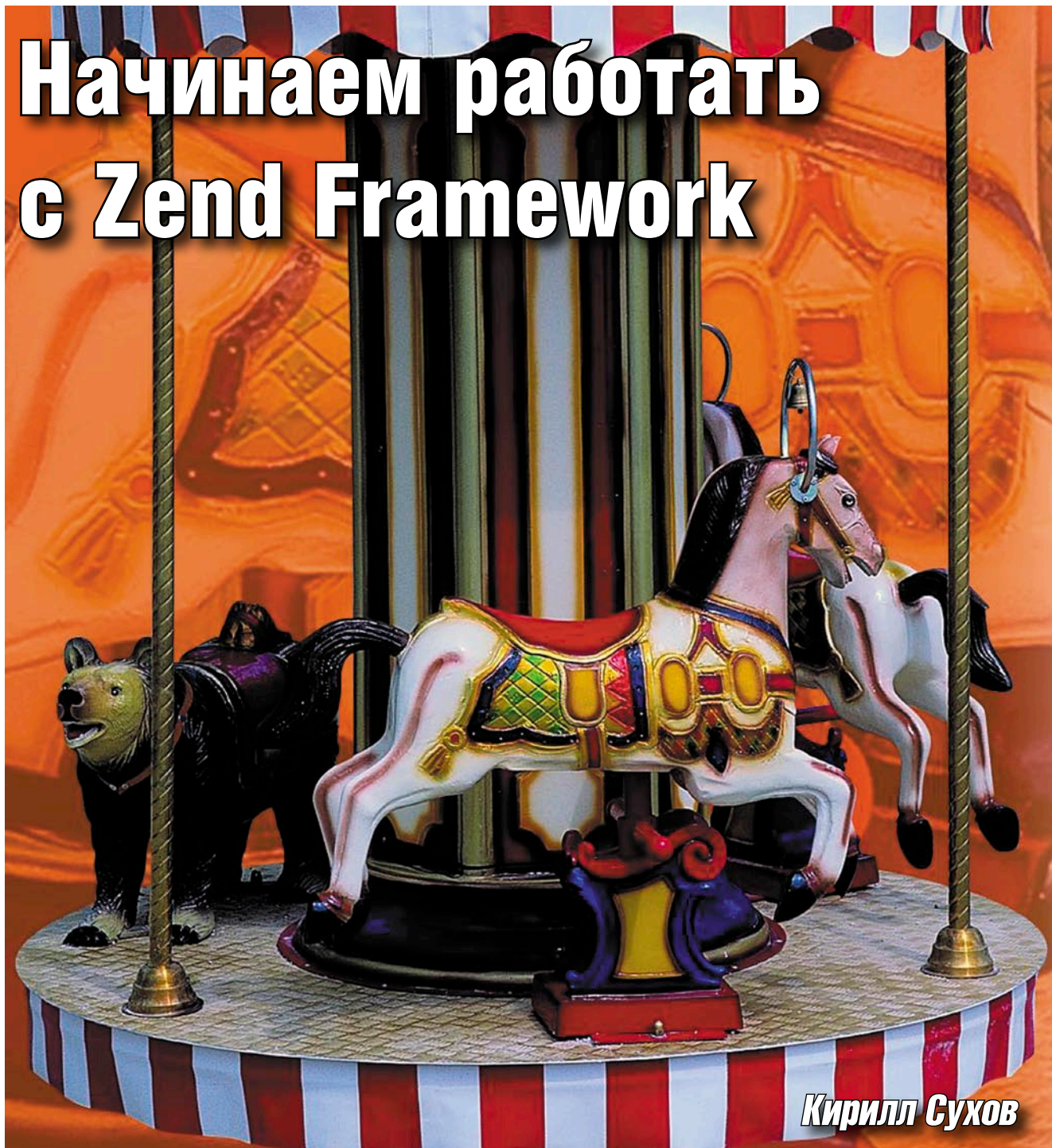


Начинаем работать с Zend Framework



Кирилл Сухов

С появлением пятой версии языка PHP с изменённой объектной моделью фреймворк-среды для разработки веб-приложений на PHP стали появляться и множиться лавинообразно. Zend Technologies Inc., фирма занимающаяся разработкой ядра интерпретатора, тоже не осталась в стороне. Быть может, несколько поздноато (на рынке уже доминировали CakePHP, Symfony, Segun), но был выпущен очень интересный и перспективный продукт. Познакомьтесь – Zend Framework.

