof Traversable) {
    $config = ArrayUtils::iteratorToArray($config);
  }
  if (!empty($config['module_config']['search_index'])) {
    return $config['module_config']['search_index'];
  } else {
    return FALSE;
  }
}
```

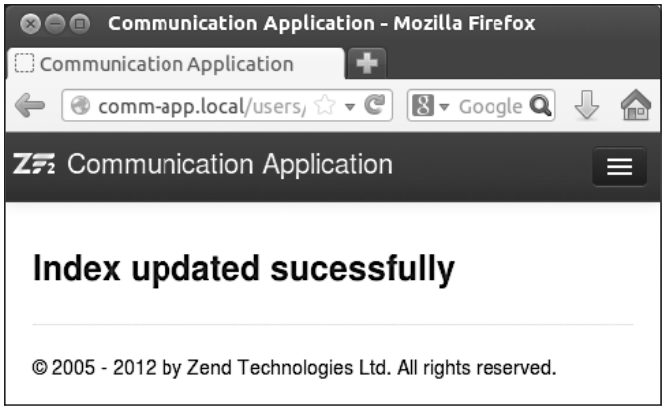
4. Документ индекса должен быть сгенерирован в следующем формате.

Поле индекса	Описание
upload_id	Неиндексированное поле, которое будет использоваться для получения выгруженного файла, возвращаемого в результатах поиска
label	Это поле используется для индексирования поля label таблицы uploads
owner	Это поле служит для индексирования поля name пользователя, который выгрузил документ

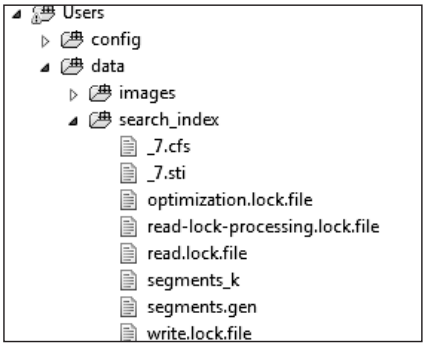
5. Создайте новое действие для генерации индекса:

```
public function generateIndexAction()
{
  $searchIndexLocation = $this->getIndexLocation();
  $index = Lucene\Lucene::create($searchIndexLocation);

  $userTable = $this->getServiceLocator()->get('UserTable');
  $uploadTable = $this->getServiceLocator()->get('UploadTable');
  $allUploads = $uploadTable->fetchAll();
  foreach($allUploads as $fileUpload) {
    $uploadOwner = $userTable->getUser($fileUpload->user_id);
    // создание полей lucene

    $fileUploadId = Document\Field::unIndexed(
      'upload_id', $fileUpload->id);
    $label = Document\Field::Text('label', $fileUpload->label);
    $owner = Document\Field::Text('owner', $uploadOwner->name);
    // создание нового документа и добавление всех полей

    $indexDoc = new Lucene\Document();
    $indexDoc->addField($label);
    $indexDoc->addField($owner);
    $indexDoc->addField($fileUploadId);
    $index->addDocument($indexDoc);
  }
  $index->commit();
}
```

6. Теперь откройте в веб-браузере URL-адрес действия (http://comm-app.local/users/search/generateIndex), и если все работает должным образом, то вы увидите файлы индекса, созданные в папке search_index.



Как видно на рисунке, файлы индекса сгенерированы и сохранены в папке search_index.



Что сейчас произошло?

Мы создали метод индексирования данных, содержащихся в MySQL-таблице хранилища данных в Lucene; наш следующий шаг — организовать выполнение запросов к Lucene-индексу, получить и вывести результаты.

Поиск

Поиск в индексе с помощью библиотеки `ZendSearch\Lucene` — относительно простая задача. Нужно открыть индекс для запросов и передать строку запроса в метод `find()` класса `ZendSearch\Lucene\Index`. Методы поиска возвращают массив с результатами, найденными по запросу, а сам массив можно использовать для визуализации результатов поиска.

Запрос к индексу можно выполнить двумя способами: передать строку запроса в виде обычного текста в функцию поиска либо создать собственный объект `Query` с помощью класса `ZendSearch\Lucene\Searh\Query`.

СОВЕТ

Дополнительную информацию о различных вариантах запросов библиотеки ZendSearch\Lucene можно получить в документации для разработчиков по адресу https://zf2.readthedocs.org/en/release-2.2.0/modules/zendsearch.lucene.queries.html.

В следующем упражнении мы будем работать с запросами в виде обычного текста, и вы сможете управлять результатами поиска с помощью операторов `:`, `+`, `-`, а также путем поиска полей. Вот несколько примеров:

- поиск всех документов, выгруженных пользователем Anne, можно выполнить с помощью запроса `owner:Anne`;
- поиск всех документов со словом `report`, выгруженных пользователем с именем Anne, можно выполнить помощью запроса `report AND owner:Anne`;
- поиск всех документов со словом `report`, за исключением тех, которые были выгружены пользователем Anne, можно выполнить с помощью запроса `report -owner:Anne`.

Время действовать — вывод результатов поиска

1. Для вывода результатов поиска мы создадим новую форму, в которой будет находиться текстовое поле поиска, и визуализируем результаты непосредственно под формой поиска. Форма будет размещена в контроллере `SearchController` (файл CommunicationApp/module/Users/src/Users/Controller/SearchController.php).

2. Создайте новое представление для вывода окна запроса и результатов поиска. Это представление будет размещено в файле CommunicationApp/module/Users/view/users/search/index.phtml.

```
<h3>Document Search</h3>
<?php
// Форма поиска
echo $this->form()->openTag($form);
foreach ($form as $element) {
  echo $this->formElement($element);
  echo $this->formElementErrors($element);
}
```

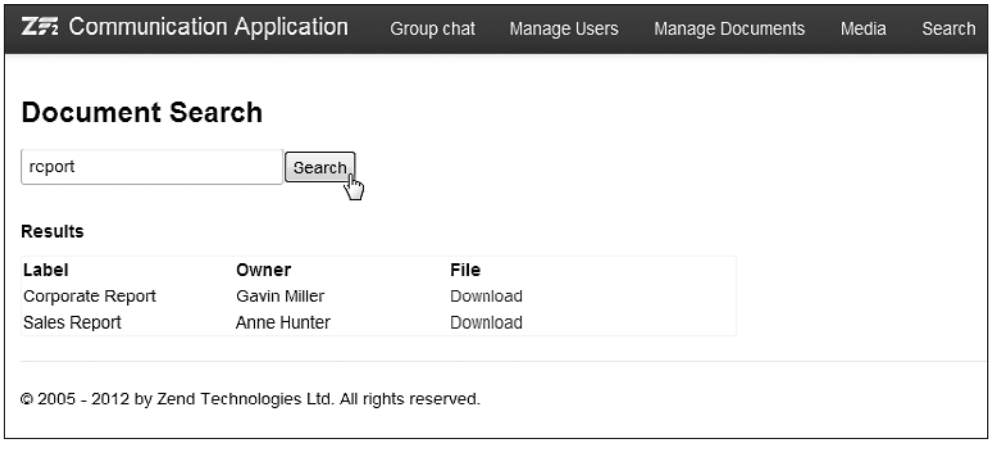
```
echo $this->form()->closeTag();

// Результаты поиска
if (count($searchResults)) {
?>
<h5>Results</h5>
<table style="width: 600px; border:1px solid #f5f5f5;">
  <tr>
  <th width="30%" align="left"> Label</th>
  <th width="30%" align="left"> Owner</th>
  <th align="left"> File</th>
  </tr>
  <?php foreach ($searchResults as $searchResult) {
  ?>
  <tr>
    <td><?php echo $searchResult->label; ?></td>
    <td><?php echo $searchResult->owner; ?></td>
    <td><a href="<?php echo $this->escapeHtml($this->url(
      'users/upload-manager', array('action'=>'fileDownload', 'id' =>
      $searchResult->upload_id)));?>">Download</a></td>
    </tr>
    <?php
    }
    ?>
    </table>
<?php }?>
```

3. Теперь создайте новое действие, которое будет выводить форму поиска и выполнять запрос к Lucene-индексу, используя информацию, введенную в форму поиска. Это действие будет размещено в контроллере `SearchController` (файл CommunicationApp/module/Users/src/Users/Controller/SearchController.php).

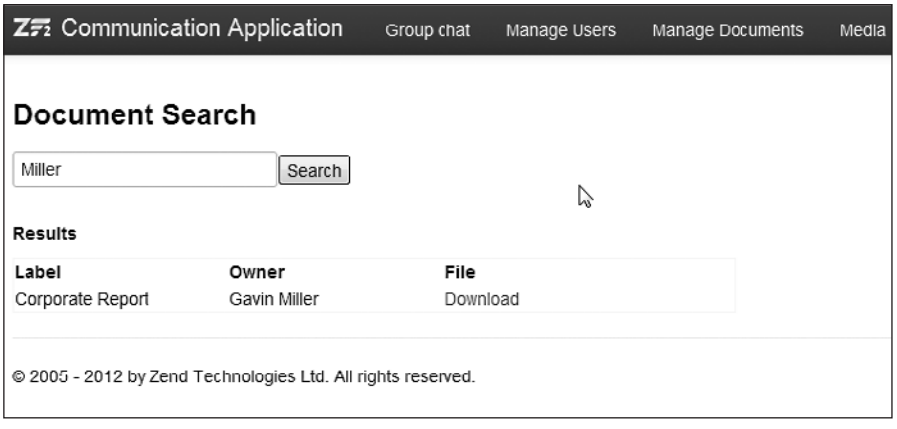
```
public function indexAction()
{
  $request = $this->getRequest();
  if ($request->isPost()) {
    $queryText = $request->getPost()->get('query');
    $searchIndexLocation = $this->getIndexLocation();
    $index = Lucene\Lucene::open($searchIndexLocation);
    $searchResults = $index->find($queryText);
  }

  // Подготовка формы поиска
  $form = new \Zend\Form\Form();
  $form->add(array(
    'name' => 'query',
    'attributes' => array(
    'type' => 'text',
```

```
        'id' => 'queryText',
        'required' => 'required'
      ),
        'options' => array(
        'label' => 'Search String',
        ),
      ));
      $form->add(array(
        'name' => 'submit',
        'attributes' => array(
          'type' => 'submit',
          'value' => 'Search'
        ),
      ));
      $viewModel = new ViewModel(array(
        'form' => $form,
        'searchResults' => $searchResults
        )
      );
      return $viewModel;
    }
```

4. Протестируйте страницу в браузере; вы должны увидеть результаты поиска для ключевых слов в полях Label и Owner.



При поиске по полю Owner вы получите следующие результаты.



Что сейчас произошло?

Мы создали страницу результатов поиска, которая позволяет нам искать выгруженные документы по их меткам и владельцам. Полученные результаты выводятся в настроенном представлении, с помощью которого мы можем загрузить документ из результатов поиска.

Наш следующий шаг — распространить возможности поиска на контент выгруженных документов; для этого нам понадобится изменить способ генерации индекса.

Индексирование документов Microsoft Office

Как мы видели в предыдущем примере, индексирования метаинформации документа обычно недостаточно. Как правило, запрашиваемая строка присутствует только в контенте документа. Чтобы осуществить такой поиск, нам нужно выполнить синтаксический разбор документа и проиндексировать его контент; библиотека `ZendSearch\Lucene` предлагает методы для индексирования контента документов следующих типов.

- Для HTML-документов:

```
ZendSearch\Lucene\Document\Html::loadHTMLFile($filename)
ZendSearch\Lucene\Document\Html::loadHTML($htmlString)
```

- Для документов Word 2007:

```
ZendSearch\Lucene\Document\Docx::loadDocxFile($filename)
```

- Для документов Powerpoint 2007:

```
ZendSearch\Lucene\Document\Pptx::loadPptxFile($filename)
```

- Для документов Excel 2007:

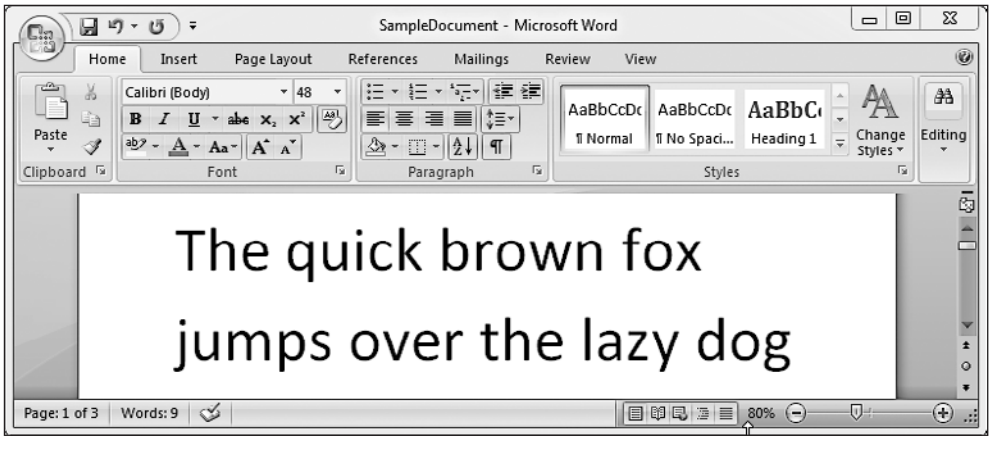
```
ZendSearch\Lucene\Document\Xlsx::loadXlsxFile($filename)
```

Все эти методы возвращают документ типа `ZendSearch\Lucene\Document`, в который можно в дальнейшем добавлять другие индексные поля.

Мы начнем с того, что проиндексируем документы, доступные в разделе `uploads`.

Время действовать — индексирование файлов документов

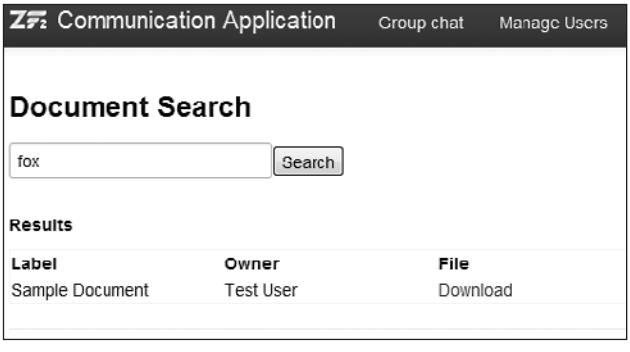
1. Чтобы проиндексировать офисные документы, добавьте новый раздел `uploads` для образцов документов Word и Excel. Мы выгрузим документ Word и электронную таблицу Excel.



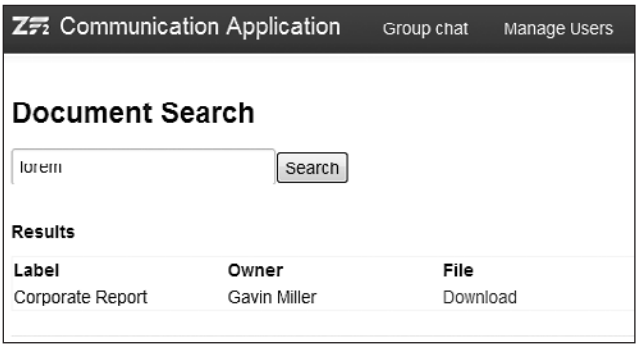
2. Добавьте указанные строки к функции индексирования, находящейся в контроллере `SearchController` (файл CommunicationApp/module/Users/src/Users/Controller/SearchController.php), чтобы она считывала и индексировала документы Word и электронные таблицы Excel по отдельности:

```
if (substr_compare($fileUpload->filename,
    ".xlsx",
    strlen($fileUpload->filename) - strlen(".xlsx"),
    strlen(".xlsx")) === 0) {
  // Индексирование таблицы excel
  $uploadPath = $this->getFileUploadLocation();
  $indexDoc = Lucene\Document\Xlsx::loadXlsxFile(
    $uploadPath ."/" . $fileUpload->filename);
  } else if (substr_compare($fileUpload->filename,
    ".docx",
    strlen($fileUpload->filename) - strlen(".docx"),
    strlen(".docx")) === 0) {
  // Индексирование документа Word
  $uploadPath= $this->getFileUploadLocation();
  $indexDoc = Lucene\Document\Docx::loadDocxFile(
    $uploadPath ."/" . $fileUpload->filename);
  } else {
    $indexDoc = new Lucene\Document();
  }

