Kohana:敏捷的 PHP 框架

原文：Kohana: The Swift PHP Framework 作者：Cristian Gilè 中文翻译：淡水

Kohana 是 PHP5 的开发框架，基于 MVC 架构。你选择它的原因有好些，但是最主要的还是安全，轻量级和简单。在这个指南里，我将介绍它的主要特点，并以一个简单的事例告诉你 Kohana 可以为你节省多少时间。

Step 1： Kohana 是什么？

Kohana 是 PHP5 的开发框架，基于 MVC 架构。MVC 将应用逻辑分开，让我们的代码更加干净并且更快的找到 bug。这种模式是这样的：

- Model 表示应用中的数据运行，通常是数据库
- View 包含显示部分，如 HTML,CSS 和 JavaScript
- Controller 接受用户的输入和提交到 view 或 model

Kohana 起源于 Codeigniter(CI)，CI 是 EllisLab 的开源作品。他们有很多相似的地方，但是 Kohana 的所有代码是重新编写或完全改写了。你可以访问 Kohana 的官方网站，它的主要特点如下：

- 高安全性
- 很轻巧
- 容易学习
- 使用 MVC 模式
- 100%UTF-8 编码
- 松耦合结构
- 易于扩展

Step 2： 下载 Kohana

访问 Kohana 官网下载最新版本的 Kohana，也可以访问 Kohana 中文网 下载。

Step 3： 安装 Kohana

当你完成了下载：

1. 解压
2. 重命名“Kohana_vx.x.x”文件夹，比如改为“kohana”，把他上传到你的 web 服务器的根目录。
3. 编辑全局配置文件 application/config/config.php 如下：
   $config['site_domain'] = 'localhost/kohana';
4. 如果你使用的是 unix-like 系统，子文件夹可能丢失了其权限。改变他们的权限到 755.
5. 确保 application/logs 和 application/cache 文件夹都是可写的。改变权限到 666

6. 现在，在你的浏览器中访问 http://localhost/kohana/ ．框架会自动调用 install.php 脚本检测你的服务器是否符合需求。

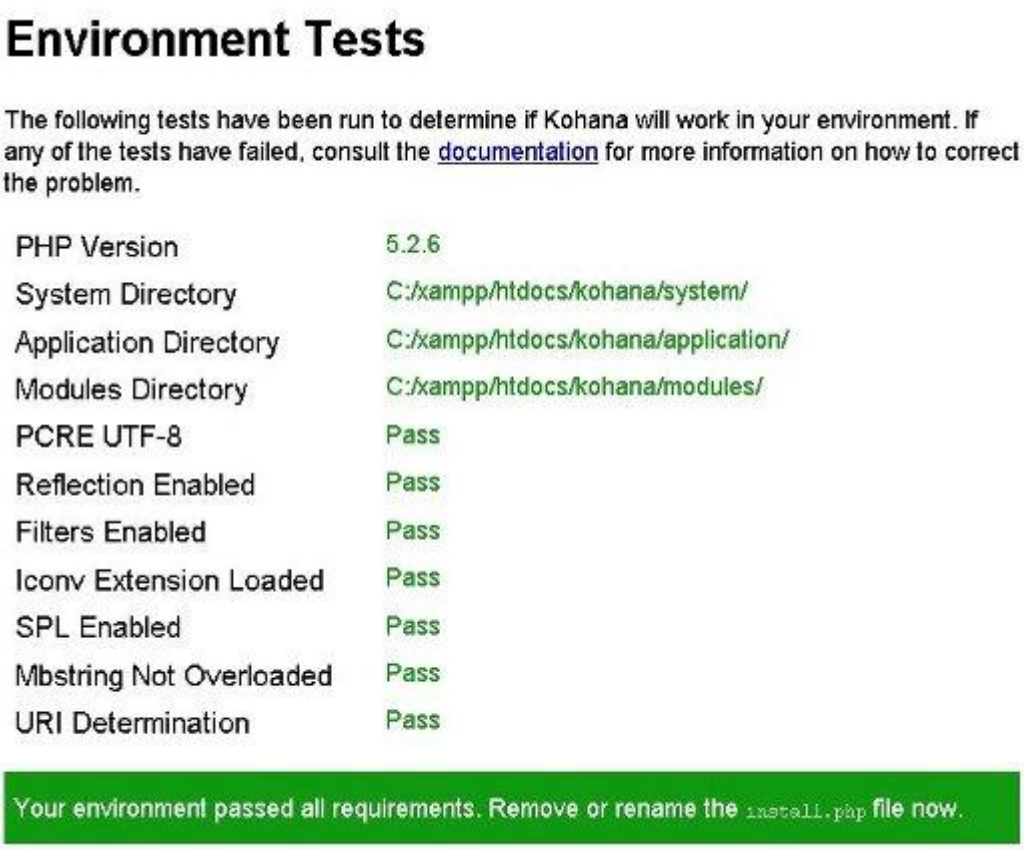
只要很小的配置，kohana 几乎就可以在任何环境下运行。服务器要求：

- 支持 Unicode
- PHP 版本至少 5.2.3
- 一个 HTTP 服务器。我建议你使用 XAMPP.XAMPP 是易于使用的 MySQL,PHP 和 Perl 的集成安装包。
- 数据库（MsSQL, MySQL, MySQLi, PostgreSQL, PDOsqlite）