В первой части статьи («Выбираем фреймворк-среду для веб-разработки» в №10 за 2007 г.) мы познакомились с двумя фреймворк-системами для веб-разработки

на языке PHP – CakePHP и symfony. Мы даже начали осваивать их основные возможности, но очень быстро пришлось прерваться – объём материала не позволил развернуться как сле-

дует. На применении следующей среды – Zend Framework – я хочу остановиться несколько подробнее.

Интерес к возможности построения полноценной PHP-фреймворк-

Компоненты системы

- **Zend_Acl** – компонент, реализующий функции access control list (ACL), причём реализующий в полном объёме концепции, с группами, ресурсами и т. д.
- **Zend_Auth** – компонент, отвечающий за авторизацию пользователей.
- **Zend_Cache** – инструмен для кэширования данных запроса, обладающий большими возможностями настройки.
- **Zend_Config** – инструмент для гибкой работы с конфигурационными файлами.
- **Zend_Console_Getopt** – компонент для интеграции cli-приложений.
- **Zend_Controller** – компонент, как нетрудно догадаться, является контроллером в MVC-модели приложения. Разумеется, именно он отвечает за исполнение запросов.
- **Zend_Date** – API для различных действий с датой и временем.
- **Zend_Db** – провайдер для доступа к базам данных. Основан на PHP Data Objects (PDO). Предоставляет ряд функций, привычных для работы с СУБД, – от экранирования служебных символов до разбора результата запроса.
- **Zend_Db_Table** – инструмент для работы с таблицами базы данных. Предоставляет объектно-ориентированный интерфейс для таких действий, как выборка добавление, удаление, модификация записей и не только.
- **Zend_Feed** – компонент предназначен для лёгкого создания служб RSS и Atom feeds.
- **Zend_Filter** – компонент предоставляет фильтры для строковых данных, такие как isEmail() или getAlpha(). Незаменим для быстрой валидации отправляемых форм.
- **Zend_Gdata (Zend Google Data Client)** – клиент для сервисов google.com, таких как Spreadsheets, Calendar, Blogger и CodeSearch.
- **Zend_InputFilter** – фильтр, работающий с массивом элементов формы.
- **Zend_HttpClient** – инструмент для построения http-запросов.
- **Zend_Json** – инструмент для работы с данными в нотации Json (JavaScript Object Notation).
- **Zend_Loader** – пожалуй, самый важный класс в модели приложения, основанного на Zend Framework. Его работа – динамическая загрузка файлов классов и ресурсов.
- **Zend_Locale** – компонент предназначен для локализации приложения. Предоставляет доступ к репозиторию CLDR (Common Locale Data Repository).
- **Zend_Log** – компонент для ведения различных видов логов (консольные, бинарные, с разным уровнем оповещения).
- **Zend_Mail** – инструмент для работы с почтой. В том числе с различными MIME-форматами и протоколами.
- **Zend_Mime** – а это в дополнение предыдущему – готовый конвертор MIME-форматов.
- **Zend_Measure** – компонент для работы с различными единицами измерения. Возможно, кому-то жизненно необходимо вычитать из метров ярды.
- **Zend_Memory** – инструмент, предназначенный для управления блоками памяти, выделяемыми приложению.
- **Zend_Pdf** – компонент для работы с документами PDF-формата. Любой работы. Его возможности тянут на отдельную статью. Большую.
- **Zend_Registry** – исключительно полезный компонент, предназначенный для хранения (регистрации) объектов с целью их последующего использования, разумная альтернатива использованию глобальных переменных.
- **Zend_Rest_Client** и **Zend_Rest_Server** – клиент и сервер по принципу REST Web Services – веб-сервер, организованный по принципу архитектуры REST (Representational State Transfer).
- **Zend_Search_Lucene** – фактически полноценный поисковый движок, написанный на PHP. Для работы не требует базу данных, храня все данные в файловой системе.
- **Zend_Service_Amazon, Zend_Service_Flickr, и Zend_Service_Yahoo** – компоненты, предоставляющие простой доступ к программным интерфейсам веб-серверов. Каких – понятно из их названий.
- **Zend_Session** – как следует из названия, инструмент для работы с сессиями. Предоставляет объектно-ориентированный интерфейс для доступа к данным сессии, а также имеет встроенные средства защиты этих данных как от вмешательства извне, так и от конфликтов в именовании сессионных переменных.
- **Zend_Translate** – компонент для перевода контента. Да, теперь и эта головная боль на совести Framework.
- **Zend_Validate** – название не обманывает, это компонент для валидации данных.
- **Zend_Uri** – предназначен для работы с другими компонентами, позволяет создавать, управлять и валидировать обобщённый идентификатор ресурса (Uniform Resource Identifiers, URIs).
- **Zend_View** – компонент для создания представления в MVC-модели.
- **Zend_XmlRpc** – инструмент для создания XML-RPC-клиента. Да, да, вот так скромно и со вкусом.