  $indexDoc->addField($label);
  $indexDoc->addField($owner);
  $indexDoc->addField($fileUploadId);
  $index->addDocument($indexDoc);
```

3. Теперь обновите индекс (перейдите по адресу http://comm-app.local/users/search/generateIndex), вернитесь на страницу Document Search (Поиск документа) и попробуйте выполнить поиск по ключевым словам, которые присутствуют в документе. Вы должны увидеть результаты поиска, как показано на рисунке.



Результаты поиска в контенте документов Office будут выглядеть следующим образом.



Что сейчас произошло?

В последнем упражнении мы познакомились с реализацией индексирования и поиска в контенте документов Microsoft Office. Как видите, реализовать эти возможности с помощью библиотеки `ZendSearch\Lucene` относительно несложно.

Самостоятельная работа

Перед тем как приступить к изучению следующей главы, выполним несложное задание. Теперь, когда мы реализовали индексирование и поиск, измените приложение так, чтобы индекс обновлялся каждый раз при изменении выгруженных

файлов. При выгрузке нового файла документ должен добавляться в индекс, при удалении выгруженного файла документ должен удаляться из индекса и т. д.

Контрольные вопросы

1. Поля какого типа не маркируются, но индексируются и сохраняются?

   1) `keyword`;

   2) `unStored`;

   3) `text`;

   4) `unIndexed`.

2. Для файлов какого формата библиотека `ZendSearch\Lucene` не поддерживает индексирование контента?

   1) .docx;

   2) .pdf;

   3) .xslx;

   4) .html.

Заключение

В этой главе мы изучили реализацию простого поискового интерфейса с использованием библиотеки `ZendSearch\Lucene`. Эти знания будут очень полезны при создании поисковых механизмов в любом веб-приложении, с которым вам придется работать. В следующей главе мы узнаем, как реализовать простой интернет-магазин с помощью Zend Framework 2.0.

Создание простого магазина

8

За последние несколько лет электронная коммерция превратилась из простой онлайновой рекламы в полнофункциональные онлайновые магазины. Все больше и больше продуктов и услуг можно заказать через Интернет с помощью различных сетевых платежных систем. В такой среде значительную роль стали играть приложения электронной торговли и платежные шлюзы.

В этой главе мы создадим простой онлайновый магазин, чтобы продемонстрировать процесс разработки виртуальной товарной корзины. Для обслуживания платежей в нашем примере будет использоваться платежная система PayPal Express Checkout. Вот несколько ключевых тем этой главы:

- разработка товарной корзины;
- создание административного интерфейса для онлайнового магазина;
- конфигурирование Zend Framework 2.0 для использования системы PayPal;
- знакомство с системой PayPal Express Checkout;
- реализация системы PayPal Express Checkout.

Товарная корзина

При создании онлайнового магазина одной из первых необходимо разработать товарную корзину. В идеале корзина должна давать конечному пользователю возможность выбирать и добавлять в корзину различные продукты и расплачиваться за них на веб-сайте.

Процедура покупки происходит следующим образом.

1. Покупатель посещает страницу, где перечислены продукты.

2. Покупатель выбирает продукт и попадает на страницу с его подробным описанием.

3. Покупатель принимает решение приобрести продукт и помещает его в корзину в желаемом количестве.

4. Покупатель перенаправляется на страницу товарной корзины; здесь он может при необходимости внести изменения в свой заказ.

5. Покупатель выбирает режим оплаты и вводит информацию для совершения платежа.

6. Если операция проходит успешно, то покупатель получает возможность обновить параметры отгрузки товара.

7. Покупатель подтверждает заказ.

8. Заказ поступает ритейлеру, где он обрабатывается для передачи покупателю.

Итак, давайте приступим к созданию витрины нашего магазина; наш первый шаг — разработать структуру таблиц, которая будет поддерживать процесс торговли. Мы создадим две таблицы:

- `store_products` — таблица со всей информацией о продуктах;
- `store_orders` — таблица со всей информацией о заказах.

Время действовать — создание витрины магазина

Чтобы упростить задачу, мы сократим процедуру покупки, опустив некоторые ее шаги. В нашем примере заказ будет включать в себя только один продукт; мы также исключим этапы обновления параметров отгрузки и подтверждения заказа.

1. Создайте таблицы для хранения данных о продуктах и заказах:

```
CREATE TABLE IF NOT EXISTS store_products (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(255) NOT NULL,
  desc varchar(255) NOT NULL,
  cost float(9,2) NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE IF NOT EXISTS store_orders (
  id int(11) NOT NULL AUTO_INCREMENT,
```

```
    store_product_id int(11) NOT NULL,
    qty int(11) NOT NULL,
    total float(9,2) NOT NULL,
    status enum('new', 'completed', 'shipped', 'cancelled') DEFAULT NULL,
    stamp timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    first_name varchar(255) DEFAULT NULL,
    last_name varchar(255) DEFAULT NULL,
    email varchar(255) DEFAULT NULL,
    ship_to_street varchar(255) DEFAULT NULL,
    ship_to_city varchar(255) DEFAULT NULL,
    ship_to_state varchar(2) DEFAULT NULL,
    ship_to_zip int(11) DEFAULT NULL,
    PRIMARY KEY (id)
);
```

2. Создайте сущности для таблиц `StoreOrder` и `StoreProduct`, а также необходимые объекты шлюзов для доступа к данным таблиц.

3. Создайте контроллер `StoreController`, который будет использоваться в качестве товарной корзины.

4. Контроллер `StoreController` должен поддерживать следующие действия:

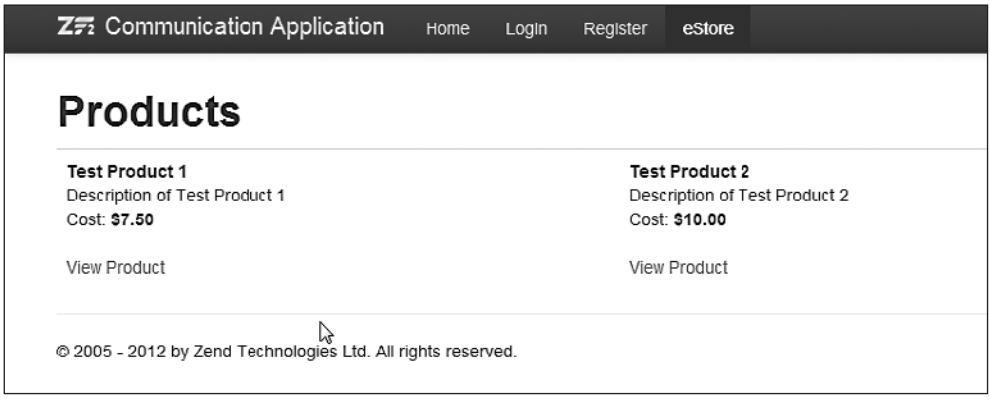
   - `indexAction()` — перечисление всех продуктов на веб-сайте;
   - `productDetailAction()` — вывод подробной информации об определенном продукте и предоставление покупателю возможности добавить продукт в корзину;
   - `shoppingCartAction()` — визуализация товарной корзины перед началом оплаты;
   - `paypalExpressCheckoutAction()` — перенаправление пользователя на страницу платежной системы PayPal Express Checkout;
   - `paymentConfirmAction()` — перенаправление со страницы PayPal Express Checkout обратно к корзине после успешной оплаты;
   - `paymentCancelAction()` — перенаправление со страницы PayPal Express Checkout обратно к корзине после неуспешной оплаты.

5. Создайте необходимые представления для показа содержимого товарной корзины.

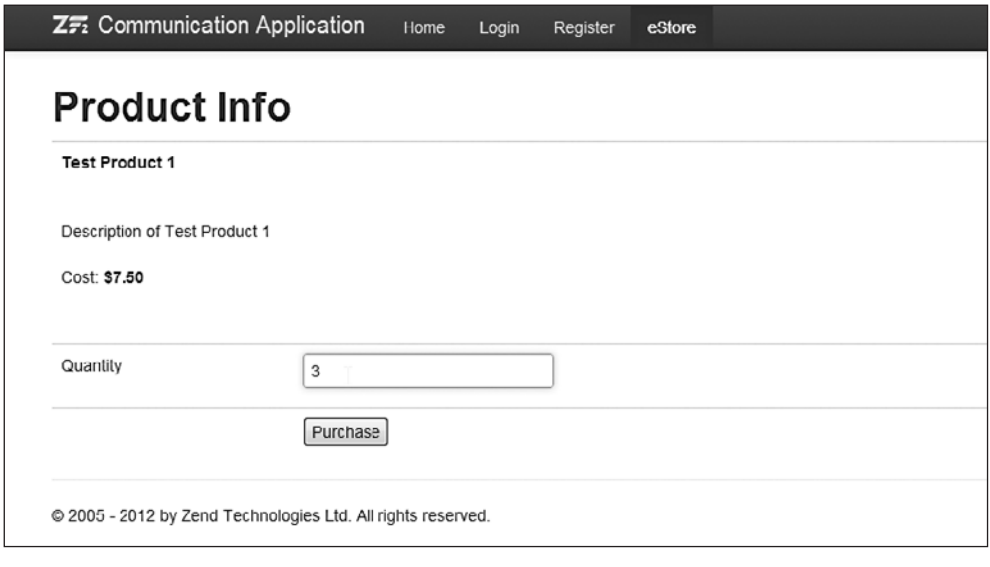
6. Добавьте в таблицу `StoreOrder` необходимые методы для подсчета суммы заказа при добавлении в нее продуктов.

7. В итоге пользовательский интерфейс должен выглядеть так, как показано на рисунке. На странице со списком продуктов выводятся все продукты

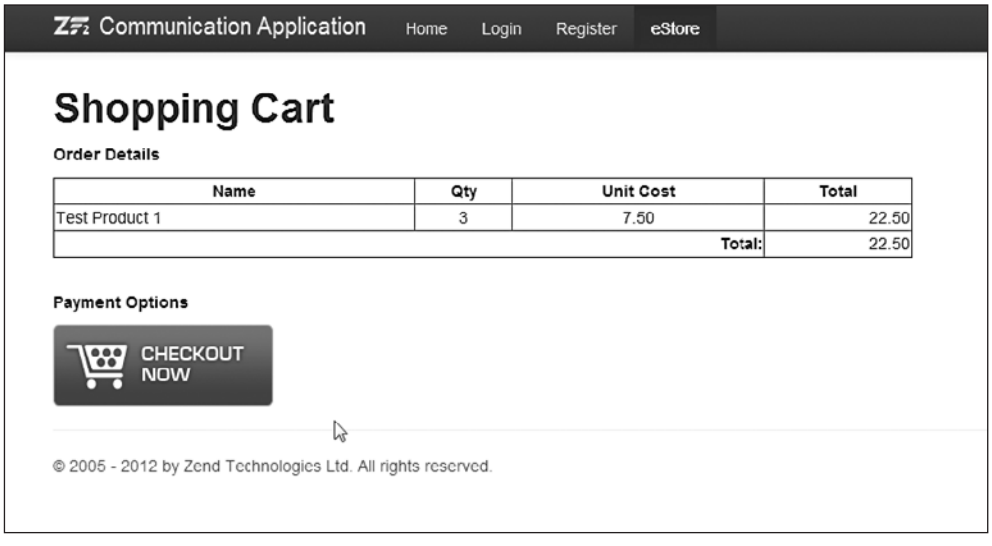
веб-сайта/категории; в данном случае на рисунке показаны два тестовых продукта.



Страница с подробной информацией о продукте позволяет покупателю познакомиться с особенностями продукта и добавить его в корзину в указываемом количестве.



На странице Shopping Cart (Товарная корзина) перечисляются все продукты, добавленные в товарную корзину, с указанием цен за единицу товара, их количества и промежуточной суммы.



Что сейчас произошло?

Мы создали интерфейс товарной корзины для нашего нового магазина. Позже мы добавим в него механизм поддержки платежей, но прежде чем мы сделаем это, давайте создадим для нашего магазина простой административный интерфейс, позволяющий управлять товарным складом и заказами.

Администрирование товарного склада

Интерфейс администрирования товарного склада служит для проверки состояния заказов после их создания и для управления списками продуктов, находящихся на складе и доступных для продажи. В отношении административного пользовательского интерфейса склада нужно отметить два ключевых аспекта:

- администратор должен иметь возможность добавлять, удалять и контролировать продукты;
- администратор должен иметь возможность проконтролировать заказ и изменить его статус.

Время действовать — создание административного интерфейса товарного склада

1. Создайте новый контроллер для администрирования склада и назовите его `StoreAdminController`.

2. Основные действия этого контроллера:

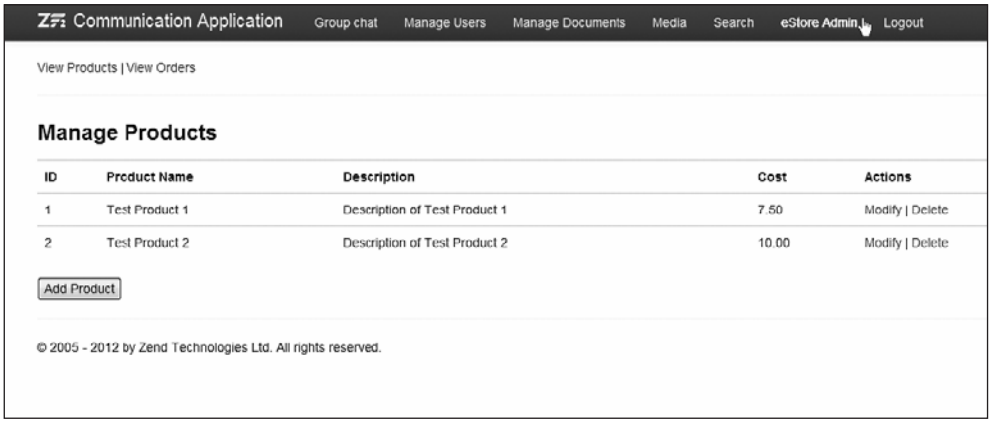
   - `indexAction()` — перечисление всех продуктов;
   - `addProductAction()` — добавление нового продукта;
   - `deleteProductAction()` — удаление существующего продукта;
   - `listOrdersAction()` — перечисление всех заказов;
   - `viewOrderAction()` — просмотр конкретного заказа;
   - `updateOrderStatusAction()` — обновление статуса заказа.

3. Создайте необходимые представления и свяжите с ними соответствующие действия.

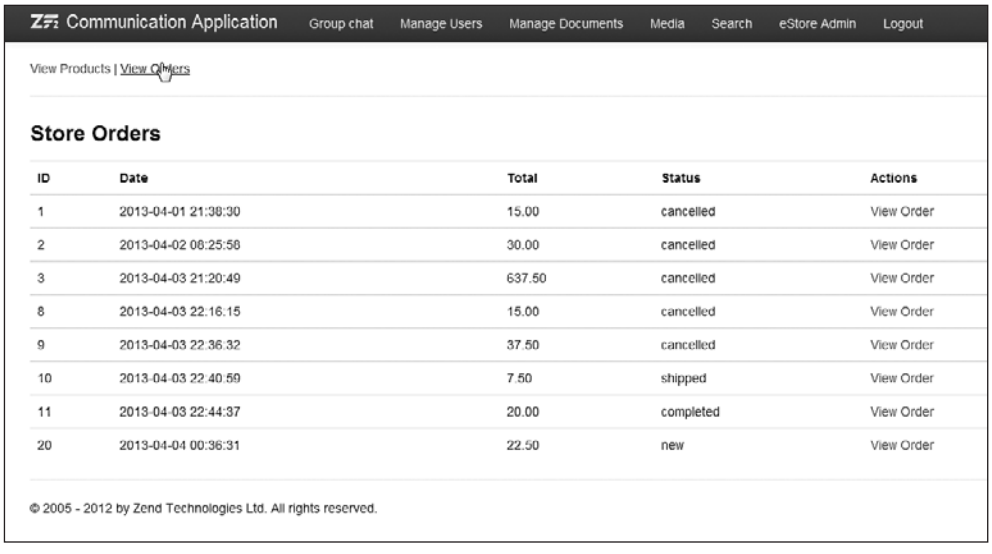
4. Откройте инструмент `phpMyadmin` и создайте тестовые записи в таблицах `store_products` и `store_orders`, чтобы проверить функциональность административного пользовательского интерфейса.

5. Откройте выбранный по вашему желанию браузер, войдите в приложение и откройте интерфейс eStore Admin (Администратор электронного магазина).

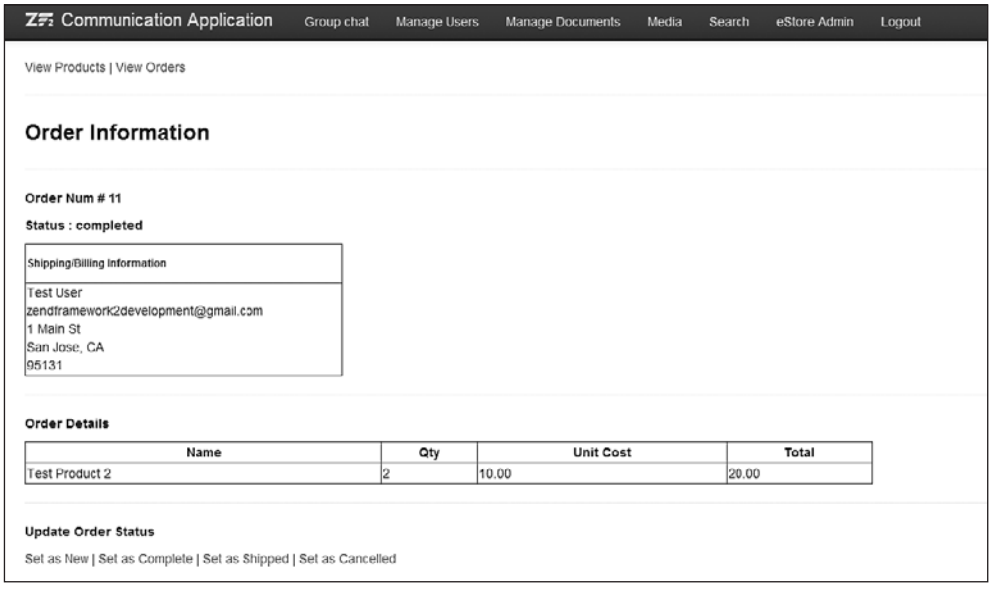
   Страница Manage Products (Управление продуктами) позволяет добавлять, удалять и менять продукты с помощью административного интерфейса.



На странице со списком заказов перечисляются все заказы, сделанные в магазине, причем каждый заказ можно просмотреть и изменить его статус.



Страница Order Information (Сведения о заказе) с информацией о заказах и элементами интерфейса, позволяющими менять их статус, показана на рисунке.



Что сейчас произошло?

Административный пользовательский интерфейс товарного склада готов, и наш следующий шаг — установить платежную систему PayPal Express и интегрировать ее с нашим магазином, тем самым давая пользователям возможность расплачиваться за товары с помощью системы PayPal. Но перед тем как мы перейдем к следующему разделу, сделайте дополнительное несложное задание.

Самостоятельная работа

Теперь, когда вы знаете, как встроить поисковые возможности в приложение Zend Framework 2.0, попробуйте добавить функцию поиска произвольного текста в раздел Manage Products нашего приложения онлайнового магазина.

Совершение платежей с помощью PayPal

PayPal является самой распространенной платежной системой в мире; один из ключевых факторов ее успеха — простой в использовании прикладной программный интерфейс и исчерпывающая документация, поддерживающая этот платежный шлюз. Система PayPal предоставляет новым клиентам разнообразные возможности по установке платежного механизма, причем среди этих возможностей особый интерес представляют доступные варианты интеграции. PayPal предлагает различные платежные продукты, в том числе:

- Express Checkout;
- PayPal Payments Standards (Website Payments Standards);
- PayPal Payments Pro (Website Payments Pro).

В этой главе мы будем работать с продуктом Express Checkout, поскольку он является наиболее простой реализацией платежной системы PayPal.

PayPal и Zend Framework 2.0

На момент написания этой книги в Zend Framework отсутствовали встроенные пакеты, поддерживающие интеграцию с системой PayPal. Такая интеграция может быть выполнена средствами сторонних производителей. В данном примере мы воспользовались одним из таких пакетов под названием SpeckPaypal.

Время действовать — установка платежной системы PayPal

1. Откройте сайт https://packagist.org/ и выполните поиск по ключевому слову `speckpaypal`.

2. Ознакомьтесь с подробной информацией о репозитарии.

3. Измените конфигурационный файл приложения Composer, включив в него репозитарий `speckpaypal`:

```
"require": {
  "php": ">=5.3.3",
  "zendframework/zendframework": "2.0.*",
  "webino/webino-image-thumb": "1.0.0",
  "zendframework/zendgdata": "2.*",
  "speckcommerce/speck-paypal": "dev-master"
}
```

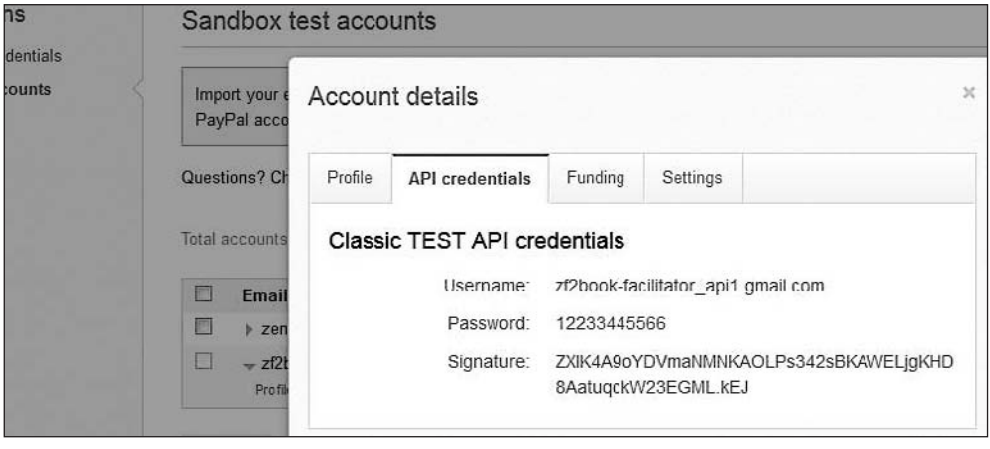
4. Обновите зависимости проекта с помощью команды обновления программы Composer:

```
.
Loading composer repositories with package information
Updating dependencies
  - Removing zendframework/zendframework (2.0.7)
  - Installing zendframework/zendframework (2.0.8)
  Downloading: 100%

  - Installing speckcommerce/speck-paypal (dev-master d951518)
  Cloning d951518fd2c98148da5609e23a41697e6cfca06e

Writing lock file
Generating autoload files
```

5. Теперь нам понадобятся учетные данные прикладного программного интерфейса (API) для доступа к модулю PayPal Express Checkout. Такой доступ можно получить, войдя на сайт https://developer.paypal.com с вашими учетными данными в системе PayPal.

6. Откройте раздел Sandbox Accounts (Учетные записи среды Sandbox) в разделе Applications (Приложения).

7. Выберите соответствующую учетную запись продавца и перейдите на вкладку API Credentials (Учетные данные API) в разделе Profile (Профиль).



8. Запишите учетные данные прикладного программного интерфейса.

9. Теперь создайте новую конфигурацию в конфигурационном файле модуля (CommunicationApp/module/Users/config/module.config.php) и назовите индекс массивов `speck-paypal-api`:

```
'speck-paypal-api' => array(
  'username' => '',
  'password' => '',
  'signature' => '',
  'endpoint' => 'https://api-3t.sandbox.paypal.com/nvp'
)
```

10. Различные службы PayPal имеют разные конечные точки; у Express Checkout конечной точкой является адрес https://api-3t.sandbox.paypal.com/nvp. При переходе в реальную рабочую среду конечную точку нужно заменить на https://api-3t.paypal.com/nvp.

Что сейчас произошло?

Мы установили системы PayPal и SpeckPayPal в нашем приложении, и теперь наш следующий шаг — протестировать механизм получения платежей, совершаемых с помощью платежной системы PayPal Express Checkout.

Платежная система PayPal Express Checkout

Платежная система PayPal Express Checkout позволяет продавцам принимать платежи с кредитных карт или через систему PayPal со своих веб-сайтов, перенаправляя покупателей в систему PayPal Express Checkout для безопасного совершения платежа и возвращая их на сайт продавца после завершения транзакции.

Процедура использования PayPal Express Checkout состоит из следующих этапов.

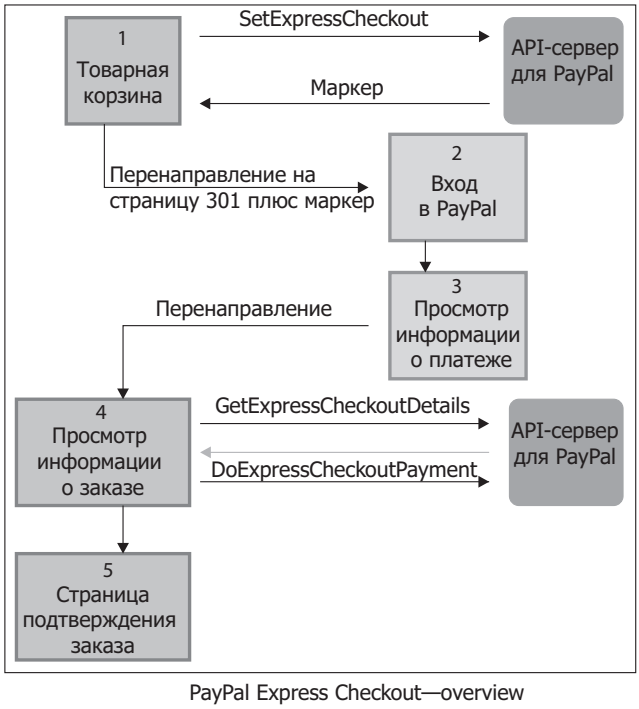
1. Покупатель на странице Shopping Cart (Товарная корзина) выбирает вариант оплаты с помощью PayPal Express Checkout; продавец вызывает прикладной программный интерфейс `SetExpressCheckout` и получает платежный маркер.

2. Покупатель перенаправляется на страницу входа в PayPal Express Checkout с использованием платежного маркера; здесь покупатель может ввести информацию для входа в PayPal или создать в PayPal новую учетную запись.

3. На следующей странице покупателю предлагается возможность просмотра информации о платеже перед продолжением процесса оплаты.

4. Далее покупатель перенаправляется обратно на страницу продавца, а затем продавец вызывает прикладной программный интерфейс `GetExpressCheckoutDetails` и получает информацию о покупателе. Покупатель просматривает информацию о заказе и подтверждает его. Продавец завершает обработку платежа, вызывая прикладной программный интерфейс `DoExpressCheckoutPayment`.

5. Покупателю выводятся результаты транзакции и сводная информация о заказе.



PayPal Express Checkout—overview

ДОПОЛНИТЕЛЬНО О PAYPAL EXPRESS CHECKOUT

Более подробную информацию о PayPal Express Checkout можно получить на веб-сайте PayPal https://www.paypal.com/webapps/mpp/express-checkout.

Документация о PayPal Express Checkout для разработчиков доступна по адресу https://developer.paypal.com/webapps/developer/docs/classic/express-checkout/integration-guide/ECGettingStarted/.

Время действовать — прием платежей с помощью PayPal

Чтобы реализовать механизм приема платежей с помощью PayPal, выполните следующие действия.

1. Добавьте кнопку на страницу Shopping Cart (Товарная корзина) с логотипом Checkout by PayPal (при желании). Эта кнопка должна быть связана с функцией `paypalExpressCheckoutAction()`.

2. Добавьте в контроллер склада метод, который будет использоваться для генерации запроса к PayPal:

```
protected function getPaypalRequest()
{
  $config = $this->getServiceLocator()->get('config');
  $paypalConfig = new \SpeckPaypal\Element\Config(
    $config['speck-paypal-api']);

  $adapter = new \Zend\Http\Client\Adapter\Curl();
  $adapter->setOptions(array(
    'curloptions' => array(
      CURLOPT_SSL_VERIFYPEER => false,
    )
  ));

  $client = new \Zend\Http\Client;
  $client->setMethod('POST');
  $client->setAdapter($adapter);

  $paypalRequest = new \SpeckPaypal\Service\Request;
  $paypalRequest->setClient($client);
  $paypalRequest->setConfig($paypalConfig);

  return $paypalRequest;
}
```

3. Измените функцию `paypalExpressCheckoutAction()` так, чтобы она передавала информацию о заказе в PayPal и перенаправляла покупателя в систему PayPal Express Checkout:

```
public function paypalExpressCheckoutAction()
{
  $request = $this->getRequest();
  $orderId = $request->getPost()->get('orderId');

  $orderTable = $this->getServiceLocator()->get('StoreOrdersTable');
  $order = $orderTable->getOrder($orderId);

  $paypalRequest = $this->getPaypalRequest();

  $paymentDetails = new \SpeckPaypal\Element\PaymentDetails
  (array('amt' => $order->total));
  $express = new \SpeckPaypal\Request\SetExpressCheckout(
    array('paymentDetails' => $paymentDetails)
  );

  $express->setReturnUrl(
    'http://comm-app.local/users/store/paymentConfirm');
  $express->setCancelUrl(
    'http://comm-app.local/users/store/paymentCancel');

  // Отправка информации о заказе в PayPal
  $response = $paypalRequest->send($express);
  $token = $response->getToken();

  $paypalSession = new \Zend\Session\Container('paypal');
  $paypalSession->tokenId = $token;
  $paypalSession->orderId = $orderId;

  // Перенаправление покупателя в PayPal Express Checkout
  $this->redirect()->toUrl('https://www.sandbox.paypal.com/
  webscr?cmd=_express-checkout&token=' . $token);
}
```

4. Добавьте метод `paymentConfirmAction()` для обработки успешного платежа от Express Checkout; этот метод будет получать информацию о платеже от PayPal, подтверждать платеж и обновлять статус заказа в нашей системе с помощью следующего кода.

   - Получение информации о платеже от PayPal:

```
// Получение информации о плательщике от PayPal
$paypalSession = new \Zend\Session\Container('paypal');
$paypalRequest = $this->getPaypalRequest();

$expressCheckoutInfo =
  new \SpeckPaypal\Request\GetExpressCheckoutDetails();
$expressCheckoutInfo->setToken($paypalSession->tokenId);
$response = $paypalRequest->send($expressCheckoutInfo);
```

- Подтверждение заказа в PayPal:

```
// Получение экспресс-платежа
$orderTable = $this->getServiceLocator()->get('StoreOrdersTable');
$order = $orderTable->getOrder($paypalSession->orderId);
$paymentDetails = new \SpeckPaypal\Element\PaymentDetails(
  array('amt' => $order->total
));

$token = $response->getToken();
$payerId = $response->getPayerId();

$captureExpress =
        new \SpeckPaypal\Request\DoExpressCheckoutPayment(
  array(
    'token' => $token,
    'payerId' => $payerId,
    'paymentDetails' => $paymentDetails
));
$confirmPaymentResponse = $paypalRequest->send($captureExpress);
```

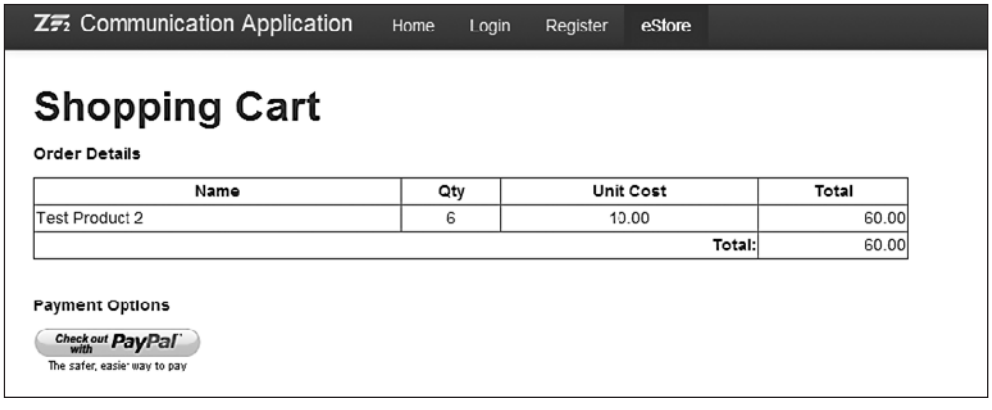
- Сохранение заказа с обновленной информацией об отгрузке и оплате:

```
// Сохранение информации о заказе
$order->first_name = $response->getFirstName();
$order->last_name = $response->getLastName();
$order->ship_to_street = $response->getShipToStreet();
$order->ship_to_city = $response->getShipToCity();
$order->ship_to_state = $response->getShipToState();
$order->ship_to_zip = $response->getShipToZip();
$order->email = $response->getEmail();
$order->store_order_id = $paypalSession->orderId;
$order->status = 'completed';
$orderTable->saveOrder($order);
```

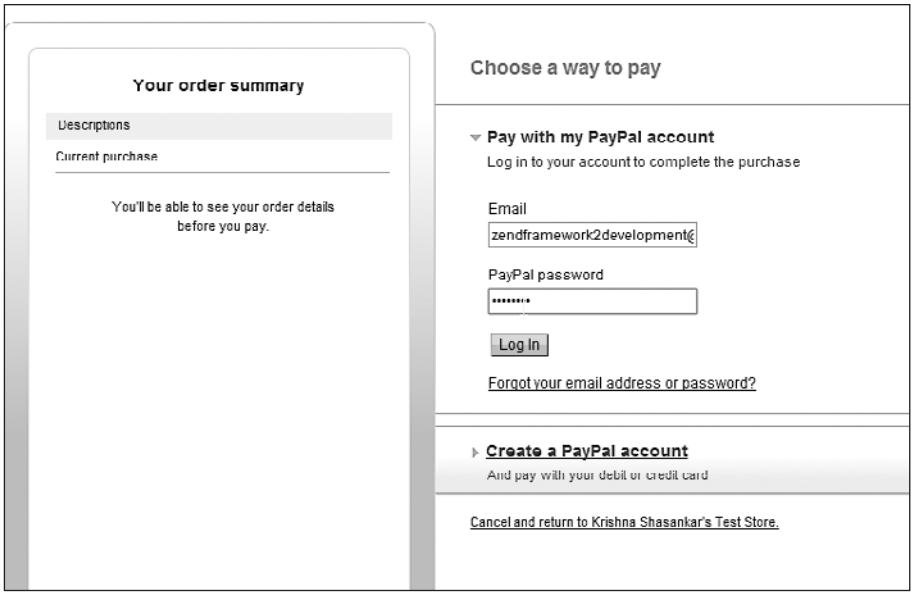
5. Добавьте метод `paymentCancelAction()`, обрабатывающий неудачный платеж от Express Checkout:

```
public function paymentCancelAction()
{
  $paypalSession = new \Zend\Session\Container('paypal');
  $storeOrdersTG = $this->getServiceLocator()->get(
    'StoreOrdersTableGateway');
  $storeOrdersTG->update(
     array('status' => 'cancelled'),
     array('id' => $paypalSession->orderId));
  $paypalSession->orderId = NULL;
  $paypalSession->tokenId = NULL;
  $view = new ViewModel();
  return $view;
}
```

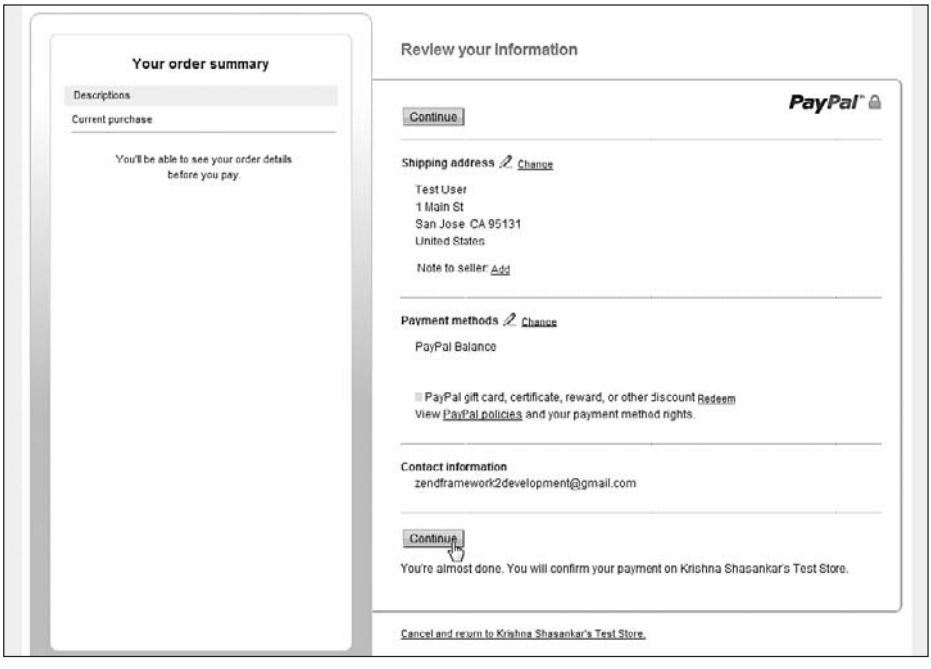
6. Теперь снова войдите на сайт https://developer.paypal.com.
7. Сгенерируйте в среде Sandbox новую учетную запись типа PERSONAL.
8. Получите доступ к магазину и попробуйте сделать заказ с помощью новой учетной записи среды Sandbox. В итоге магазин должен выглядеть так, как показано на рисунке.



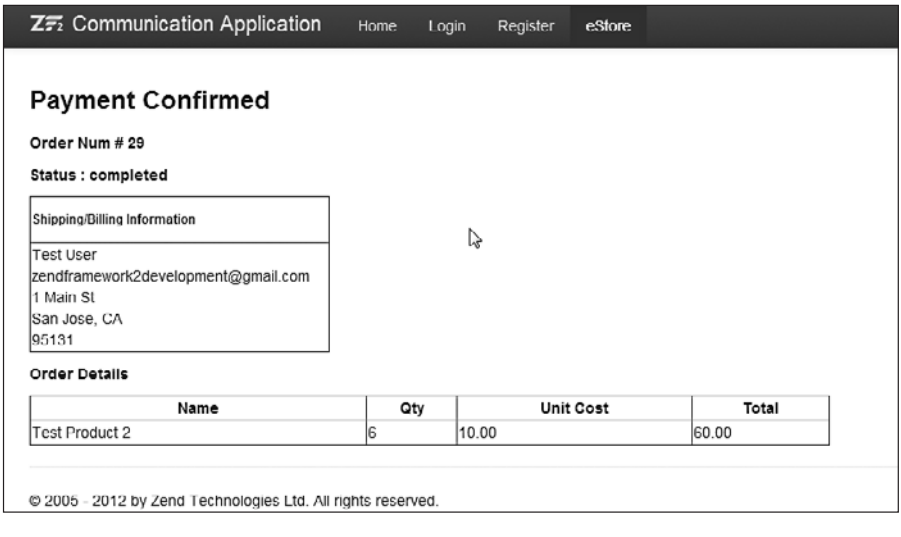
После того как вы дадите команду рассчитаться за товар на странице Shopping Cart, вы будете перенаправлены на страницу входа Pay with my PayPal account (Оплата с использованием моей учетной записи PayPal), как показано на следующем рисунке.



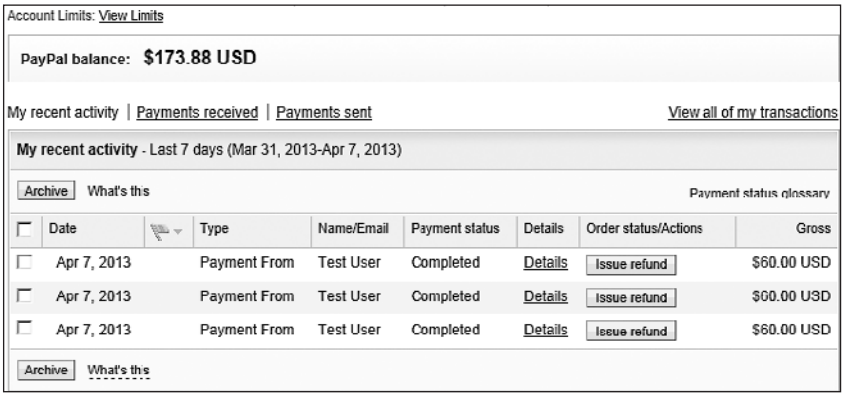
На следующем рисунке изображена страница просмотра заказа в PayPal Express Checkout; она используется для вывода информации о платеже, который совершается из учетной записи PayPal покупателя в адрес продавца.



После успешного размещения заказа пользователь перенаправляется на страницу подтверждения заказа.



9. Теперь войдите на сайт Sandbox учетной записи продавца, чтобы убедиться, что платежи проведены.



Что сейчас произошло?

Мы воспользовались системой PayPal Express Checkout для приема платежей в нашем веб-приложении и завершили создание простого онлайнового магазина. Как видите, API системы PayPal позволяет относительно легко установить платежный шлюз.

Самостоятельная работа

Сделайте еще одно задание: с помощью API `DoDirectPayment` выполните платеж напрямую на веб-сайте без перенаправления покупателя на сайт системы PayPal и обратно.

Контрольные вопросы

1. Какой из перечисленных методов используется для отправки начальной информации о платеже для перенаправления в систему PayPal?

   1) `RedirectExpressCheckout`;

   2) `SetExpressCheckout`;

   3) `GetExpressCheckoutDetails`;

   4) `DoExpressCheckoutPayment`.

2. Какое из перечисленных полей необходимо для запроса информации о платеже у системы PayPal?

   1) `token`;

   2) `payerId`;

   3) `paymentDetails`;

   4) `orderID`.

Заключение

В этой главе мы получили базовые сведения по созданию простого онлайнового магазина и получения платежей через систему PayPal. Как показывают приведенные примеры, использование модулей Zend Framework упрощает разработку приложения, поскольку позволяет разработчикам загружать и устанавливать модули сторонних производителей, которые необходимо интегрировать в приложение. В следующей главе с помощью Zend Framework 2.0 мы будем работать со спецификацией HTML5.

Поддержка HTML5

9

HTML5 — это последняя версия спецификации HTML; ее окончательный проект вряд ли будет закончен в ближайшее время, однако большинство браузеров поддерживают основную часть функциональных возможностей, описанных в последнем проекте этой спецификации.

Перечислим все наиболее важное, что предлагает HTML5:

- аудио- и видеотеги;
- поддержка спецификации CSS3;
- поддержка двух- и трехмерной графики средствами SVG и CSS3;
- локальные хранилища, веб-процессы, геолокация;
- элементы форм в HTML5.

В рамках этой книги мы сконцентрируем основное внимание на множестве новых элементов форм, которые появились в HTML5. В предыдущих версиях HTML веб-разработчики были вынуждены пользоваться лишь ограниченным набором стандартных средств ввода, определенных в более ранних спецификациях HTML. С появлением HTML5 в нашем распоряжении появились разнообразные элементы для различных способов ввода пользовательской информации.

Новые элементы ввода, доступные в HTML5:

- `datetime`;
- `datetime-local`;
- `time`;
- `date`;
- `week`;
- `month`;

- email;
- url;
- number;
- range;
- color;
- tel;
- search.

СПЕЦИФИКАЦИЯ HTML5

Более подробно ознакомиться со спецификацией HTML 5.0 можно на веб-сайте консорциума W3C: http://www.w3.org/TR/html5/.

Следующая ссылка указывает на спецификацию элемента <input>: http://www.w3.org/TR/html5/forms.html#the-input-element.

В этой главе мы научимся использовать эти элементы ввода.

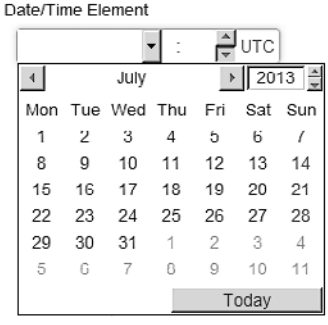
Элементы ввода в HTML5

В настоящее время Zend Framework 2.0 поддерживает все новые типы ввода, появившиеся в HTML5. Все эти типы, наряду с остальными, доступны в компоненте `Zend\Form\Element`. Далее перечислены новые типы ввода, а также указаны соответствующие элементы и даны их описания.

- Тип `datetime`.
  - Элемент: `Zend\Form\Element\DateTime`.
  - Служит для визуализации поля ввода Date/Time Element (Элемент даты/времени) с часовым поясом, установленным в значение UTC.
  - HTML-тег:

```
<input type="datetime" name="element-date-time">
```

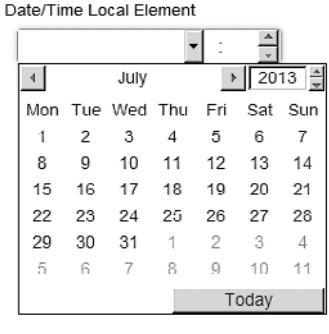
- Вид элемента типа `datetime` в браузере Opera 12.0.



- Тип `datetime-local`.
  - Элемент: `Zend\Form\Element\DateTimeLocal`.
  - Служит для визуализации поля ввода Date/Time Local Element (Элемент локальной даты/времени) для часового пояса браузера клиента.
  - HTML-тег:

```
<input type="datetime-local" name="element-date-time-local">
```

  - Вид элемента типа `datetime-local` в браузере Opera 12.0.



- Тип `time`.
  - Элемент: `Zend\Form\Element\Time`.
  - Используется для визуализации поля Time Element (Элемент времени).
  - HTML-тег:

```
<input type="time" name="element-time">
```

- Вид элемента типа `time` в браузере Opera 12.0.

- Тип `date`.
  - Элемент: `Zend\Form\Element\Date`.
  - Используется для визуализации поля Date Element (Элемент даты).
  - HTML-тег:

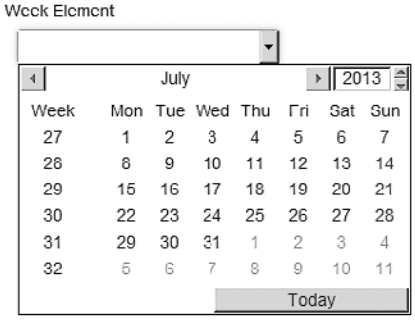
```
<input type="date" name="element-date">
```

  - Вид элемента типа `date` в браузере Opera 12.0.

- Тип `week`.
  - Элемент: `Zend\Form\Element\Week`.
  - Используется для визуализации поля Week Element (Элемент недели).
  - HTML-тег:

```
<input type="week" name="element-week">
```

- Вид элемента типа `week` в браузере Opera 12.0.



- Тип `month`.
  - Элемент: `Zend\Form\Element\Month`.
  - Используется для визуализации поля Month Element (Элемент месяца).
  - HTML-тег:

```
<input type="month" name="element-month">
```

  - Вид элемента типа `month` в браузере Opera 12.0.

- Тип `email`.
  - Элемент: `Zend\Form\Element\Email`.
  - Используется для визуализации поля ввода адреса электронной почты.
  - HTML-тег:

```
<input type="email" name="element-email">
```

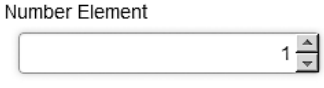
- Тип `url`.
  - Элемент: `Zend\Form\Element\Url`.
  - Используется для визуализации поля ввода URL-адреса.
  - HTML-тег:

```
<input type="url" name="element-url">
```

- Тип `number`.
  - Элемент: `Zend\Form\Element\Number`.
  - Служит для визуализации поля ввода Number Element (Числовой элемент).
  - HTML-тег:

```
<input type="number" name="element-number">
```

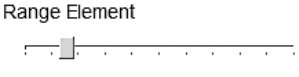
  - Вид элемента типа `number` в браузере Opera 12.0.



- Тип `range`.
  - Элемент: `Zend\Form\Element\Range`.
  - Используется для визуализации поля ввода Range Element (Элемент диапазона) с помощью ползунка.
  - HTML-тег:

```
<input type="range" name="element-range">
```

  - Вид элемента типа `range` в браузере Opera 12.0.

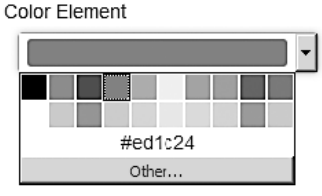


- Тип `color`.
  - Элемент: `Zend\Form\Element\Color`.
  - Используется для визуализации поля ввода Color Element (Элемент цвета) с помощью цветовой палитры.

- HTML-тег:

```
<input type="color" name="element-color">
```

- Вид элемента типа `color` в браузере Opera 12.0.



Время действовать — HTML5-элементы ввода

В этом примере мы создадим тестовую форму для визуализации элементов ввода различных типов, относящихся к спецификации HTML5.

1. Создайте тестовое действие `formAction()` для визуализации элемента формы; его можно поместить в новый контроллер `Html5TestController` (файл module/Users/src/Users/Controller/Html5TestController.php).

2. Добавьте ссылки на `Zend\Form\Form` и `Zend\Form\Element`:

```
use Zend\Form\Element;
use Zend\Form\Form;
```

3. Добавьте в форму различные HTML5-элементы:

```
$form = new Form();

// Элемент Date/Time
$dateTime = new Element\DateTime('element-date-time');
$dateTime->setLabel('Date/Time Element')->setAttributes(array(
  'min' => '2000-01-01T00:00:00Z',
  'max' => '2020-01-01T00:00:00Z',
  'step' => '1',
));
$form->add($dateTime);

// Элемент Date/Time Local
$dateTime = new Element\DateTimeLocal('element-date-time-local');
$dateTime->setLabel('Date/Time Local Element')->setAttributes(array(
  'min' => '2000-01-01T00:00:00Z',
  'max' => '2020-01-01T00:00:00Z',
```

```
  'step' => '1',
));
$form->add($dateTime);

// Элемент Time
$time = new Element\Time('element-time');
$time->setLabel('Time Element');
$form->add($time);

// Элемент Date
$date = new Element\Date('element-date');
$date->setLabel('Date Element')->setAttributes(array(
  'min' => '2000-01-01',
  'max' => '2020-01-01',
  'step' => '1',
));
$form->add($date);

// Элемент Week
$week = new Element\Week('element-week');
$week->setLabel('Week Element');
$form->add($week);

// Элемент Month
$month = new Element\Month('element-month');
$month->setLabel('Month Element');
$form->add($month);

// Элемент Email
$email = new Element\Email('element-email');
$email->setLabel('Email Element');
$form->add($email);

// Элемент URL
$url = new Element\Url('element-url');
$url->setLabel('URL Element');
$form->add($url);

// Элемент Number
$number = new Element\Number('element-number');
$number->setLabel('Number Element');
$form->add($number);

// Элемент Range
$range = new Element\Range('element-range');
$range->setLabel('Range Element');
$form->add($range);

// Элемент Color
$color = new Element\Color('element-color');
```

```
$color->setLabel('Color Element');
$form->add($color);
```

Что сейчас произошло?

Мы создали простую форму, которая целиком состоит из HTML5-элементов, поддерживаемых в Zend Framework 2.0. Эта форма в ее текущем виде может быть визуализирована путем создания требуемого представления. Наша следующая задача — создать такое представление с помощью помощников, позволяющих визуализировать все добавленные в форму HTML5-элементы.

Помощники представлений для визуализации HTML5-элементов

Zend Framework предоставляет помощников представлений для визуализации всех элементов форм, описанных в предыдущем разделе. Для динамической визуализации произвольных элементов ввода на основе их типов можно воспользоваться помощником `formElement()`, однако такая практика не рекомендуется.

Следующая таблица содержит список стандартных помощников для HTML5-элементов ввода.

Тип элемента	Помощник	Функция помощника
datetime	Zend\Form\View\Helper\FormDateTime	formDateTime()
datetimelocal	Zend\Form\View\Helper\FormDateTimeLocal	formDateTimeLocal()
time	Zend\Form\View\Helper\FormTime	formTime()
date	Zend\Form\View\Helper\FormDate	formDate()
week	Zend\Form\View\Helper\FormWeek	formWeek()
month	Zend\Form\View\Helper\FormMonth	formMonth()
email	Zend\Form\View\Helper\FormEmail	formEmail()
url	Zend\Form\View\Helper\FormUrl	formUrl()
number	Zend\Form\View\Helper\FormNumber	formNumber()
range	Zend\Form\View\Helper\FormRange	formRange()
color	Zend\Form\View\Helper\FormColor	formColor()

Помимо стандартных помощников представлений, Zend Framework предоставляет помощников для типов `tel` и `search` — эти типы являются расширениями текстового ввода, однако некоторые браузеры (особенно мобильные) поддерживают стилизованный ввод для обоих этих элементов. В следующей таблице перечислены дополнительные помощники для HTML5-элементов ввода.

Тип элемента	Помощник	Функция помощника
tel	Zend\Form\View\Helper\FormTel	formTel()
search	Zend\Form\View\Helper\FormSearch	formSearch()

Время действовать — помощники представлений для визуализации HTML5-элементов

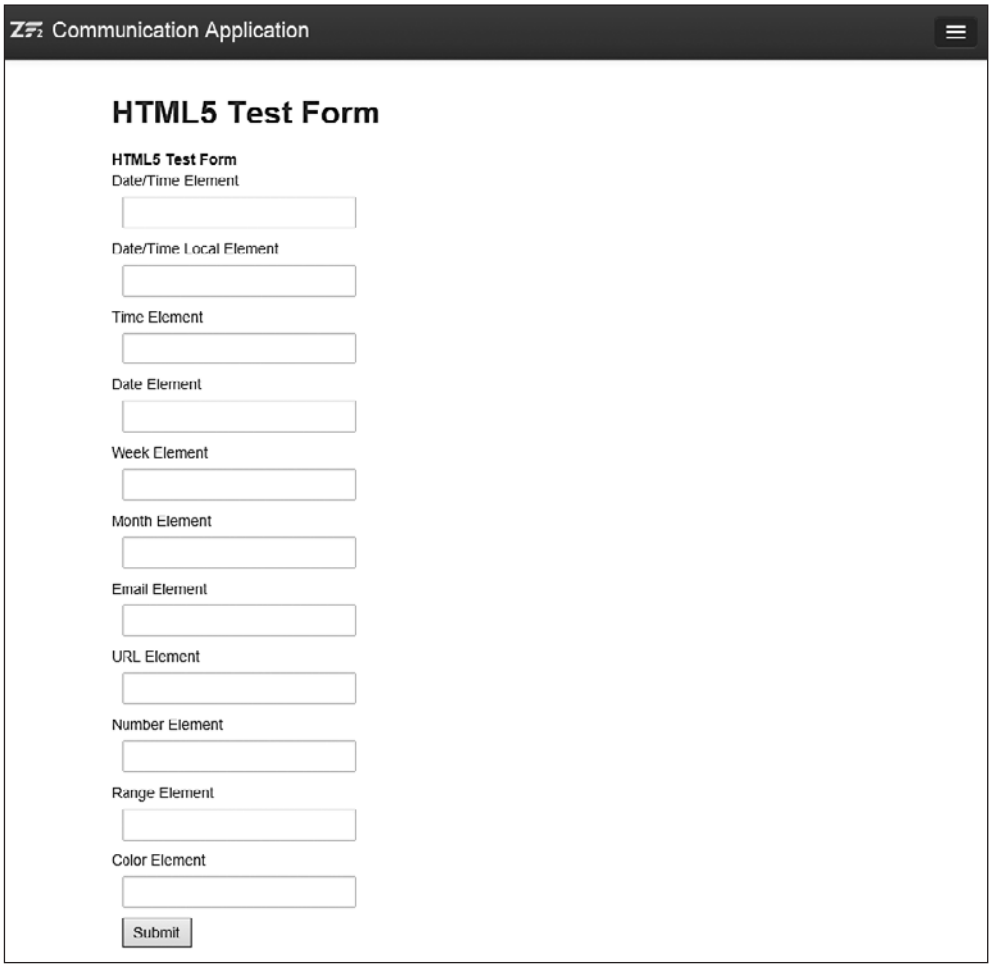
В этом упражнении мы визуализируем все элементы формы, которую создали в предыдущем упражнении. Для этого мы воспользуется доступными в Zend Framework помощниками для визуализации HTML5-элементов.

1. Создайте простое представление для визуализации формы.

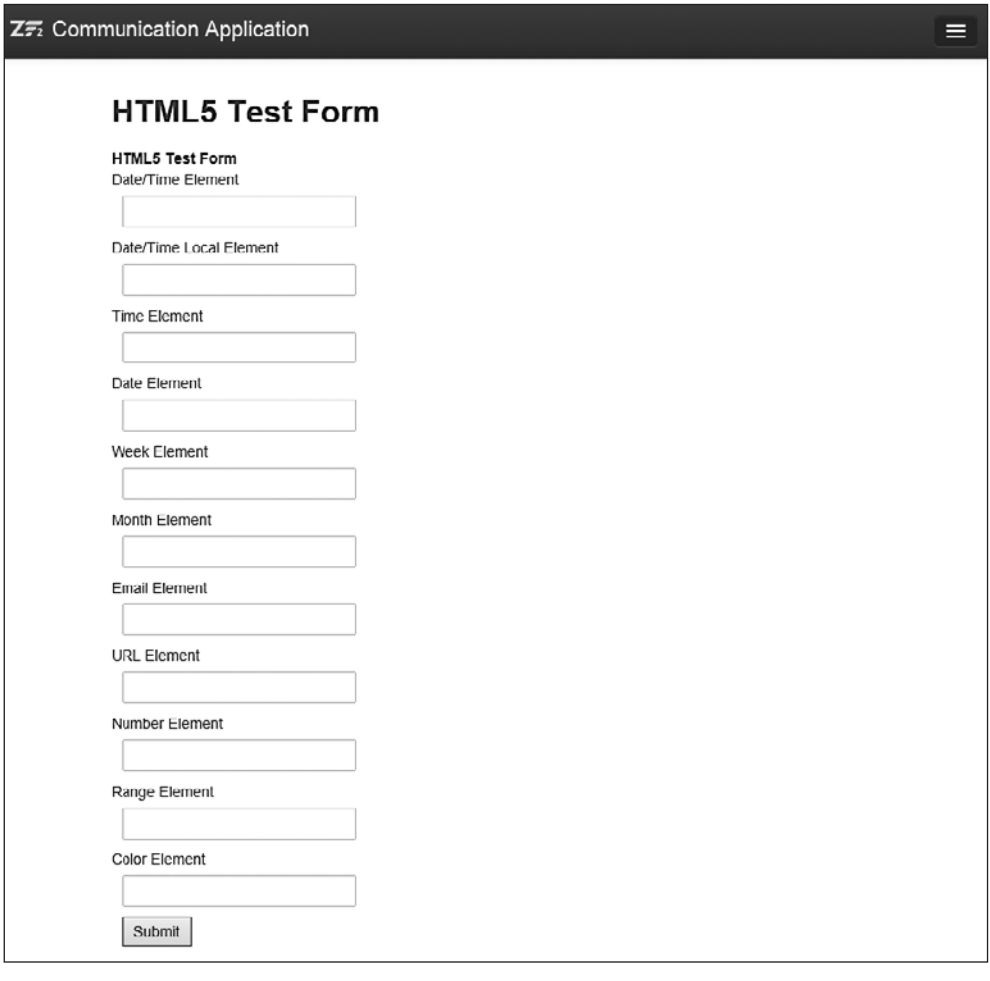
2. Воспользуйтесь помощниками представлений для визуализации различных элементов формы, как показано в следующем коде:

```
$this->formDateTime($form->get('element-date-time'));
$this->formDateTimeLocal($form->get('element-date-time-local'));
$this->formTime($form->get('element-time'));
$this->formDate($form->get('element-date'));
$this->formWeek($form->get('element-week'));
$this->formMonth($form->get('element-month'));
$this->formEmail($form->get('element-email'));
$this->formUrl($form->get('element-url'));
$this->formNumber($form->get('element-number'));
$this->formRange($form->get('element-range'));
$this->formColor($form->get('element-color'));
```

3. Протестируйте форму в браузере, совместимом с HTML5, например в Opera 12. Вы должны увидеть форму, подобную представленной на рисунке.



4. Теперь протестируйте ту же форму в браузере, который не совместим с HTML5, например Internet Explorer 9. Вы должны увидеть форму, подобную представленной на рисунке. Как видите, неподдерживаемые элементы ввода заменяются текстовыми полями.



Что сейчас произошло?

Мы создали первую HTML5-форму, используя в ней элементы форм, доступные в Zend Framework 2. На данный момент наилучшая поддержка HTML5 обеспечивается браузером Opera 12; другие браузеры, например Chrome и Safari, также поддерживают HTML5 на хорошем уровне. Тестируя HTML5-формы, не забудьте удостовериться в том, что делаете это в подходящем браузере, например в Opera 12.

СОВМЕСТИМОСТЬ БРАУЗЕРОВ С HTML5

Различные браузеры по-разному поддерживают спецификации HTML5; Opera и Chrome лучшие с точки зрения совместимости, однако ни один из них не обеспечивает стопроцентную совместимость. С каждой новой версией браузеров функциональные возможности HTML5 получают все большую и большую поддержку. В Интернете имеется много ресурсов, с помощью которых вы можете проверить совместимость вашего браузера со спецификацией HTML5. По адресу http://html5test.com/ находится портал, на котором имеются рейтинги и сравнения браузеров по поддержке спецификации HTML5. Еще один полезный веб-сайт, который позволяет пользователям проверить возможность применения конкретных функциональных возможностей спецификации HTML5 в интересующем их браузере, находится по адресу http://caniuse.com/.

Самостоятельная работа

Перед тем как приступать к изучению нетривиальных HTML5-атрибутов, выполните несложное задание. Теперь, когда вы создали форму со всеми стандартными HTML5-элементами, попробуйте расширить ее, используя помощников представлений для визуализации элементов ввода `tel` и `search`.

HTML5-атрибуты

Возможно, в начале этой главы вы обратили внимание на то, что мы воспользовались такими новыми атрибутами, как `min`, `max` и `step`. Эти атрибуты, определенные в спецификации HTML5, позволяют разработчикам дополнительно настроить элемент ввода. Некоторые важные атрибуты перечислены в следующем списке:

- `max` — применяется к полям Number, Range и Date, позволяя указать максимальное вводимое значение;
- `min` — применяется к полям Number, Range и Date, позволяя задать минимальное вводимое значение;
- `step` — применяется к полям Number, Range и Date, позволяя задать шаг ввода значений;
- `list` — применяется к различным элементам ввода текста, позволяя разработчикам связать поле со списком данных и тем самым дать конечным пользователям возможность выбрать данные в списке;
- `placeholder` — применяется к различным элементам ввода текста, позволяя разработчикам показывать заменители вместо текста до тех пор, пока элемент не окажется в фокусе;

- `pattern` — применяется к различным элементам ввода текста, позволяя разработчикам проверять вводимые пользователем данные на соответствие регулярному выражению и генерировать ошибку при обнаружении несоответствия;
- `required` — предотвращает отправку пользователем формы с пустыми значениями обязательных полей;
- `multiple` — применяется к элементам файлового ввода, позволяя выгрузить несколько файлов с помощью одного элемента файлового ввода.

Время действовать — выгрузка нескольких файлов средствами HTML5

Чтобы реализовать выгрузку нескольких файлов, вам понадобится установить значение атрибута `multiple` для элемента файлового ввода равным `TRUE`. Если браузер поддерживает множественную выгрузку файлов, то пользователю будет предоставлена возможность выбрать несколько файлов; в противном случае элемент управления позволит выбрать один файл.

1. Создайте новую форму `ImageUpload` и установите атрибут `multiple` элемента `File` в значение `TRUE`:

```
<?php
// Имя файла: module/Users/src/Users/Form/MultiImageUploadForm.php
namespace Users\Form;

use Zend\Form\Form;
use Zend\Form\Element;
use Zend\InputFilter;

class MultiImageUploadForm extends Form
{
  public function __construct($name = null, $options = array())
  {
    parent::__construct($name, $options);
    $this->addElements();
    $this->addInputFilter();
  }

  public function addElements()
  {
    $imageupload = new Element\File('imageupload');
    $imageupload->setLabel('Image Upload')
       ->setAttribute('id', 'imageupload')
       ->setAttribute('multiple', true);
    // Разрешение множественной выгрузки файлов
    $this->add($imageupload);
```

```
    $submit = new Element\Submit('submit');
    $submit->setValue('Upload Now');
    $this->add($submit);
  }

  public function addInputFilter()
  {
    $inputFilter = new InputFilter\InputFilter();
    // Файловый ввод
    $fileInput = new InputFilter\FileInput('imageupload');
    $fileInput->setRequired(true);
    $fileInput->getFilterChain()->attachByName('filerenameupload',
      array(
        'target' => './data/images/temp.jpg',
        'randomize' => true
    )
  );
  $inputFilter->add($fileInput);
  $this->setInputFilter($inputFilter);
  }
}
```

ФИЛЬТР ZEND\FILTER\FILE\RENAMEUPLOAD

Фильтр RenameUpload используется для переименования и переноса выгруженного файла в новое место, указанное в параметре target. Чтобы получить дополнительную информацию, обратитесь к документации фреймворка по адресу http://framework.zend.com/manual/2.2/en/modules/zend.filter.file.rename-upload.html.

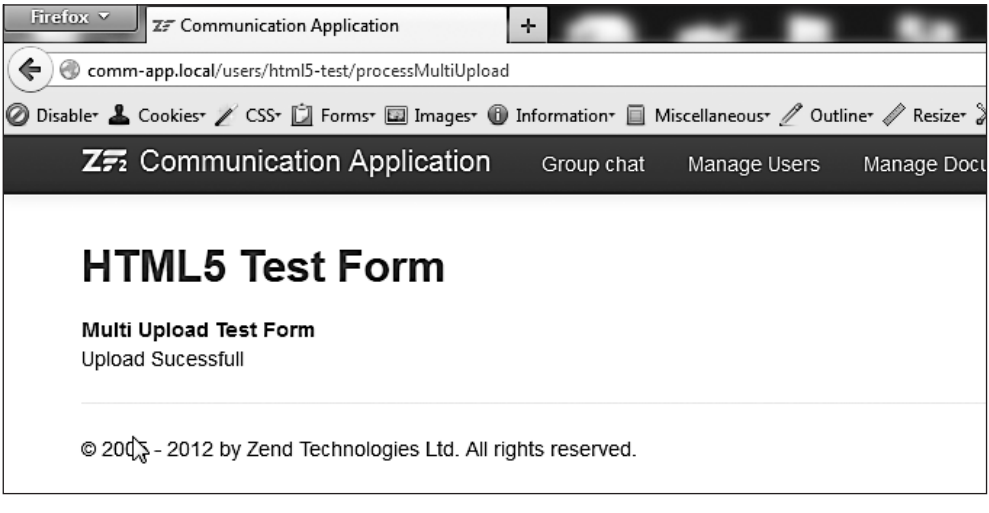
2. Создайте действие для выгрузки файлов и перенаправления пользователя на страницу подтверждения выгрузки:

```
public function multiUploadAction()
{
  // Подготовка формы
  $form = $this->getServiceLocator()->get('MultiImageUploadForm');
  $request = $this->getRequest();
  if ($request->isPost()) {
    $post = array_merge_recursive(
      $request->getPost()->toArray(),
      $request->getFiles()->toArray()
    );
    $form->setData($post);
    if ($form->isValid()) {
      $data = $form->getData();
      // Форма корректна, сохраняем ее!
      return $this->redirect()->toRoute(
        'users/html5-test', rray('action' => 'processMultiUpload'));
```

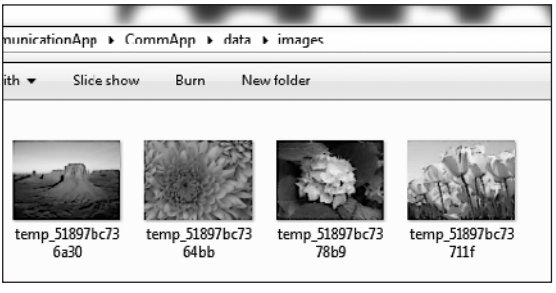
```
        }
    }
    $viewModel = new ViewModel(array('form' => $form));
    return $viewModel;
}
```

3. Теперь протестируйте форму в браузере, который поддерживает множественную выгрузку файлов согласно спецификации HTML5, например в Opera 12. Как видите, файловый интерфейс позволяет выбрать несколько файлов.

4. После завершения выгрузки вы увидите страницу подтверждения.



5. Вы можете проверить, успешно ли выгружены файлы и применены ли фильтры, перейдя в каталог data/images и поискав в нем выгруженные файлы. Вы увидите, что имена всех файлов начинаются с префикса temp и оканчиваются суффиксом вида _<случайное число>:



ФИЛЬТРЫ И МНОЖЕСТВЕННАЯ ВЫГРУЗКА ФАЙЛОВ

При использовании фильтров с множественной выгрузкой файлов одни и те же параметры фильтрации применяются ко всем успешно выгруженным файлам.

Что сейчас произошло?

Мы создали форму множественной выгрузки файлов с использованием HTML5-атрибутов и элементов Zend Framework. Мы также применили фильтр для переименования файлов и понаблюдали за фильтрацией при множественной выгрузке файлов.

Контрольные вопросы

1. Какой из перечисленных типов является новым типом ввода, поддерживаемым спецификацией HTML5?

   1) `text`;

   2) `radio`;

   3) `checkbox`;

   4) `number`.

2. Какие из следующих типов ввода не имеют соответствующего элемента форм, определенного в Zend Framework 2.1?

   1) `tel`;

   2) `date`;

3) `color`;

4) `search`.

Заключение

HTML5 представляет собой развитую и мощную спецификацию HTML, которая пока лишь частично поддерживается большинством браузеров. С появлением на рынке новых версий браузеров поддержка этой спецификации будет существенно расширяться. В следующей главе мы воспользуемся Zend Framework 2 для создания мобильных веб-приложений.

Создание мобильных приложений 10

Одна из важных проблем разработки мобильных приложений заключается в необходимости обеспечивать совместимость мобильного приложения с большим количеством платформ. Такие фреймворки, как PhoneGap и Titanium, позволяют разработчикам создавать кроссплатформенные мобильные приложения, однако у этого подхода есть недостаток — необходимость управлять множеством проектов по разработке мобильных служб и веб-служб для различных платформ. Попытка решить эту проблему в Zend Framework сделана с помощью среды разработки Zend Studio 10, которая основана на фреймворке PhoneGap и поддерживает комплексные мобильные приложения в облачной среде.

Среда разработки Zend Studio 10 существенно упрощает создание мобильных приложений на основе Zend Framework 2 с помощью платформы, известной как **Cloud Connected Mobile Tool**. В этой главе мы изучим основы создания облачных мобильных приложений с использованием Zend Studio. Вот некоторые ключевые темы настоящей главы:

- создание первого **облачного мобильного приложения** с использованием технологии CCM (Cloud Connected Mobile);
- тестирование «родного» приложения;
- реализация простого поискового интерфейса.

Облачные мобильные приложения

Теперь среда разработки Zend Studio включает в себя платформу CCM-инструмент, который позволяет разработчикам создавать «родные» мобильные приложения с помощью облачной платформы. CCM поддерживает разработку облачных веб-

служб на основе механизмов RPC и REST с использованием Zend Framework 2 и Zend Server Gateway.

CCM позволяет также создавать «родные» мобильные приложения путем интеграции с различными комплектами разработки мобильных программ (Android SDK/ADT для операционной системы Android, Xcode для iOS и Windows PhoneSDK для Windows Phone). Это дает разработчикам возможность создавать и тестировать приложения в тех средах и устройствах, для которых они предназначены.

Кроме того, CCM-инструмент включает в себя простой и несложный в использовании редактор графических пользовательских интерфейсов, позволяющий разработчикам без больших усилий создавать развитые пользовательские интерфейсы для своих мобильных приложений.

Среда разработки Zend Studio 10

Ваш первый шаг к созданию собственного мобильного приложения — установить среду разработки Zend Studio 10. Zend Studio 10 поддерживает разработку облачных мобильных приложений и позволяет разработчикам развертывать мобильные приложения в облаке.

Программа Zend Studio 10 продается в онлайновом магазине Zend Store; кроме того, существует ее 30-дневная бесплатная пробная версия. Дополнительную информацию можно получить на сайте http://www.zend.com/en/products/studio/.

Среда разработки phpCloud

Zend Developer Cloud — это облачная среда разработки на языке PHP, которая позволяет разработчикам избежать сложностей, связанных с созданием, настройкой и поддержкой среды PHP-разработки при создании и развертывании приложений в облаке.

Эта среда включает в себя установленный фреймворк Zend Framework 2 с широким набором PHP-расширений; разработчики могут пользоваться различными инструментами, такими как Zend Studio, Eclipse PDT, и командно-строковым интерфейсом для создания и развертывания своих приложений в облаке разработчика. Zend Developer Cloud также предоставляет возможности для интеграции разрабатыва-

емого приложения в другие внешние облачные службы, например Amazon и IBM Smart Cloud.

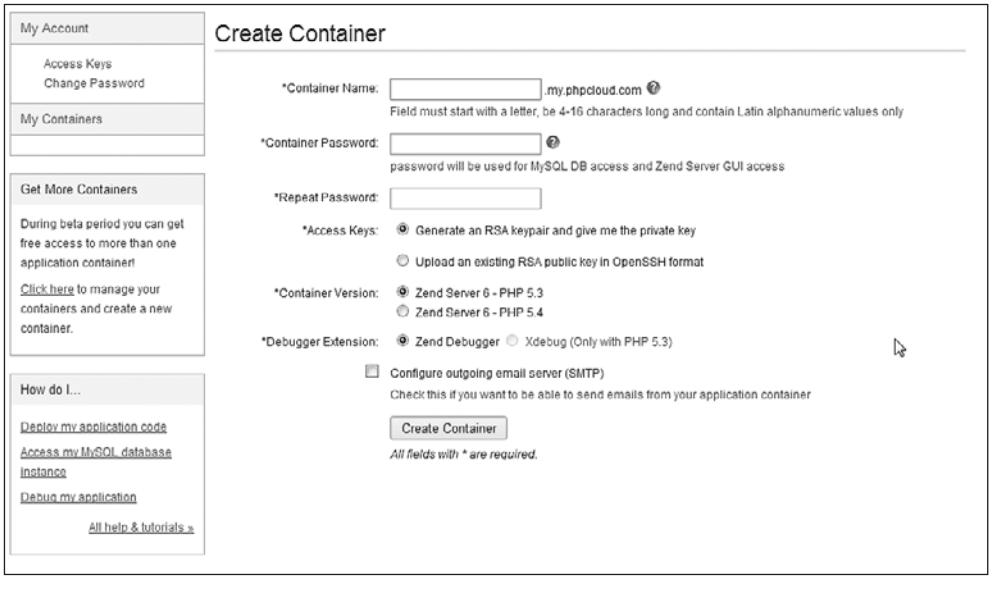
В настоящее время среда Zend Developer Cloud доступна в виде бесплатной бета-версии для разработчиков. Дополнительную информацию о среде Zend Developer Cloud можно получить на веб-сайте http://www.phpcloud.com/.

Время действовать — конфигурирование учетной записи phpCloud

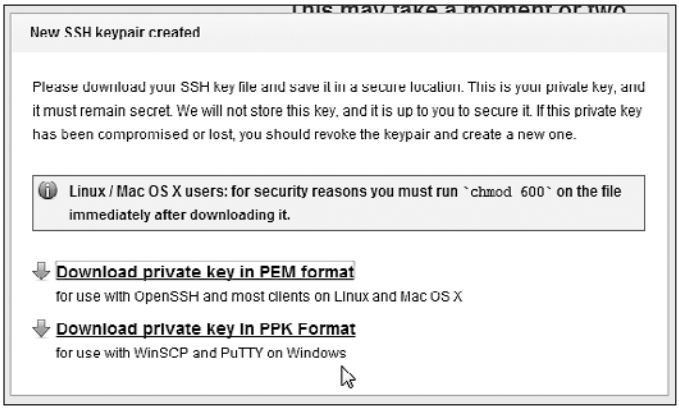
В этом упражнении мы создадим учетную запись в среде phpCloud и настроим облачную среду в Zend Studio 10.

1. Посетите страницу https://my.phpcloud.com/user/login, зарегистрируйте новую учетную запись в среде phpCloud и войдите в нее.

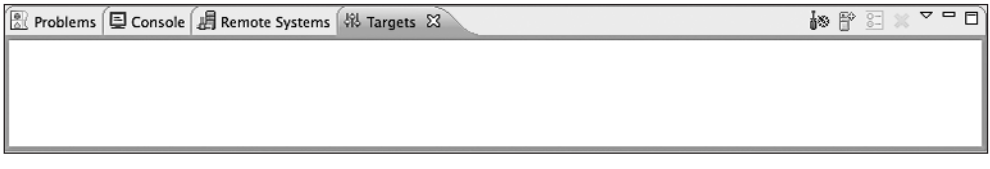
2. После выполнения входа вы получите приглашение создать контейнер. Можете задать имя, которое будет частью URL-адреса контейнера, а также указать на необходимость генерации пары SSH-ключей либо воспользоваться собственными SSH-ключами; в данном случае мы выберем первый вариант. Экран создания контейнера показан на следующем рисунке.



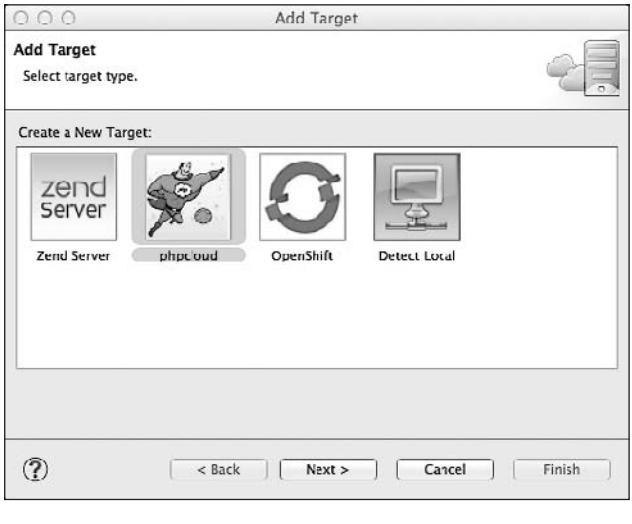
3. Загрузите SSH-ключи; мы будем использовать их для задания цели разработки в среде Zend Studio.



4. В программе Zend Studio перейдите на вкладку Window ▸ Show View ▸ Targets (Окно ▸ Показать представления ▸ Цели).

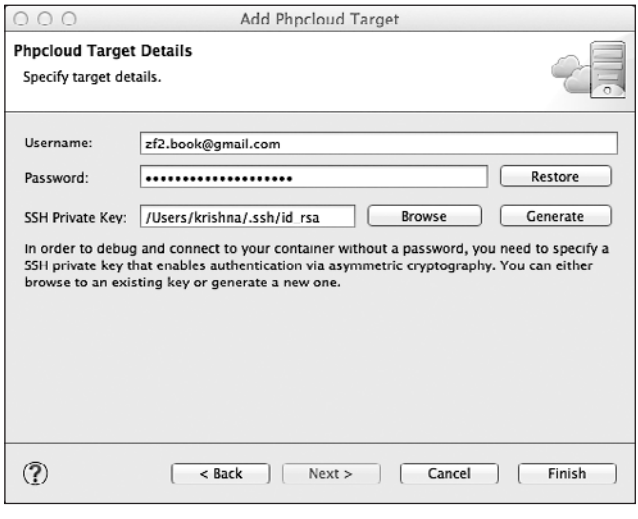


5. Щелкните на значке Add Target (Добавить цель) и выберите вариант phpcloud, как показано на рисунке.

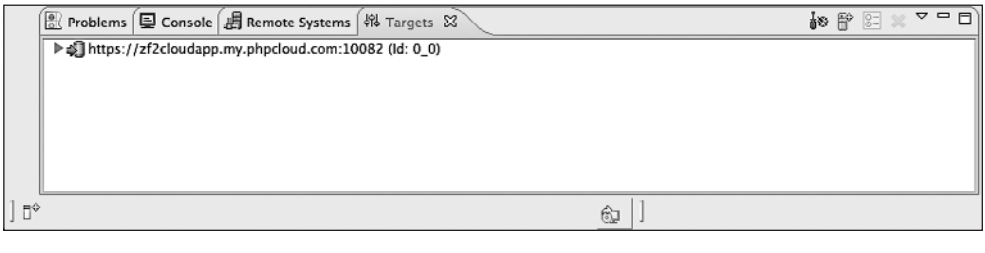


6. На странице phpCloud Target Details (Параметры цели phpCloud) вас попросят указать следующие сведения:

   - Username (Имя пользователя) — имя пользователя в среде Zend Developer Cloud;
   - Password (Пароль) — пароль в среде Zend Developer Cloud;
   - SSH Private Key (Закрытый SSH-ключ) — SSH-ключ, который был сгенерирован на странице создания контейнера phpcloud.



7. После щелчка на кнопке Finish (Готово) вы увидите, что новая цель добавлена в список целей.



Что сейчас произошло?

Мы успешно создали учетную запись в среде phpCloud и настроили облачную среду в Zend Studio 10.

PhoneGap и Zend Studio

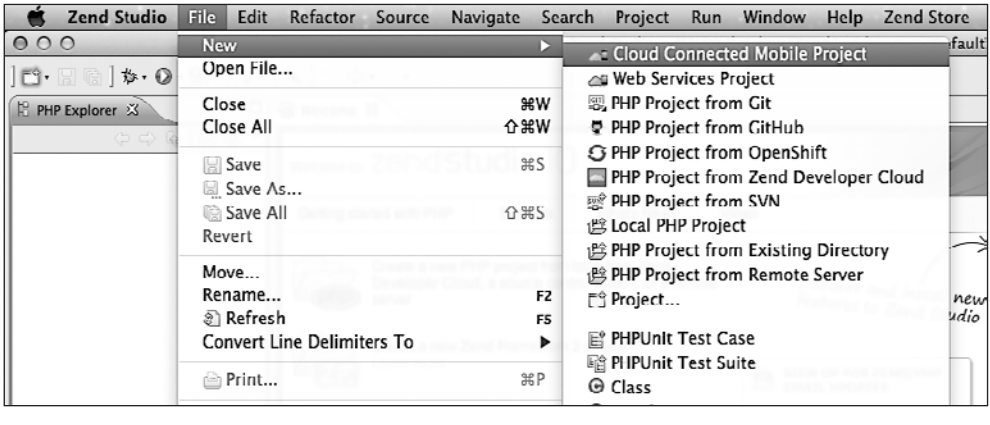
PhoneGap представляет собой фреймворк для разработки мобильных приложений, который позволяет разработчикам создавать мобильные приложения с помощью технологий HTML, CSS и JavaScript. Фреймворк PhoneGap дает возможность превратить эти приложения в «родные» мобильные программы без необходимости переписывать приложения на специальных «родных» языках, таких как Objective-C для iOS.

Теперь фреймворк PhoneGap интегрируется в среду разработки Zend Studio 10, что дает разработчикам возможность легко создавать и тестировать мобильные приложения независимо от внешних библиотек.

Дополнительную информацию о разработке облачных мобильных приложений с помощью Zend Studio 10 можно получить по адресу http://files.zend.com/help/Zend-Studio-10/zend-studio.htm#cloud_connect_mobile.htm.

Время действовать — создание первого облачного мобильного приложения

1. В меню New (Создать) выберите пункт Cloud Connected Mobile Project (Проект облачного мобильного приложения).

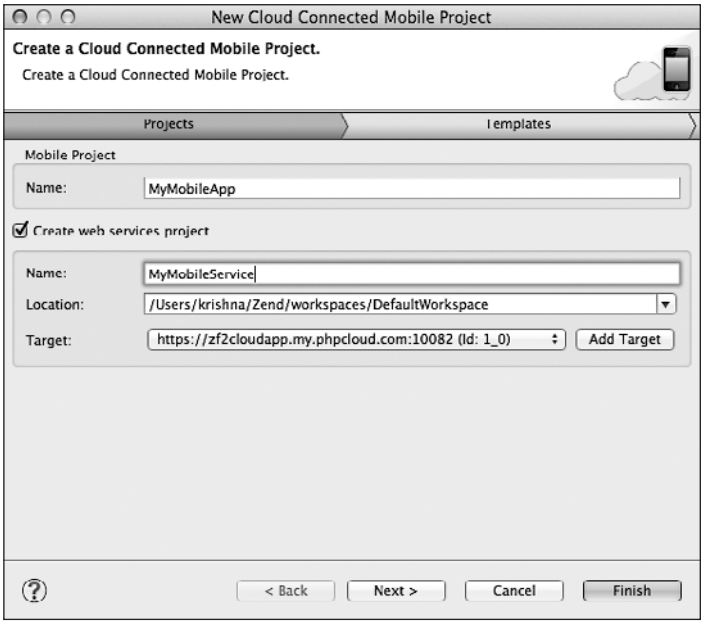


2. Мастер Project (Проект) запросит у вас следующую информацию:

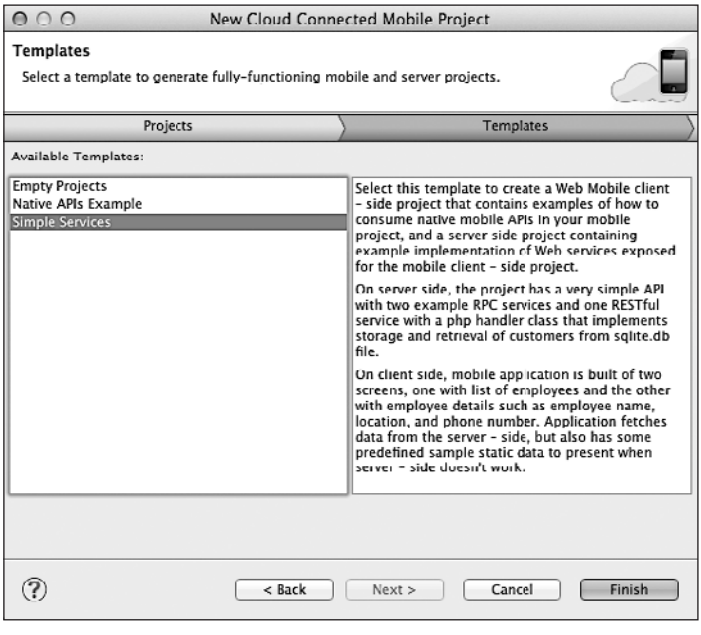
   - Mobile Project Name (Имя мобильного проекта) — имя проекта клиентской части мобильного приложения;
   - Web Services Project Name (Имя проекта веб-служб) — имя проекта веб-служб для мобильного приложения;
   - Web Services Project Deployment Target (Цель развертывания проекта веб-служб) — здесь вы можете выбрать ранее созданную цель phpcloud.

ПРИМЕЧАНИЕ

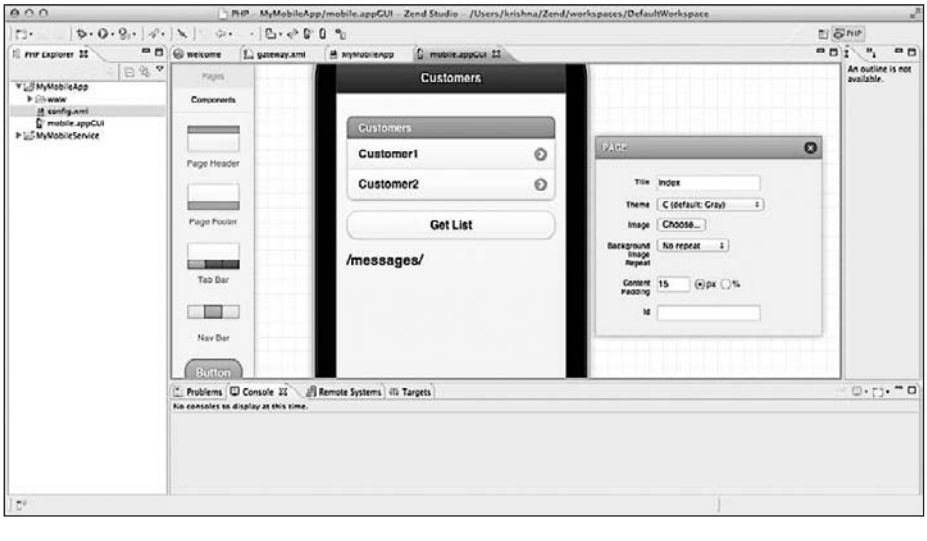
Среда разработки Zend Studio 10 поддерживает различные варианты развертывания приложений; она способна автоматически обнаруживать установку Zend Server или развертывать приложение на одной из целей — локальном или удаленном сервере Zend Server, Zend Server Cloud (phpCloud) или OpenShift Cloud.



3. На странице выбора шаблона выберите шаблон Simple Services (Простые службы), который позволяет создать простой проект с примером клиент-серверной программы.



4. После щелчка на кнопке Finish (Готово) будут созданы проекты мобильного приложения и веб-служб. Дизайнер пользовательского интерфейса в мобильном проекте позволяет легко внести изменения в мобильный интерфейс, как показано на рисунке.



5. Теперь запустите проект из среды разработки Zend Studio; должен запуститься интерфейс эмулятора Zend Framework, как показано на рисунке.

Кнопка Get List (Получить список) позволяет вернуть список клиентов веб-служб путем удаленного вызова процедуры. Если результатом запроса станет не ответ, а ошибка, например Ajax error. Error: Access is denied. Trying static data! (Ошибка AJAX: доступ запрещен. Попытка использовать статические данные!), проверьте значение переменной `gatewayURL` в сценарии MobileApplication/www/js/my.js. Убедитесь, что в этой переменной указан правильный URL-адрес для развертывания, который должен выглядеть следующим образом:

```
var gatewayURL = 'http://zf2cloudapp.my.phpcloud.com/MobileService';
```

Что сейчас произошло?

Мы успешно создали наше первое мобильное приложение, используя проекты облачных мобильных приложений в Zend Framework. В следующих разделах мы узнаем, как расширить эти веб-службы с помощью Zend Framework 2.0, чтобы наделить мобильное приложение дополнительной функциональностью.

«Родные» приложения и веб-приложения

«Родные» мобильные приложения имеют существенные преимущества по сравнению с мобильными веб-приложениями. «Родные» приложения выполняются

в памяти устройства, поэтому им требуется минимум сетевого взаимодействия; как правило, эти приложения загружаются и работают быстрее. Еще одно важное преимущество «родных» мобильных приложений —доступ к параметрам устройства и его «родным» возможностям, таким как камера и акселерометр.

Время действовать — тестирование «родного» приложения

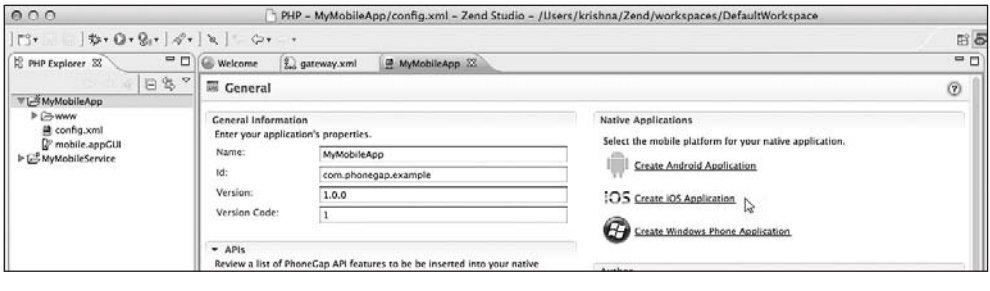
В этом упражнении мы создадим «родное» приложение для iOS с помощью раздела Native Applications (Родные приложения) среды Zend Studio. Перед тем как начать, проверьте, что среда разработки Xcode IDE установлена в вашей Mac-системе.

СОВЕТ

Для Android-приложений потребуется установить Android Development Tool (ADT); это можно сделать непосредственно из Zend Studio.

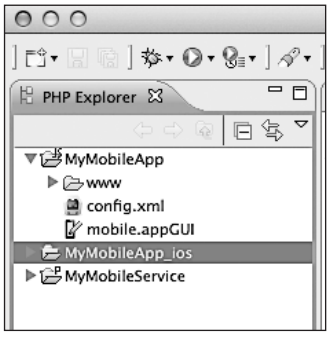
Для приложений Windows Phone потребуется установить комплект Windows Phone SDK.

1. В нашем проекте мобильного приложения выберите команду Create iOS Application (Создать приложение iOS).



2. Вас попросят ввести сведения о проекте; укажите значения Company Name (Название компании) и Bundle ID (Идентификатор приложения). Идентификатор приложения — это уникальное имя, которое используется для идентификации приложения; обычно оно указывается в формате `com.<название-компании>.<имя-приложения>`. При регистрации приложения в Apple Store проверьте, что этот идентификатор совпадает с идентификатором приложения, предоставленным компанией Apple.

3. Как видно из следующего рисунка, новый iOS-проект создан в рабочем пространстве.



СОВЕТ

Среда разработки Zend Studio позволяет создать несколько зависимых проектов мобильных приложений. Если требуется внести какие-либо изменения в код клиентской части, то это можно сделать в родительском мобильном проекте, при этом все зависимые клиентские проекты будут обновлены автоматически.

Дополнительную информацию о создании «родных» приложений можно получить в документации к Zend Studio по адресу http://files.zend.com/help/Zend-Studio-10/zend-studio.htm#creating_native_applications.htm.

4. Если вы запустите проект, то приложение вызовет эмулятор iOS и запустит мобильное приложение, как показано на рисунке.

Что сейчас произошло?

Мы создали новое iOS-приложение, воспользовавшись поддержкой «родных» приложений в Zend Studio; в следующем разделе средствами Zend Framework 2 мы предоставим этому приложению веб-службы.

Самостоятельная работа

Теперь, когда вы создали «родное» iOS-приложение, попробуйте создать его Android-версию с помощью Zend Studio. Для этого вам понадобится установить в Zend Studio модуль Android Development Tool.

Zend Server Gateway

Zend Server Gateway представляет собой легковесный шлюз для веб-служб на базе Zend Framework 2, который позволяет связывать маршруты веб-служб с их различными контроллерами и действиями. Zend Server Gateway отвечает за аутентификацию, валидацию, фильтрацию и маршрутизацию для используе-

мых в CCM-проектах прикладных программных интерфейсов технологий RPC и RESTful.

Конфигурационные данные маршрутизации отражены в файле config/gateway.xml; управлять маршрутами и конфигурациями можно с помощью интерфейса редактирования шлюза, имеющегося в Zend Studio.

Время действовать — создание мобильного поискового интерфейса

В этом упражнении мы создадим простой интерфейс для поиска существующих клиентских записей по именам клиентов.

1. Создайте функцию поиска в модели `CustomerRepository` (файл MyMobileService\src\MyCompany\Model\CustomerRepository.php):