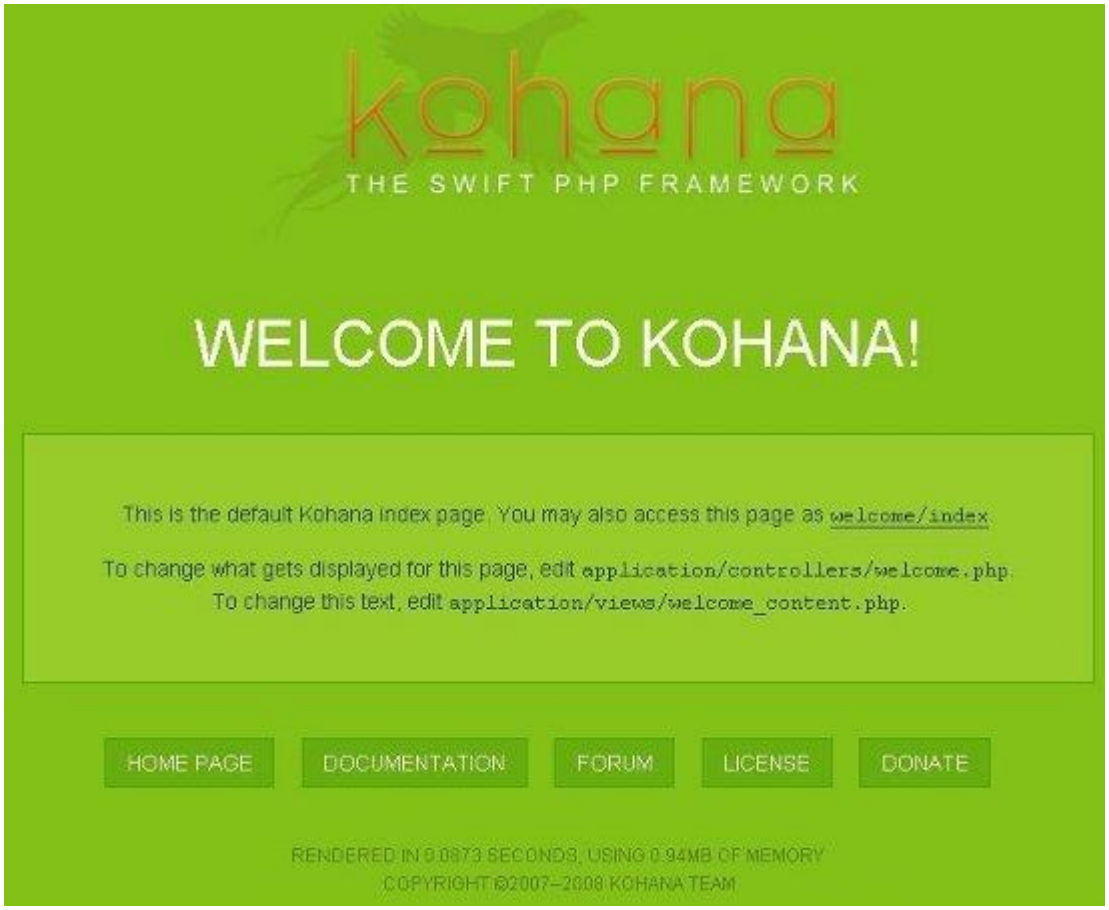
也需要一些扩展

- PCRE
- Iconv
- Mcrypt
- SPL

如果您安装成功完成，您将被重定向到这个测试页：



如果有测试不通的，你需要修改它们以达到要求。
如果所有的测试都通过了。你要重命名 install.php 脚本，或是删除它。刷新一下，你会看到如下的欢迎页面：

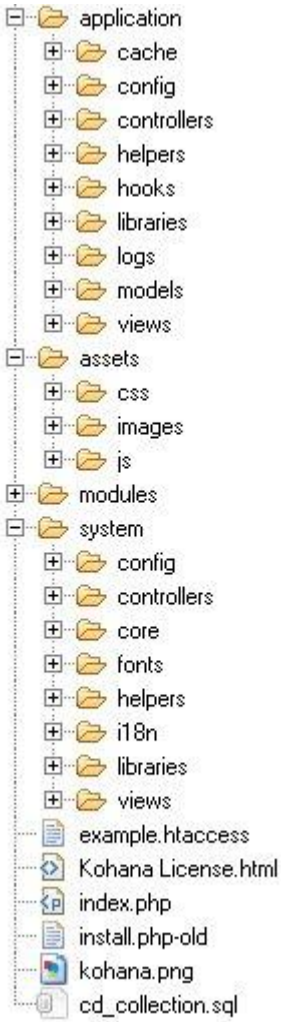


Step 4：配置 Kohana

Kohana 已经准备就绪。没有任何其他的必要配置。这个框架是不是让人惊奇呢？让我们回顾一些代码。

Step 5：你的第一个 Kohana 项目

一般的编程教程都是以“Hello world”开始的。我认为，一个简单的应用可以更清楚的了解框架是怎样工作的。所以，我将建立一个 CD 收藏管理系统——只是一个有趣的演示。在开始写代码之前，简要了解一下 Kohana 的文件系统是必要的。



我们的应用将会放在 application 文件夹中。在这个文件夹中有多个子文件夹，在我们的项目中，需要用到以下的：

- **Config** 文件夹，所有的配置文件都是以静态数组的形式放在其中。
- **Controller** 文件夹 放置我们自己的控制器类
- **Models** 文件夹 放置我们自己的模型类
- **Views** 文件夹 放在我们的 html 视图（或是其他任何用于在终端显示的标记语言或脚本）

其他的子文件夹在这个教程里用不到，如果你想了解，你可以到 Kohana 的官网去看看。

System 文件夹放置了 kohana 的核心文件和一些工具，如 libraries，helpers 和预定义的配置文件。在这个项目里，我们将使用一些 libraries 和一些 helpers 这些好的工具将会加快我们的进度。
Assets 文件夹不是原来 kohana 里的。我有一些媒体文件比如 CSS,JS 和图片等，都是放到这里的。我会告诉你如何在项目中引用他们。
Modules 文件夹是放置可重用的代码的。比如 kohana 内置的认证模块就是一个例子。