среды превысил критическую массу, после того как летом 2005 года вышла пятая версия языка. Это вполне понятно, ведь именно с PHP 5 появилась полноценная ООП-модель, и объекты в PHP стали настоящими объектами, а не красиво упакованными ассоциативными массивами.

Уже после появления многих известных сегодня систем стало известно, что Zend Technologies Inc. – компания, разрабатывающая ядро PHP, готовит свой «фирменный» Zend Framework. Интерес к этому продукту со стороны общественности был очень велик, но то, что фирма предоставила в начале 2006 года, большого энтузиазма не вызвало. В ранних версиях среда казалась слабо упорядоченным набором

классов, которые по заявленному функционалу вроде бы замахивались на многое, но никак не могли конкурировать со средствами, уже существующими на рынке. Возможно, что Zend Technologies Inc. сознательно выложили свою разработку на очень ранней версии, рассчитывая на помощь и оценку сообщества. По крайней мере, тот продукт, который был выпущен спустя полтора года, вполне претендует на звание лучшего в своей нише. Впрочем, давайте посмотрим сами.

Установка

Системные требования среды касаются именно системы – должен быть в наличии интерпретатор PHP версии не ни-

же 5.1.4 и присутствовать поддержка веб-сервером модуля `mod_rewrite` (последнее не обязательно, но её отсутствие делает использование Zend Framework неудобным).

Прежде всего скачаем дистрибутив Zend Framework с <http://framework.zend.com/download/stable>. Полученный архив распаковываем куда-нибудь во временную папку и пока не обращаем на него внимания. Я сейчас сознательно опускаю тот момент, что можно было бы попросту расположить дистрибутив на территории нашего веб-сервера и начать пользоваться средой немедленно. Не то что мы сознательно не ищем лёгких путей, просто они не всегда бывают правильными. Вместо этого вручную построим дерево папок нашего приложения. Почему вручную? Дело в том, что в отличие от остальных фреймворк-систем ZF не предлагает жёстко заданной, предопределённой структуры приложения. Эту среду вообще можно использовать аморфно как набор классов, но как я уже говорил, мы не из таких.

Документация предлагает воспользоваться стандартной структурой, вроде той, что описана ниже:

```
/zend
/app
  /controllers
  /models
  /views
    /filters
    /helpers
    /scripts
  /library
  /public
    /images
    /scripts
    /styles
```

После этого следует поместить папку `/library` из дистрибутива среды в нашу папку `/library` и прописать путь к ней в директиве `include_patch` конфигурационного файла `php.ini`. Кроме этого, сразу создадим файл `zend/.htaccess`, прописав в нём единственный параметр:

```
RewriteEngine on
RewriteRule .* index.php
```

Таким нехитрым способом мы перенаправляем любые запросы к нашему приложению – скрипту `zend/index.php`. Для директории `/public` мы это правило отменяем, разместив там аналогичный файл с директивой:

```
RewriteEngine off
```

Теперь займёмся файлом `index.php`, задача которого заключается в начальной загрузке компонентов приложения. Вот его примерный код:

```
<?php
include "Zend/Loader.php";
Zend_Loader::loadClass('Zend_Controller_Front');
//установка контроллера
$frontController = Zend_Controller_Front::getInstance();
$frontController->setControllerDirectory(
    './application/controllers');

//запуск приложения
$frontController->dispatch();
?>
```

Пока почти всё понятно. Сначала мы подключаем файл, содержащий класс `Zend_Loader`, статический метод которого,

`loadClass()`, позволяет подключать любой класс фреймворка. В данном случае он вызывает класс `Zend_Controller_Front`, находя файл, его содержащий, по пути, получаемому из названия (то есть `Zend/Controller/Front.php`).

Далее мы получаем экземпляр первичного контроллера, указываем ему расположение других контроллеров приложения, явно разрешаем все исключения (по умолчанию они запрещены) и запускаем приложение. В результате своей работы скрипт должен выдать сообщение об ошибке, что вполне естественно, так как никаких контроллеров мы ещё не создали. Сейчас мы этим обязательно займёмся, но пока давайте познакомимся с базовым набором компонентов Zend Framework (см. врезку «Компоненты системы»).

Реализация архитектуры MVC

Всё это множество разнообразных и полезных компонентов внушает оптимизм, их можно использовать прямо сейчас, не читая дальше, лишь сверяясь с мануалом, но фреймворк-среда отличается от простого набора полезных классов наличием системного подхода.

