УДК 004.021

СОЗДАНИЕ ВЕБ-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ PHP-ФРЕЙМВОРКА YII

В.С. РАДЧЕНКО
(Представлено: Д.В. ПЯТКИН)

Представлен путь, пройденный при разработке ознакомительного приложения при изучении YII.

РАЗРАБОТКА ПРОЕКТА

В качестве проекта, для ознакомления с возможностями фреймворков, была выбрана разработка сайта-каталога одежды и обуви, который будет обладать следующими особенностями:

1) товары разделены по категориям, с возможностью создания подкатегорий;
2) удобная административная панель;
3) многоязычность, поддержка русского, румынского и английского языков;
4) поддержка высоких нагрузок(кэширование).

Для разработки такого проекта был выбран MVC-фреймворк yii, который обладает относительно низким «порогом вхождения» и большим русскоязычным сообществом.

Работу над созданием любого сайта можно разделить на следующие этапы:

1) выбор и установка необходимых инструментов;
2) разработка дизайна проекта и верстки;
3) проектирование и создание базы данных;
4) создание основы приложения и конфигурация;
5) установка верстки и доработка каркаса;
6) настройка кэширования, многоязычности и «красивых» ссылок.

1. Приложения, использованные при разработке проекта

Для оптимизации разработки проекта, а также удобства были использованы следующие приложения:

1) пакет «denwer»;
2) графический интерфейс «HeidiSQL» для управления СУБД MySQL;
3) редактор программного кода PhpStorm.

Каждое из этих приложений обладает рядом уникальных, незаменимых свойств, облегчающих разработку и тестирование web-приложения.

1.1. Система управления СУБД MySQL “HeidiSQL”

HeidiSQL - бесплатный клиент с открытым исходным кодом, а так же графический интерфейс для управления и администрирования СУБД MySQL. Проект находится в активном развитии и поддерживает множество возможностей:

1) поддержка множества одновременно открытых подключений с помощью TCP/IP, именованных каналов или SSH-туннелирования, с возможностью сохранения авторизационных данных.
2) управление пользователями и их правами на сервере в рамках базы данных или глобально.
3) поддержка управления серверными переменными;
4) просмотр серверной статистики и управление запущенными процессами с возможностью проанализировать выполняемые SOL-запросы и прервать “плохие”.
5) поддержка экспорта баз данных в SQL файл или на другой сервер, с возможностью последующего импорта;
6) просмотр и управление базами данных, таблицами, отображениями, триггерами и хранимыми процедурами.

2. Проектирование и создание базы данных

Во время проектирования база данных были выделены следующие таблицы:

1. таблица категорий «Category», которая хранит в себе название категорий на 3-х языках в древовидном виде Adjacency List: titleRu, titleEn, titleRo и parent;
2. таблица товаров «Price» с названиями и описаниями на 3-х языках, цену, размеры, наличие товара, а также дату добавления и поле для связи с категорией: titleRu, titleRo, titleEn, descrRu, descrRo, descrEn, idCategory, cost, time, present, sizes;
3. таблица для хранения пути к фотографиям «Photo» с полем для связи с товаром: idItem, path;
4. таблица для хранения текстовых страниц «Text» с названиями и описаниями на 3-х языках, а также красивым адресом для ссылок: titleRu, titleRo, titleEn, descrRu, descrRo, descrEn, name.

Помимо перечисленных выше полей каждая таблица обладает первичным ключом id для связи с другими таблицами и удобства управления записями.

3. Доработка правил валидации моделей и провайдеров данных

Так как модель отвечает за выборку и валидацию (проверку) данных, то в ней описываются условия выборки и проверки данных.

Правила валидации описываются в методе rules модели. Этот метод должен вернуть массив с описанием правил валидации, например:

```
public function rules()
{array(('titleRu,titleEn,titleRo','length','max'=>255,'min'=>4),('titleRu,titleEn,titleRo','required'),('parent',
'numerical', 'integerOnly'=>true,'allowEmpty'=>false),
);
}
```

Формат описания правила выглядит следующим образом:

array(<имя св-ва/св-в модели (полей в таблице)>,<псевдоним валидатора>,<имя доп. параметра валидатора 1>=><значение параметра 1>, …)

Каждое имя валидатора является псевдонимом класса. Это позволяет создавать и использовать свои классы валидации. Основные валидаторы:

1) default (CDefaultValueValidator) - присваивает значение по умолчанию выбранным атрибутам;
2) in (CRangeValidator) - проверяет, содержится ли значение атрибута в указанном наборе значений;
3) length (CStringValidator) - проверяет, находится ли длина строкового значения атрибута в в указанном интервале;
4) numerical (CNumberValidator) - проверяет, является ли значение атрибута числом;
5) required (CRequiredValidator) - проверяет, не является ли значение атрибута пустым;
6) match: псевдоним класса CRegularExpressionValidator, проверяющего значение атрибута на соответствие регулярному выражению;
7) unique: псевдоним класса CUniqueValidator, который проверяет, является ли значение атрибута уникальными в пределах столбца таблицы базы данных;

Дополнительные параметры каждого из валидаторов подробно описаны в руководстве по yii.