我们概要的了解了 kohana 的文件系统，虽然说得比较简单但是对我们的教程来说已经足够了。我不想将太多的理论。

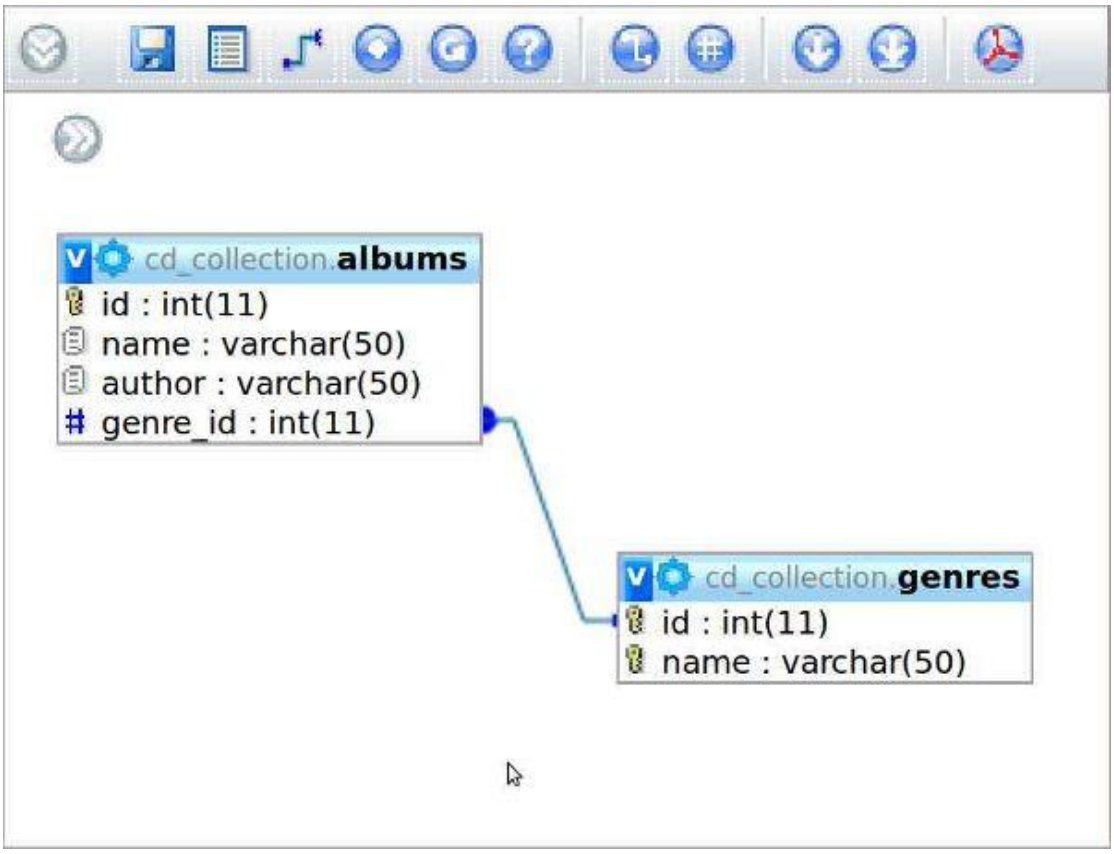
Step 6：项目数据库

虽然我选择 MySQL 作为我的数据库管理系统，不过还请记住 kohana 还支持 MsSQL, MySQLi, PostgreSQL，PDOsqlite。建立一个"cd_collection"数据库，你也可以用你喜欢的名字。用 phpMyAdmin 或是其他工具在 MySQL 上运行如下的 SQL：

```
CREATE TABLE `albums` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(50) collate utf8_bin NOT NULL,
  `author` varchar(50) collate utf8_bin NOT NULL,
  `genre_id` int(11) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `genre_id` (`genre_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=19 ;

INSERT INTO `albums` (`id`, `name`, `author`, `genre_id`) VALUES
(2, 'Lines, Vines And Trying Times', 'Jonas Brothers', 16),
(3, 'The E.N.D.', 'The Black Eyed Peas', 16),
(4, 'Relapse', 'Eminem', 18),
(5, 'Monuments And Melodies', 'Incubus', 1),
(6, 'Thriller', 'Michael Jackson', 16),
(7, 'Back in Black', 'AC/DC', 4),
(8, 'The Dark Side of the Moon', 'Pink Floyd', 4),
(9, 'Bat out of Hell', 'Meat Loaf', 4),
(10, 'Backstreet Boys', 'Millennium', 16),
(11, 'Rumours', 'Fleetwood Mac', 4),
(12, 'Come on Over', 'Shania Twain', 16),
(13, 'Led Zeppelin IV', 'Led Zeppelin', 4),
(14, 'Jagged Little Pill', 'Alanis Morissette', 4),
(15, 'Sgt. Pepper''s Lonely Hearts Club Band', 'The Beatles', 16),
(16, 'Falling into You', 'C √ ©line Dion', 16),
(17, 'Music Box', 'Mariah Carey', 16),
(18, 'Born in the U.S.A.', 'Bruce Springsteen', 4);

CREATE TABLE `genres` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(50) collate utf8_bin NOT NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=22 ;