```
public function getSearch($query)
{
  $where = new \Zend\Db\Sql\Where();
  $where->like('name', "%$query%");
  return $this->customerTable->select($where)->toArray();
}
```

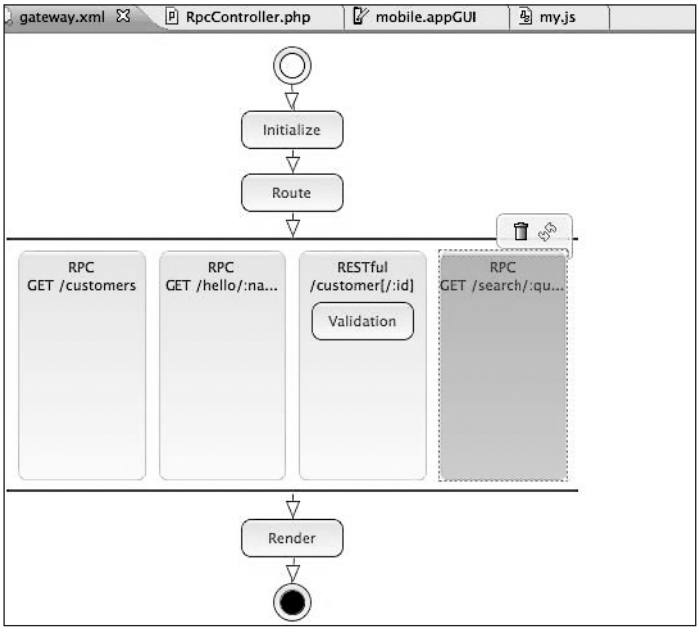
2. Добавьте в контроллер `RpcController` (файл MyMobileService\src\MyCompany\Controller\RpcController.php) новое действие, которое будет обрабатывать запрос к веб-службе:

```
public function getSearchCustomersAction ($query)
{
  $cr = new CustomerRepository();
  return $cr->getSearch($query);
}
```

3. В редакторе шлюза создайте новую RPC-службу и задайте ее параметры:

   - URL (URL-адрес) — `/search`;
   - Method (Метод) — `GET`;
   - Request Parameters(Add) (Параметры запроса(Добавить)): Name (Имя) — `query`; Source (Источник) — `Route`;
   - Handler Method (Метод обработчика) — `MyCompany\Controller\RpcController::getSearchCustomersAction`.

4. Вы можете проверить работу RPC-службы, щелкнув на ней правой кнопкой мыши и выбрав в контекстном меню пункт Test Service (Протестировать службу). Справа вы увидите интерфейс для ввода тестовых данных и проверки реакции службы.



5. В редакторе графических пользовательских интерфейсов для мобильных приложений создайте новую страницу `searchCustomers` и добавьте в нее следующие элементы:

   - Text Box (Текстовое поле) — `custsearchinput`;
   - Button (Кнопка) — `searchbutton`;
   - List View (Список) — `custlistview`.

6. В разделе `binding` кнопки поиска свяжите кнопку с веб-службой `GET /search:query()`. Свяжите текстовое поле `custsearchinput` с параметром маршрута `query` в разделе `data`. Это действие свяжет искомый текст с параметром маршрута `query`. Обратите внимание на то, что параметр маршрута `route` уже связан с действием `getSearchCsutomerAction`.

7. Измените JavaScript-метод `onGetSearchquery` в файле MyMobileApp/www/js/my.js, включив в него обработку RPC-ответа:

```
function onGetSearchquery(response) {
  // Сделать собственный алгоритм обработки ответа сервера
```

```
    customers = response;

    var newCustomers = '';
    $.each(customers, function(index, item) {
      newCustomers += '<li data-theme="">'
        + '<a href="#page2?empId=' + index + '" data-transition="none">'
        + item.name + '</a>' + '</li>';
    });

    $('#custlistview li[role!=heading]').remove();
    $('#custlistview').append(newCustomers).listview('refresh');
  }
```

8. Убедитесь, что вы создали ссылку на страницу Search со страницы index с помощью кнопки.

9. Запустите проект в «родном» режиме; вы сможете увидеть страницу поиска.

Что сейчас произошло?

Мы создали новые веб-службы для существующего облачного мобильного приложения и протестировали приложение в «родном» эмуляторе. Работая в среде Zend Studio 10, вы видите, как просто с ее помощью создавать мобильные приложения, которые поддерживаются веб-службами в облаке.

Контрольные вопросы

1. Для каких из перечисленных платформ можно разрабатывать «родные» мобильные приложения в среде Zend Studio 10?

   1) Android;

   2) Firefox OS;

   3) MeeGo;

   4) Brew.

2. С помощью каких из перечисленных веб-служб невозможно создавать облачные мобильные приложения в Zend Server Gateway?

   1) RPC;

   2) SOAP;

   3) REST.

Заключение

Облачные мобильные приложения — это значительный шаг, сделанный в Zend Framework к тому, чтобы дать PHP-разработчикам возможность создавать и поддерживать мобильные приложения с использованием облачной платформы. Благодаря технологии CCM, Zend Framework предлагает исключительно мощную, но в то же время простую в использовании платформу для создания таких приложений.

Закончив работу над этой главой, вы подошли к концу книги. В процессе чтения вы много узнали о различных вариантах применения Zend Framework и выполнили большое количество упражнений. Эта книга показала вам те «кубики», из которых строятся приложения в Zend Framework 2; для дальнейшего изучения Zend Framework есть множество тем, которые очень подробно изложены в документации http://framework.zend.com/manual/2.2/en/index.html.

Спасибо за то, что вы прочли эту книгу. Пожалуйста, присылайте отзывы о впечатлениях, которые вы получили при чтении.

Приложение. Ответы на контрольные вопросы

Глава 1, «Начало работы с Zend Framework»

1. Ответ 3.
2. Ответ 4.

Глава 2, «Создание первого приложения с помощью Zend Framework»

1. Ответ 2.
2. Ответ 4.

Глава 3, «Создание коммуникационного приложения»

1. Ответ 2.
2. Ответ 1.

Глава 4, «Управление данными и совместное использование документов»

1. Ответ 4.
2. Ответ 3.

Глава 5, «Чат и электронная почта»

1. Ответы 1 и 2.
2. Ответы 2 и 4.

Глава 6, «Совместный доступ к мультимедиа»

1. Ответ 4.
2. Ответ 4.

Глава 7, «Поиск с помощью библиотеки Lucene»

1. Ответ 1.
2. Ответ 4.

Глава 8, «Создание простого магазина»

1. Ответ 2.
2. Ответ 1.

Глава 9, «Поддержка HTML5»

1. Ответ 4.
2. Ответы 1 и 4.

Глава 10, «Создание мобильных приложений»

1. Ответ 1.
2. Ответ 2.

К. Шасанкар

Zend Framework 2.0 разработка веб-приложений

Перевел с английского А. Кузнецов

Заведующий редакцией	*А. Кривцов*
Ведущий редактор	*Ю. Сергиенко*
Литературный редактор	*А. Жданов*
Художник	*Л. Адуевская*
Корректор	*В. Ганчурина*
Верстка	*Л. Егорова*

ООО «Питер Пресс», 192102, Санкт-Петербург, ул. Андреевская (д. Волкова), д. 3, литер А, пом. 7Н.

Налоговая льгота — общероссийский классификатор продукции ОК 005-93, том 2; 95 3005 — литература учебная.

Подписано в печать 30.09.13. Формат 70×100/16. Усл. п. л. 16,770. Тираж 1500. Заказ 0000.

Отпечатано в полном соответствии с качеством предоставленных издательством материалов
в ГППО «Псковская областная типография». 180004, Псков, ул. Ротная, 34.

КНИГА-ПОЧТОЙ

ЗАКАЗАТЬ КНИГИ ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР» МОЖНО ЛЮБЫМ УДОБНЫМ ДЛЯ ВАС СПОСОБОМ:

- на нашем сайте: **www.piter.com**
- по электронной почте: **postbook@piter.com**
- по телефону: **(812) 703-73-74**
- по почте: **197198, Санкт-Петербург, а/я 127, ООО «Питер Мейл»**
- по ICQ: **413763617**

ВЫ МОЖЕТЕ ВЫБРАТЬ ЛЮБОЙ УДОБНЫЙ ДЛЯ ВАС СПОСОБ ОПЛАТЫ:

- Наложенным платежом с оплатой при получении в ближайшем почтовом отделении.
- С помощью банковской карты. Во время заказа Вы будете перенаправлены на защищенный сервер нашего оператора, где сможете ввести свои данные для оплаты.
- Электронными деньгами. Мы принимаем к оплате все виды электронных денег: от традиционных Яндекс.Деньги и Web-money до USD E-Gold, MoneyMail, INOCard, RBK Money (RuPay), USD Bets, Mobile Wallet и др.
- В любом банке, распечатав квитанцию, которая формируется автоматически после совершения Вами заказа.

Все посылки отправляются через «Почту России». Отработанная система позволяет нам организовывать доставку Ваших покупок максимально быстро. Дату отправления Вашей покупки и предполагаемую дату доставки Вам сообщат по e-mail.

ПРИ ОФОРМЛЕНИИ ЗАКАЗА УКАЖИТЕ:

- фамилию, имя, отчество, телефон, факс, e-mail;
- почтовый индекс, регион, район, населенный пункт, улицу, дом, корпус, квартиру;
- название книги, автора, количество заказываемых экземпляров.

ИЗДАТЕЛЬСКИЙ ДОМ ПИТЕР®
WWW.PITER.COM

ВАМ НРАВЯТСЯ НАШИ КНИГИ? ЗАРАБАТЫВАЙТЕ ВМЕСТЕ С НАМИ!

У Вас есть свой сайт?
Вы ведете блог?
Регулярно общаетесь на форумах? Интересуетесь литературой, любите рекомендовать хорошие книги и хотели бы стать нашим партнером?
ЭТО ВПОЛНЕ РЕАЛЬНО!

СТАНЬТЕ УЧАСТНИКОМ ПАРТНЕРСКОЙ ПРОГРАММЫ ИЗДАТЕЛЬСТВА «ПИТЕР»!

- *Зарегистрируйтесь на нашем сайте в качестве партнера по адресу* ***www.piter.com/ePartners***
- *Получите свой персональный уникальный номер партнера*
- *Выбирайте книги на сайте* ***www.piter.com,*** *размещайте информацию о них на своем сайте, в блоге или на форуме и добавляйте в текст ссылки на эти книги (на сайт* ***www.piter.com****)*

ВНИМАНИЕ! *В каждую ссылку необходимо добавить свой персональный уникальный номер партнера.*

С этого момента получайте 10% от стоимости каждой покупки, которую совершит клиент, придя в интернет-магазин «Питер» по ссылке с Вашим партнерским номером. А если покупатель приобрел не только эту книгу, но и другие издания, Вы получаете дополнительно по 5% от стоимости каждой книги.

Деньги с виртуального счета Вы можете потратить на покупку книг в интернет-магазине издательства «Питер», а также, если сумма будет больше 500 рублей, перевести их на кошелек в системе Яндекс.Деньги или Web.Money.

Пример партнерской ссылки:
http://www.piter.com/book.phtml?978538800282 – обычная ссылка
http://www.piter.com/book.phtml?97853880028**2&refer=0000** – партнерская ссылка, где 0000 – это ваш уникальный партнерский номер

Подробно о Партнерской программе ИД «Питер» читайте на сайте WWW.PITER.COM

ИЗДАТЕЛЬСКИЙ ДОМ ПИТЕР®
WWW.PITER.COM

**ПРЕДСТАВИТЕЛЬСТВА ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР»
предлагают профессиональную и популярную литературу по различным направлениям: история и публицистика, экономика и финансы, менеджмент и маркетинг, компьютерные технологии, медицина и психология.**

РОССИЯ

Санкт-Петербург: м. «Выборгская», Б. Сампсониевский пр., д. 29а
тел./факс: (812) 703-73-73, 703-73-72; e-mail: sales@piter.com

Москва: м. «Электрозаводская», Семеновская наб., д. 2/1, стр. 1
тел./факс: (495) 234-38-15; e-mail: sales@msk.piter.com

Воронеж: тел.: 8 951 861-72-70; e-mail: voronej@piter.com

Екатеринбург: ул. Бебеля, д. 11а
тел./факс: (343) 378-98-41, 378-98-42; e-mail: office@ekat.piter.com

Нижний Новгород: тел.: 8 960 187-85-50; e-mail: nnovgorod@piter.com

Новосибирск: Комбинатский пер., д. 3
тел./факс: (383) 279-73-92; e-mail: sib@nsk.piter.com

Ростов-на-Дону: ул. Ульяновская, д. 26
тел./факс: (863) 269-91-22, 269-91-30; e-mail: piter-ug@rostov.piter.com

Самара: ул. Молодогвардейская, д. 33а, офис 223
тел./факс: (846) 277-89-79, 229-68-09; e-mail: samara@piter.com

УКРАИНА

Киев: Московский пр., д. 6, корп. 1, офис 33
тел./факс: (044) 490-35-69, 490-35-68; e-mail: office@kiev.piter.com

Харьков: ул. Суздальские ряды, д. 12, офис 10
тел./факс: (057) 7584145, +38 067 545-55-64; e-mail: piter@kharkov.piter.com

БЕЛАРУСЬ

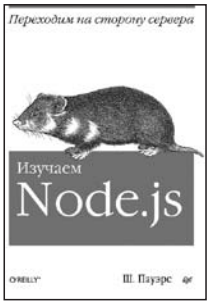
Минск: ул. Розы Люксембург, д. 163
тел./факс: (517) 208-80-01, 208-81-25; e-mail: minsk@piter.com

Издательский дом «Питер» приглашает к сотрудничеству зарубежных торговых партнеров или посредников, имеющих выход на зарубежный рынок
тел./факс: (812) 703-73-73; e-mail: spb@piter.com

Издательский дом «Питер» приглашает к сотрудничеству авторов
тел./факс издательства: (812) 703-73-72, (495) 974-34-50

Заказ книг для вузов и библиотек
тел./факс: (812) 703-73-73, доб. 6250; e-mail: uchebnik@piter.com

Заказ книг по почте: на сайте www.piter.com; по тел.: (812) 703-73-74, доб. 6225



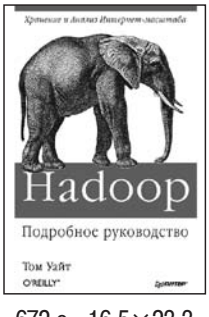
400 с., 16,5×23,3

БЕСТСЕЛЛЕРЫ O'REILLY

Ш. Пауэрс

Изучаем Node.js

Node.js является серверной технологией, которая основана на разработанном компанией Google JavaScript-движке V8. Это прекрасно масштабируемая система, поддерживающая не программные потоки или отдельные процессы, а асинхронный ввод-вывод, управляемый событиями. Она идеально подходит для веб-приложений, которые не выполняют сложных вычислений, но к которым происходят частые обращения. По целям использования Node сходен с фреймворками Twisted на языке Python и EventMachine на Ruby. В отличие от большинства программ JavaScript, этот фреймворк исполняется не в браузере клиента, а на стороне сервера. С помощью этого практического руководства вы сможете быстро овладеть основами Node. Книга понравится всем, кто интересуется новыми технологиями, например, веб-сокетами или платформами создания приложений. Эти темы раскрываются в ходе рассказа о том, как использовать Node в реальных приложениях.



672 с., 16,5×23,3

БЕСТСЕЛЛЕРЫ O'REILLY

Т. Уайт

Hadoop. Подробное руководство

Apache Hadoop – фреймворк с открытым исходным кодом, в котором реализована вычислительная парадигма, известная как MapReduce, позволившая Google построить свою империю. Эта книга покажет вам, как использовать всю мощь Hadoop, чтобы создавать надежные, масштабируемые, распределенные системы и обрабатывать гигантские наборы данных. Программисты найдут здесь методики анализа, администраторы узнают, как установить и запустить кластеры Hadoop. Если вы работаете с большими массивами данных, гигабайтами или петабайтами информации, то Hadoop – это идеальное решение. «Hadoop: Подробное руководство» – книга, в которой досконально и доступно описаны все возможности Apache Hadoop. Издание охватывает последние изменения Hadoop, в том числе материалы по новой исполнительной среде MapReduce, называемой MapReduce 2, которая реализована на базе системы YARN (Yet Another Resource Negotiator) – общей системы управления ресурсами для распределенных приложений.

ИЗДАТЕЛЬСКИЙ ДОМ ПИТЕР®
WWW.PITER.COM