Как уже можно догадаться, Zend Framework реализует MVC (Model-View-Controller). По сравнению с такими средами, как CakePHP или Symfony, подход ZF несколько своеобразен, но гораздо более гибок. Впрочем, давайте рассмотрим его реализацию на примере простого приложения.

Создаём модель

Для начала определимся с задачей и исходя из неё определим модель данных. Я не стал долго думать и для первого приложения выбрал актуальную для меня задачу – приведение в порядок коллекции компакт-дисков с музыкой в MP3-формате. Вообще говоря, эта задача не так уж банальна, но пока организуем информацию самым примитивным образом. Сначала создадим таблицу в базе данных:

```
USE records
CREATE TABLE 'Records'
(
    id int(11) NOT NULL auto increment,
    group varchar(100) NOT NULL,
    notes text NOT NULL,
    PRIMARY KEY (id)
);
```

Насколько я понимаю, пояснений не требуется, за исключением одного – наличие автоинкрементного поля обязательно.

Все операции с записями таблиц (добавление, удаление, вставка, модификация) в среде Zend Framework совершаются посредством работы с классом `Zend_Db_Table`. Тот в свою очередь при инициализации использует класс `Zend_Config`, представляющий собой гибкий и мощный инструмент, предназначенный для работы с конфигурационными файлами. Это могут быть как XML-файлы, так и `.ini`-файлы, использующие простой текстовый формат. В последнем случае наши настройки будут следующими:

```
zend/application/config.ini

[general]
db.adapter = PDO_MYSQL
```

```
db.config.host = localhost
db.config.username = geol
db.config.password = fig_podberesh
db.config.dbname = records
```

Далее добавляем в файл начальной загрузки строки:

```
Zend_Loader::loadClass('Zend_Db');
Zend_Loader::loadClass('Zend_Db_Table');
Zend_Loader::loadClass('Zend_Config_Ini');
Zend_Loader::loadClass('Zend_Registry');

$config = new Zend_Config_Ini('./application/config.ini', 'general');
$registry = Zend_Registry::getInstance();
$registry->set('config', $config);

$db = Zend_Db::factory($config->db->adapter,
    $config->db->config->toArray());
Zend_Db_Table::setDefaultAdapter($db);
```

О методе `getInstance()` немножко ниже, но смысл понятен из названия — он инициализирует экземпляр приложения. Остальной код должен быть понятен.

Контроллеры и действия

Любые действия (в терминах ZF — action) с данными модели (как то просмотр или внесение новых) в Zend Framework объединены в контроллеры, специфичные для приложения. Для обращения к действию `delete` контроллера `records` используется url вида:

```
http://localhost/zend/records/delete
```

При этом контроллеры логически группируют сходные действия. При обращении к контроллеру без указания действия (`http://localhost/zend/records`) выполняется действие по умолчанию, в нашем случае прописанное в файле `records/index.php`.

Все контроллеры находятся в директории `zend/application/controllers/` и представляют собой классы с именем вида: {Имя_контроллера}Controller. Каждый класс хранится в своём файле, имя которого имеет вид: {Имя_контроллера}Controller.php. Аналогично именуются действия — {Имя_действия}Action.php.

Какие действия будет реализовывать наше будущее приложение?

Таковых, как минимум, четыре, а именно: просмотр записей, редактирование записей, внесение новых и удаление существующих. Реализуем заготовки этих действий в классе, загружаемом по умолчанию:

```
zend/application/controllers/IndexController.php:

<?php

class IndexController extends Zend_Controller_Action
{
    function init()
    {
        $this->view->baseUrl = $this->request->getBaseUrl();
        Zend_Loader::loadClass('Records');
    }

    function indexAction()
    {
        print "list";
    }

    function addAction()
    {
```

```
        print "add";
    }

    function editAction()
    {
        ...
    }

    function deleteAction()
    {
        ...
    }
}
```

После этого мы уже получаем работающее приложение, и запрос вида:

```
http://localhost/zend/index/add
```

вызовет к жизни метод `IndexController::addAction`. (Надо добавить, что имя контроллера также выбирается по умолчанию.) Метод `init()`, вызываемый при инициализации объекта, в данном случае задаёт базовый url (не обязательно, но, согласитесь, крайне полезно) и загружает основной класс (тоже сообщение).

Добавляем представление

Все представления в среде Zend Framework представляют собой классы, наследуемые от класса `Zend_View` и расположенные (если не указано иного) в директории `views/`.