INSERT INTO `genres` (`id`, `name`) VALUES
(1, 'Alternative Rock'),
```

```
(2, 'Blues'),
(3, 'Classical'),
(4, 'Rock'),
(5, 'Country'),
(6, 'Dance'),
(7, 'Folk'),
(8, 'Metal'),
(9, 'Hawaiian'),
(10, 'Imports'),
(11, 'Indie Music'),
(12, 'Jazz'),
(13, 'Latin'),
(14, 'New Age'),
(15, 'Opera'),
(16, 'Pop'),
(17, 'Soul'),
(18, 'Rap'),
(20, 'Soundtracks'),
(21, 'World Music');

ALTER TABLE `albums`
  ADD CONSTRAINT `genre_inter_relational_constraint` FOREIGN KEY (`genre_id`) REFERENCES `genres` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

SQL 建立了两个表 albums 和 genres，并填充了一些数据。最后一句 SQL 添加了一个外键约束“genre_id”.
数据结构很简单，不需要太多解释。



现在，你要告诉 kohana 数据库在哪里？如何访问它？编辑全局配置文件 system/config/database.php 代码如下：

```
$config['default'] = array
(
	'benchmark'      => TRUE,
	'persistent'     => FALSE,
	'connection'     => array
	(
		'type'     => 'mysql',
		'user'     => 'root',
		'pass'     => 'root',
		'host'     => 'localhost',
		'port'     => FALSE,
		'socket'   => FALSE,
		'database' => 'cd_collection'
	),
	'character_set' => 'utf8',
	'table_prefix'  => '',
	'object'        => TRUE,
	'cache'         => FALSE,
	'escape'        => TRUE
);
```

代码告诉 kohana，连接到本地的“cd_collection”数据库，用户名是 root 密码 root。你要根据自己的情况来配置它。

Step 7：建立控制器

建立我们的第一个控制器。记住以下预定

- 控制器文件名应该小写，如 album.php
- 控制器类必须映射文件名而且首字母大写，还要添加_Controller.如：Album_Controller
- 必须从父类继承

此外，请记住 kohana 的 URL 结构和你调用控制器的方法是对应的；如 http://hostname/kohana_directory/index.php/controller/function

我们来看看下面这个简单的控制器。