После того как правила валидации были настроены, в модель были добавлены статические методы для выборки списка товаров. Каждой такой метод использует провайдер данных и выглядит примерно так:

```
public static function searchByCategory($category,$pageSize)
{
// создание условия выборки
$criteria=new CDbCriteria();
$criteria->select='Id, title'.ucfirst(Yii::app()->language).' as title,cost';
$criteria->addInCondition('idCategory',Category::getAllChildren($category));
// создание провайдера данныхnew CActiveDataProvider(__CLASS__,array(
// условие выборки
'criteria'=>$criteria,
// настройка постраничного вывода
'pagination'=>array(
'pageSize'=>$pageSize
),
// поддержка сортировки
'sort'=>array(
'defaultOrder'=>'title'.ucfirst(Yii::app()->language),
'attributes'=>array(
'title'.ucfirst(Yii::app()->language),
'time',
'cost'
)
)
));
}
```

Как видно из приведенного выше кода провайдер данных является экземпляром класса CActiveDataProvider (а так же потомком CDataProvider) и очень гибок в настройке.

4. Настройка кэширования, многоязычности и «красивых» ссылок

После того как все основные действия по доработке каркаса выполнены, осталось добавить поддержку больших нагрузок (кэширования), настроить многоязычность и красивые ссылки.

Для активации кэширования в файле конфигурации было добавлено описание компонента приложения cache с классом CFileCache, отвечающим за хранение кэша в локальных файлах.

Для настройки кэширования в контроллерах использовался компонент COutputCache, описанный в методе filters контроллера:

```
array(
// Компонент отключен для администраторских действий
'COutputCache -create, update, upload',
// Срок годности
'duration'=>24*3600*365,
// зависимости
'dependency'=>array(
'class'=>'CChainedCacheDependency',
'dependencies'=>array(
// по глобальному изменению pm_time
new CGlobalStateCacheDependency('pm_time'),
// по глобальному изменению cu_time CGlobalStateCacheDependency('cu_time'),
),
),
// данные в кэше отличаются по категории, сортировке, страницам, языку
'varyByParam'=>array('category','sort','page','lang'),
// и по статусу авторизованности пользователя
'varyByExpression'=>'Yii::app()->user->isGuest',
// кэш используется только для GET-запросов
'requestTypes'=>array('GET'),
),
```

где cu_time - последнее время обновления категорий, а pu_time - товаров и меняются по событию сохранения модели.

Так как, по-умолчанию, приложение уже поддерживает многоязычность с помощью класса CPhpMessageSource, который хранит переводы в php-файлах перевода, то в проекте использовался именно этот способ хранения.

Каждое сообщение перевода относится к какой-либо категории. Сообщения переводов хранятся по следующему пути: protected/messages/<код языка>/<имя категории>.php. Файлы переводы содержат в себе ассоциативный массив, где ключем является фраза на исходном (английский) языке, а значение - переведенное для данного кода (русский, например).

Для перевода сообщения, фреймфорк yii предоставляет статический метод Yii::t(<имя категории>,<сообщение>), а для хранения текущего языка в ссылке в файл конфигурации было подключено стороннее расширение в качестве компонента приложения urlManager: LangUrlManager.

Для активации «красивых», человекопонятных ссылок компонент приложения urlManager был настроен следующим образом:

```
'urlManager'=>array(
'class'=>'application.extensions.urlManager.CLangUrlManager',
//формировать ссылку в формате /route/p1/v1/p2/v2/...
'urlFormat'=>'path',
'rules'=>array(
// код языка всегда расположен впереди ссылки
'<lang:(en|ru|ro)>'=>'/price',
'<lang:(en|ru|ro)>/<_c>'=>'<_c>',
'<lang:(en|ru|ro)>/<_c>/<_a>'=>'<_c>/<_a>',
'<lang:(en|ru|ro)>/text/index/<page:..*>'=>'text/index',
),
// не выводить index.php в ссылке
'showScriptName'=>false,
),
```

а в корневой папке сайта (www) был добавлен файл .htaccess с активацией и настройкой mod_rewrite в web-сервере apache:

```
Options +FollowSymLinks*/*on%{REQUEST_FILENAME} !-f%{REQUEST_FILENAME} !-d. index.php
```

ЗАКЛЮЧЕНИЕ

При создании данного приложения были рассмотренны некоторые из большого количества возможностей фреймворка YII и изучены на примере созданной программы. В данной статье мыф ещё раз убедились в обширных возможностях фреймворка YII.

ЛИТЕРАТУРА

1. Русскоязычное сообщество YII. – Режим доступа: http://yiiframework.ru/. – Дата обращения: 17.09.2018.
2. Официальный сайт YII Framework [Электронный ресурс]. – Режим доступа: <http://yiiframework.com/. – Дата обращения: 18.09.2018.
3. Официальный сайт PhpStorm. – Режим доступа: https://jetbrains.ru/products/phpstorm/. – Дата обращения: 17.09.2018.
4. Сайт PhpFramework. – Режим доступа: https://geekbrains.ru/posts/php_frameworks. – Дата обращения: 17.09.2018.