Самая простая реализация представления выглядит следующим образом:

```
$view = new Zend_View();
echo $view->render('view.php');
```

Обращаться к нужному представлению из кода контроллера следует так:

```
function indexAction()
{
    $this->view->title = "Records";
    $this->view->caption = "Моя коллекция";
}
```

Скрипты для отображения страниц находятся в директории `zend/application/views/scripts/` и имеют расширение `.phtml` — (`views/scripts/{имя_контроллера}/{имя_действия}.phtml`).

```
zend/application/views/scripts/index/add.phtml:
```

```
<html>
<head>
    <title><?php echo $this->escape($this->title); ?></title>
</head>
<body>
    <h1><?php echo $this->escape($this->caption); ?></h1>
</body>
</html>
```

При рендеринге представления сгенерированный контент веб-страницы сохраняется в объекте `Response`, который служит компоновщиком для сгенерированного контента страницы, а также для HTTP-заголовков и выдачи обработчиков исключительных ситуаций.

Понятие помощников/хелперов (`action helpers`), знакомое нам по другим средам, в Zend Framework также присутствует. Во-первых, это классы вида `Zend_Controller_`

Action_Helper..., позволяющие оперировать функциональностью во время выполнения, во-вторых, более привычные Zend_View_Helper..., берущие на себя рутинную работу при построении представления. Вот пример работы последних (из официальной документации):

```
<form action="action.php" method="post">
  <p><label>Your Email:
  <?php echo $this->formText('email', 'you@example.com', ↵
    array('size' => 32)) ?>
  </label></p>
  <p><label>Your Country:
  <?php echo $this->formSelect('country', 'us', null, ↵
    $this->countries) ?>
  </label></p>
  <p><label>Would you like to opt in?
  <?php echo $this->formCheckbox('opt_in', 'yes', null, ↵
    array('yes', 'no')) ?>
  </label></p>
</form>
```

В данном случае хелперы и служат для генерации элементов формы. Вот что получится на выходе:

```
<form action="action.php" method="post">
  <p><label>Your Email:
    <input type="text" name="email" ↵
      value="you@example.com" size="32" />
  </label></p>
  <p><label>Your Country:
    <select name="country">
      <option value="us" ↵
        selected="selected">United States</option>
      <option value="il">Israel</option>
      <option value="de">Germany</option>
    </select>
  </label></p>
  <p><label>Would you like to opt in?
    <input type="hidden" name="opt_in" value="no" />
    <input type="checkbox" name="opt_in" ↵
      value="yes" checked="checked" />
  </label></p>
</form>
```

Разумеется, построением элементов формы работа хелперов не ограничивается.

Подробнее о хелперах представления можно узнать здесь: <http://framework.zend.com/manual/en/zend.view.helpers.html>.

Таким образом, скрипты для представления нашего приложения будут выглядеть примерно так:

```
zend/application/views/scripts/index/add.phtml:

<html>
```

```
<head>
  <title><?php echo $this->escape($this->title); ?></title>
</head>
<body>
  <h1><?php echo $this->escape($this->caption); ?></h1>
  <form action ="<?php echo $this->baseUrl ?>/index/ ↵
    <?php echo $this->action; ?>" method="post">
    <p><label>Band:
    <?php echo $this->formText('group', '', ↵
      array('size' => 64)) ?>
    </label></p>
    <p><label>Notes:
    <?php echo formTextarea('notes', '', ↵
      array('size' => 256)):?>
    </label></p>
  </form>

</body>
</html>
```

Аналогичным образом строим скрипты для каждого действия. Код для выполнения данного действия. Прописываемый в методах контроллера код достаточно банален, и я оставляю его на совести читателя.

Заключение

Теперь мы можем сказать, что MVC-приложение в общих чертах построено. Я не стал останавливаться на реализации каждого метода и создании представлений и шаблонов для каждого действия – после того как основа приложения построена, это уже дело техники. То же можно сказать и про модификации и расширение функционала.

Всё что нам осталось, это изучить возможности компонентов среды и по мере необходимости включать их в нашу схему. Для детального изучения этих возможностей (а они стоят изучения), кроме официальной документации, рекомендую серию статей «Понимание Zend Framework» от IBM Developerworks.

Удачи! 🍀

1. Zend Framework Tutorial – <http://framework.zend.com/manual/ru>.
2. ru-zend-framework – русскоязычная группа пользователей Zend Framework – <http://groups.google.com/group/ru-zend-framework>.
3. Статья Роба Алена (Rob Allen) Getting Started with the Zend Framework – <http://akrabat.com/zend-framework-tutorial>.
4. Серия статей «Понимание Zend Framework» на сайте IBM – http://www.ibm.com/developerworks/views/opensource/libraryview.jsp?search_by=understanding+the+zend+framework.