```
<?php defined('SYSPATH') OR die('No direct access allowed.');

class Album_Controller extends Controller
{
	public function __construct()
	{
		parent::__construct();
	}

	public function index()
	{
		echo "My first controller";
	}
}
```

你要看懂它，就要有 PHP5 的 OOP 知识。可以到这里学习。
__construct 是构造函数，它初始化类，并调用父类的构造函数。Index 方法是默认方法，所以我们可以只调用控制器不指定其方法。（例如：http://localhost/index.php/kohana/album 没有调用任何方法，这时 index 方法会被调用。）
了解了这些基本约定，我们把焦点回到我们的应用程序。Album 控制器包含了 CD 收藏管理的所有动作。在这个控制器中我们可以新建一个专辑，显示我们数据库中的相册，还可以更新和删除专辑。
让我们对上面的类做如下的变动。
在 application/controllers/ 下创建 album.php ：

```
<?php defined('SYSPATH') OR die('No direct access allowed.');

class Album_Controller extends Controller
```

```
{
    private $album_model;
    private $genre_model;

    private $list_view;
    private $create_view;
    private $update_view;

    public function __construct()
    {
        parent::__construct();
        $this->album_model      = new Album_Model;
        $this->genre_model      = new Genre_Model;
        $this->list_view        = new View('list');
        $this->update_view      = new View('update');
        $this->create_view      = new View('create');
    }

    public function index()
    {
        $this->show_albums_list();
    }

    private function show_albums_list()
    {
        $albums_list = $this->album_model->get_list();
        $this->list_view->set('albums_list',$albums_list);
        $this->list_view->render(TRUE);
    }

    public function show_create_editor()
    {
        $this->create_view->set('genres_list',$this->get_genres_list());
        $this->create_view->render(TRUE);
    }

    public function show_update_editor($id)
    {
        $album_data = $this->album_model->read($id);
        $this->update_view->set('album_id',$album_data[0]->id);
        $this->update_view->set('name',$album_data[0]->name);
        $this->update_view->set('author',$album_data[0]->author);
        $this->update_view->set('genre_id',$album_data[0]->genre_id);
        $this->update_view->set('genres_list',$this->get_genres_list());
```

```
        $this->update_view->render(TRUE);
    }

    public function create()
    {
        $album_data=array(
        'name'          => $this->input->post('name'),
        'author'        => $this->input->post('author'),
        'genre_id'    => $this->input->post('genre_id')
        );
        $this->album_model->create($album_data);
        url::redirect('album');
    }

    public function update()
    {
        $album_data = array(
            'name'          => $this->input->post('name'),
            'author'        => $this->input->post('author'),
            'genre_id'    => $this->input->post('genre_id')
        );
        $this->album_model->update($this->input->post('album_id'),$album_data);
        url::redirect('album');
    }

    public function delete($id)
    {
        $this->album_model->delete($id);
        url::redirect('album');
    }

    private function get_genres_list()
    {
        $db_genres_list    = $this->genre_model->get_list();
        $genres_list    = array();

        if(sizeof($db_genres_list) >= 1)
        {
            foreach($db_genres_list as $item)
            {
                $genres_list[$item->id] = $item->name;
            }
        }
        return $genres_list;
```

```
    }
}
```

我们来解读一下这段代码。

类首先声明了 5 个成员变量：

```
private $album_model;
private $genre_model;

private $list_view;
private $create_view;
private $update_view;
```

他们都是私有的（private），因为我只想让这个类本身去访问它们。
在构造函数和视图对象中使用这个 5 个成员变量：

```
$this->album_model      = new Album_Model;
$this->genre_model      = new Genre_Model;
$this->list_view      = new View('list');
$this->update_view      = new View('update');
$this->create_view      = new View('create');
```

建立一个 model 对象（模型对象）使用下面的语法：

```
$obj_name = new Name_Model;
```

建立一个 view 对象（视图对象）使用下面的语法：

```
$obj_name = new View('view_filename_without_extension');
```

现在有两个模型对象 album 和 genre 和三个视图对象。
在 index 方法中调用 show_albums_list 方法列出数据库中的所有专辑。

```
$albums_list = $this->album_model->get_list();
$this->list_view->set('albums_list',$albums_list);
$this->list_view->render(TRUE);
```

在这个方法中，你可以看到怎样调用 model 和 view 对象的方法。Get_list 是 model 的一个方法（稍后我们将看到他）它返回数据库中的所有专辑。其结果保存在$album_list 数组中，通过控制器中调用 view 的 set 方法。这个方法需要两参数：一个新的空变量（album_list）用来存放已有变量（$album_list）.现在新变量（album_list）包含了$album_list 的数组（待会儿我们将会看到他怎样显示）。Render 方法的参数为 TRUE 时，输出数据到浏览器。

Show_create_editor 方法显示一个新增专辑的用户界面。

```
$this->create_view->set('genres_list',$this->get_genres_list());
$this->create_view->render(TRUE);
```

风格列表就显示在视图上了。

Show_update_editor 方法显示更新专辑的用户界面。

