Тема: «Zend Framework. Общие сведения. Реализация паттерна MVC в Zend Framework»

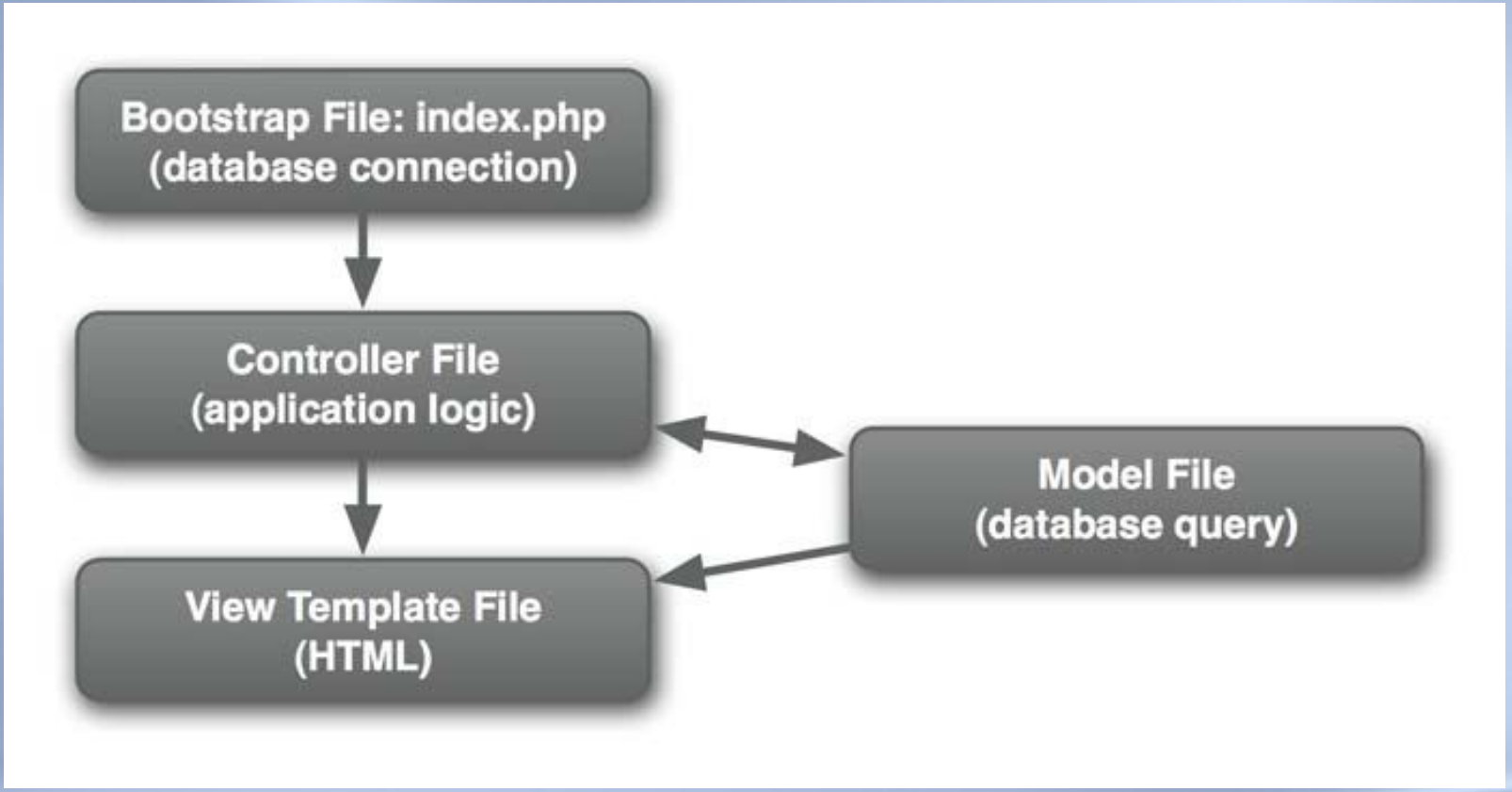
Представил ст. гр. И-52д инженер ЦКТ
Козлов А.А.

Zend Framework

- **Zend Framework** — это высококачественный фреймворк, который является библиотекой с открытым исходным кодом для разработки **веб-приложений** и **веб-сервисов** на базе **PHP**. Он лёгкий в использовании и при этом даёт мощный функционал, что позволяет построить современные, гибкие и защищенные веб-сайты. **Zend Framework** даёт в руки профессионала широкие возможности для построения качественных веб-приложений.
- Преимущества **Zend Framework** :
  - Zend Framework расширяет язык php сохраняя его дух, его главный критерий простота, использованы лучшие приёмы объектно — ориентированного программирования, дружественная лицензия, и хорошо протестированный быстро – исполняемый код. Основной упор в Zend Framework сделан на возможность построения хорошо защищённых, надежных и современных веб 2.0 приложений и веб – сервисов и всепоглощающих широко – доступных API – функций от лидирующих в данной сфере команд таких как Google, Amazon, Yahoo!, Flickr.
  - Zend Framework следует последним направлением в сфере веб – приложений, таким как: поддержка Ajax, Search — php редакция Lucene индустриального стандарта поисковых систем, и т.д.