```
$album_data = $this->album_model->read($id);
$this->update_view->set('album_id',$album_data[0]->id);
$this->update_view->set('name',$album_data[0]->name);
$this->update_view->set('author',$album_data[0]->author);
$this->update_view->set('genre_id',$album_data[0]->genre_id);
$this->update_view->set('genres_list',$this->get_genres_list());
$this->update_view->render(TRUE);
```

Read 是 model 中的方法（稍后谈到他）它返回 id 等于$id 的专辑数据（$album_data）。然后把每个部分都输出的视图。

Create 方法从视图文件得到一个新的专辑数据，并添加到数据库中。

```
$album_data=array(
'name'          => $this->input->post('name'),
'author'        => $this->input->post('author'),
'genre_id'   => $this->input->post('genre_id')
);
$this->album_model->create($album_data);
url::redirect('album');
```

$album_data 是从视图的表单中得到的 POST 过来的数组数据。保存专辑是通过 model 的 create 方法。最后一行是调用的 helper 方法。Helper 是一些帮助你的一些简单的方法，它框架自动加载的。Helper 被定义成静态方法，所以无需实例化。在这里使用 url helper 的 redirect 方法告诉 kohana 重新定向浏览器到 album 控制器。避免了重复插入（例如按 F5）。

Helper 是简单的函数，协助我们的开发的。

最后一个方法 get_genres_list 得到风格列表（$db_genres_list）,并为选择框建立一个数组（$genres_list）。

```
$db_genres_list   = $this->genre_model->get_list();
$genres_list   = array();

if(sizeof($db_genres_list) >= 1)
{
    foreach($db_genres_list as $item)
    {
        $genres_list[$item->id] = $item->name;
    }
}
return $genres_list;
```

Step 8：建立项目模型

现在来建立我们应用程序的模型。记住以下规约。

- 模型文件命名必须小写，例如：album.php
- 模型类名必须映射到文件名并且首字母大写，而且加上_Model.例如：Album_Model
- 必须从父类继承

在 application/models/ 下建立 album.php，代码如下：

```
<?php defined('SYSPATH') OR die('No direct access allowed.');

class Album_Model extends Model
{
	private $album_table;
	private $genre_table;

	public function __construct()
		{
			parent::__construct();
		$this->album_table = 'albums';
		$this->genre_table = 'genres';
	}

	public function read($id)
	{
		$this->db->where('id', $id);
		$query = $this->db->get($this->album_table);
		return $query->result_array();
	}

	public function delete($id)
	{
		$this->db->delete($this->album_table, array('id' => $id));
	}

	public function update($id,$data)
	{
		$this->db->update($this->album_table, $data, array('id' => $id));
	}

	public function create($data)
	{
		$this->db->insert($this->album_table, $data);
	}
```

```
    public function get_list()
    {
        $this->db->select('albums.id as id,albums.name as name,albums.author as author,
genres.name as genre');
        $this->db->from($this->album_table);
        $this->db->join($this->genre_table,'genres.id','albums.genre_id');
        $query = $this->db->get();
        return $query->result_array();
    }
}
```

模型中的所有方法我们都是用的 Query builder 语法。这是 kohana 提供的加快开发速度并简化数据库查询的工具。
类的上方定义了两个成员变量：

```
        private $album_table;
        private $genre_table;
```

都是私有成员变量，只有类自身才能访问。它们分别是数据库中的两个表的名称。

构造中的第一行，调用父类构造，也就加载了 kohana 的数据库功能类到$this->db;第二，三行是初始化两个成员变量。

```
parent::__construct();
$this->album_table = 'albums';
$this->genre_table = 'genres';
```

Read 方法根据唯一值（$id）检索专辑记录.

```
$this->db->where('id', $id);
$query = $this->db->get($this->album_table);
return $query->result_array();
```

delete 方法删除专辑表中的一条记录根据标识符$id.

```
$this->db->delete($this->album_table, array('id' => $id));
```

Update 方法更新数据根据标志符$id,新的值来源于数组$data.

```
$this->db->update($this->album_table, $data, array('id' => $id));
```

$date 数组必须是以字段名作为 key，字段的值作为其 value 的数组。它应该是如下格式的：

```
$data = array(
    'name'          =>  'album_name',
    'author'        =>  'author_name',
    'genre_id'      =>  'genre_id'
    );
```

Insert 方法插入一条新的记录，其值为$data 数组。

```
$this->db->insert($this->album_table, $data);
```

$data 数组必须是如下格式：

```
$data = array(
    'id'            =>  'album_id',
    'name'              =>  'album_name',
    'author'        =>  'author_name',
    'genre_id'          =>  'genre_id'
);
```

Get_list 方法检索出所有专辑的记录。

```
        $this->db->select('albums.id as id,albums.name as name,albums.author as author, genres.name as genre');
        $this->db->from($this->album_table);
        $this->db->join($this->genre_table,'genres.id','albums.genre_id');
        $query = $this->db->get();
        return $query->result_array();
```

现在，轮到 genre 模型了。在 application/models/ 下建立 genre.php 其代码如下：

```
<?php defined('SYSPATH') OR die('No direct access allowed.');

class Genre_Model extends Model
{
    private $genre_table;

    function __construct()
    {
        parent::__construct();
        $this->genre_table = 'genres';
    }

    function get_list()
    {
        $query = $this->db->get($this->genre_table);
        return  $query->result_array();
    }
}
```

这个 model 很简单，我就不多讲了。现在 model（模型）和 controller（控制器）都已经有了，现在开始做 views（视图）吧

Step 9：建立项目视图

视图文件是你应用程序的表现形式。MVC 分开的目，可以让你更专注于你的应用逻辑，提高代码的重用率，而且这样的代码很整洁。在这个项目中，我们需要 3 个视图：一个专辑列表，一个添加专辑，还有一个是编辑已有专辑的视图。

在 application/views/ 下建立 list.php 代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<?php
    echo html::stylesheet(array
        (
          'assets/css/style'
        ),
        array
        (
          'screen'
        ), FALSE);
?>
        <title>CD COLLECTION</title>
        </head>
        <body>
        <?php
                echo html::image('assets/images/add.png');
                echo html::anchor('album/show_create_editor', 'Add new album');
        ?>
        <table class="list" cellspacing="0">
        <tr>
            <td colspan="5" class="list_title">CD Collection</td>
        </tr>
        <tr>
            <td class="headers">Album name</td>
            <td class="headers">Author</td>
            <td colspan='3' class="headers">Genre</td>

        </tr>
        <?php
            foreach($albums_list as $item)
            {
                echo "<tr>";
                echo "<td class='item'>".$item->name."</td>";
                echo "<td class='item'>".$item->author."</td>";
```

```
                    echo "<td class='item'>".$item->genre."</td>";
                    echo "<td class='item'>".html::anchor('album/delete/'.$item->id,html::image('assets/images/delete.png'))."</td>";
                    echo "<td class='item'>".html::anchor('album/show_update_editor/'.$item->id,html::image('assets/images/edit.png'))."</td>";
                    echo "</tr>";
                }
            ?>
            </table>
            </body>
            </html>
```

这个视图，通过 foreach 循环打印出 html 表格，从而将所有专辑列表显示。在每一行都有两个图片，一个“红叉”，一个“记事本”。他们分别连接到 delete 和 update 方法。他们都是通过 GET 请求到专辑的 id 进行操作的。列表上方有个添加按钮，可以添加一个新的专辑。在以上代码中，我们还使用了 kohana 提供的 html helper 加快了我们的速度。

现在建立添加专辑的视图。在 application/views/ 下建立 create.php 代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<?php
    echo html::stylesheet(array
    (
        'assets/css/style'
    ),
    array
    (
        'screen'
    ), FALSE);
?>
<title>CD COLLECTION</title>
</head>
<body>
<?php echo form::open('album/create'); ?>
<table class='editor'>
<tr>
    <td colspan='2' class='editor_title'>Create new album</td>
</tr>
<?php
    echo "<tr>";
```

```
        echo "<td>".form::label('name', 'Name: ')."</td>";
        echo "<td>".form::input('name', '')."</td>";
        echo "</tr>";

        echo "<tr>";
        echo "<td>".form::label('author', 'Author: ')."</td>";
        echo "<td>".form::input('author', '')."</td>";
        echo "<tr/>";

        echo "<tr>";
        echo "<td>".form::label('genre', 'Genre: ')."</td>";
        echo "<td>".form::dropdown('genre_id',$genres_list)."</td>";
        echo "<tr/>";

        echo "<tr>";
        echo "<td colspan='2' align='left'>".form::submit('submit', 'Create album')."</td>";
        echo "</tr>";
?>
</table>
<?php echo form::close(); ?>
</body>
</html>
```

最后我吗建立 update（编辑更新）视图。在 application/views/ 下建立 update.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<?php
        echo html::stylesheet(array
        (
                'assets/css/style'
        ),
        array
        (
                'screen'
        ), FALSE);
?>
<title>CD COLLECTION</title>
</head>
<body>
<?php echo form::open('album/update'); ?>
<table class='editor'>
<tr>
```

```
        <td colspan='2' class='editor_title'>Update album</td>
</tr>
<?php
        echo "<tr>";
        echo "<td>".form::label('name', 'Name: ')."</td>";
        echo "<td>".form::input('name', $name)."</td>";
        echo "</tr>";

        echo "<tr>";
        echo "<td>".form::label('author', 'Author: ')."</td>";
        echo "<td>".form::input('author', $author)."</td>";
        echo "<tr/>";

        echo "<tr>";
        echo "<td>".form::label('genre', 'Genre: ')."</td>";
        echo "<td>".form::dropdown('genre_id',$genres_list, $genre_id)."</td>";
        echo "<tr/>";

        echo "<tr>";
        echo "<td colspan='2' align='left'>".form::submit('submit', 'Update album')."</td>";
        echo "</tr>";

?>
</table>
<?php
        echo form::hidden('album_id',$album_id);
        echo form::close();
?>
</body>
</html>
```