Zend Framework

- Работа сайта, основанного на технологии MVC



Zend Framework

Структура папок в Zend Framework

Название	Значение
zend_test/	Корневая папка сайта(название может быть изменено)
zend_test/public	Могут располагаться: графика, скрипты и CSS файлы.
zend_test/library	Папка, в которой располагается Zend Framework
zend_test/application	В этой папке хранятся контроллеры, формы, представления, модели
zend_test/application/controllers	В этой папке хранятся контроллеры.
zend_test/application/forms	В этой папке хранятся формы.
zend_test/application/models	В этой папке хранятся модели.
zend_test/application/views	В этой папке хранятся представления.

Zend Framework

Организация страниц

Каждая страница приложения – это “действие” (action), действия группируются в контроллеры. Например, для URL формата http://localhost/public/zf-tutorial/news/view, действие – view, контроллер – news. Это позволяет группировать родственные действия. Например, контроллер news может иметь действия list, archived и view.

По умолчанию ZF использует действие index. Таким образом URL http://localhost/zf-tutorial/public/news// вызовет действие index контроллера news. Аналогичным образом контроллер по умолчанию – index, URL http://localhost/zf-tutorial/public/ выполнит действие Index контроллера index.

Zend Framework

Подключение классов

Для подключения классов в Zend Framework выполняется преобразование указанного имени в путь к соответствующему файлу: символы «_» заменяются на «/», а в конце получившейся строки добавляется расширение .php. Таким образом, Zend_Controller_Front преобразуется в Zend/Controller/Front.php. Таким образом нужные нам для работы данные будут находиться в файле Front.php.

Zend Framework

Формы в Zend Framework

Объекты форм создаются через простое инстанцирование Zend_Form:

```
$form = new Zend_Form;
```

Для того, чтобы указать в форме метод ля отправки данных(method) и action нужно использовать аксессоры setAction() и setMethod():

```
$form->setAction('/resource/process')
        ->setMethod('post');
```

Есть два способа добавления элементов в форму - вы можете инстанцировать нужные элементы и передавать их объекту формы, или передавать только тип элемента, в этом случае Zend_Form инстанцирует соответствующий объект за вас.

```
// Инстанцирование элемента и его передача объекту формы:
$form->addElement(new Zend_Form_Element_Text('username'));
// Передача типа элемента объекту формы
$form->addElement('text', 'username');
```

Zend Framework

Формы в Zend Framework

По умолчанию элементы не имеют никаких валидаторов или фильтров. Поэтому при создании элементов формы придется создавать валидаторы, и, возможно, фильтры. Делать это можно (a) до передачи элементов в форму, (b) через опции конфигурирования, которые передаются при создании элемента через Zend_Form, или (c) путем извлечения элементов формы из объекта формы и их конфигурирования.

Пример готового элемента формы:

```
$this->addElement('text', 'header', array(
        'label'      => 'Name:',
        'required'   => true,
        'validators' => array(
            array('validator' => 'StringLength', 'options' => array(3, 50))
        )
));
```

Zend Framework

Аутентификация в Zend Framework

Zend_Auth предоставляет API для аутентификации и включает в себя конкретные адаптеры для общих случаев применения. Адаптер Zend_Auth используется для аутентификации посредством определенного сервиса, такого как LDAP, СУРБД или файлового хранилища. Адаптеры могут значительно различаться, но некоторые основные черты характерны для всех. Например, все адаптеры Zend_Auth принимают учетные данные, выполняют запрос к аутентификационному сервису и возвращают результат.

Каждый адаптер Zend_Auth реализует Zend_Auth_Adapter_Interface. Этот интерфейс определяет лишь один метод: authenticate(), который должен быть реализовать для выполнения аутентификационного запроса. Адаптер должен быть настроен до вызова authenticate(), настройка включает в себя установку учетных данных(например, логин и пароль) и определение специфичных значений, таких как настройки подключения к базе данных для адаптера таблиц БД.

Метод authenticate() должен вернуть экземпляр Zend_Auth_Result. Если по какой либо причине выполнение аутентификации невозможно, authenticate() должен бросить исключение, происходящее от Zend_Auth_Adapter_Exception.

Zend Framework

Аутентификация в Zend Framework

Метод authenticate() адаптера Zend_Auth возвращает экземпляр Zend_Auth_Result для представления результата попытки аутентификации. Объект Zend_Auth_Result заполняется адаптером при создании, и следующие четыре метода представляют его базовый набор операций:

- isValid() - возвращает **TRUE** только в случае успешной попытки аутентификации.
- getCode() - возвращает значение одной из констант Zend_Auth_Result для обозначения успешности попытки или типа возникшей ошибки.
- getIdentity() - возвращает идентификатор, полученный в результате аутентификации.
- getMessages() - возвращает массив сообщений о ошибках, возникших в процессе попытки аутентификации.

Zend_Auth::authenticate() сохраняет идентификатор в постоянном хранилище. По умолчанию Zend_Auth использует класс хранилища Zend_Auth_Storage_Session, который в свою очередь использует Zend Session.

Zend Framework

Спасибо за внимание .

Жду ваших вопросов !