首先是一个添加新专辑的简单编辑界面。像 author 和 name 使用了 html 的 input 控件，genre 使用了下拉列表框。当用户点击了 create（建立）按钮，所有的信息将通过 POST 传递到 album 控制器的 create 或 update 方法。当控制器收到这些变量，将会调用相应的方法插入或是更新一条记录到数据库。在以上两个视图中，都用了 Kohana 提供的 from helper。

为了让我们的程序有些样式，在 kohana 的根目录下建立 assets 文件夹和 application 文件夹平级。再在其下建立 css 和 images 文件夹。
在 css 中建立 style.css，代码如下：

```
a {
        font-family: Verdana, Geneva, Arial, Helvetica, sans-serif ;
        font-weight: normal;
        font-size: 12px;
        color: #00F;
```

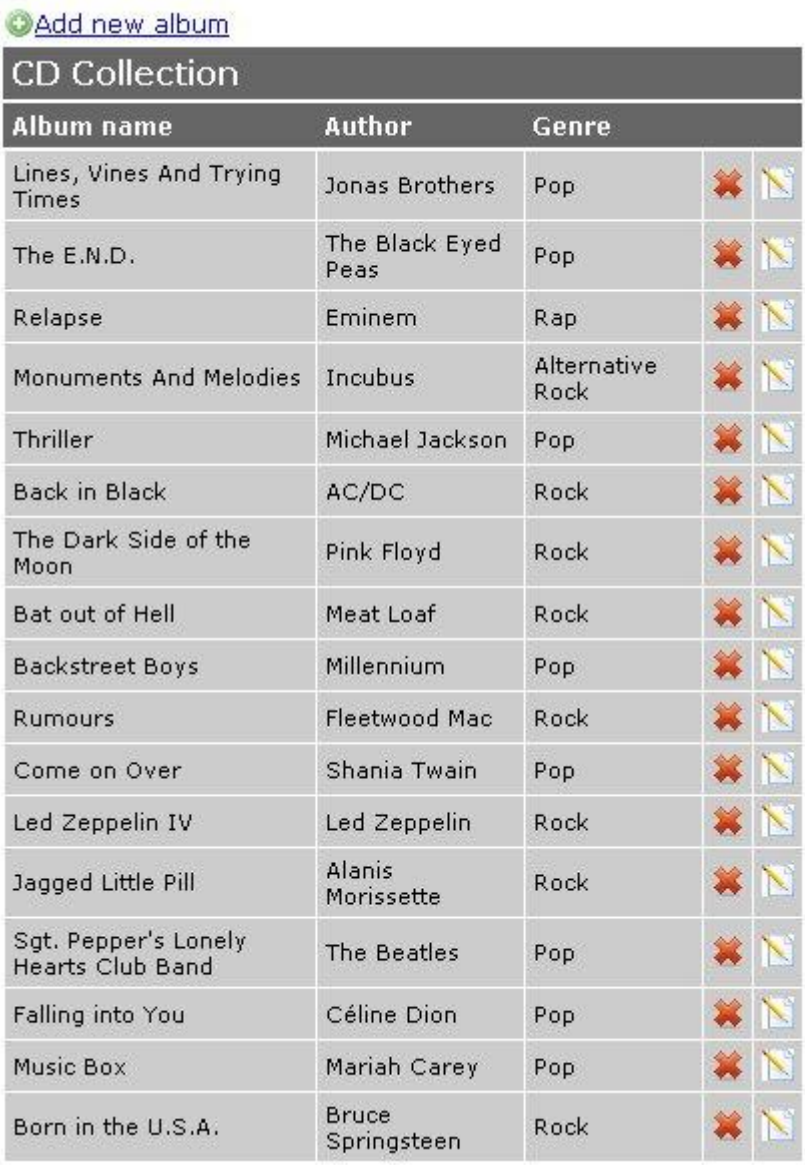
```
	vertical-align:text-top;
}

img {
	border: 0;
}

label {
	font-family: Verdana, Geneva, Arial, Helvetica, sans-serif ;
	font-weight: normal;
	font-size: 12px;
}

input {
	border: 1px solid #000;
}

select {
	width:185px;
}

table.editor
{
	text-align: center;
	font-family: Verdana, Geneva, Arial, Helvetica, sans-serif ;
	font-weight: normal;
	font-size: 11px;
	color: #fff;
	width: 280px;
	background-color: #666;
	border: 0px;
	border-collapse: collapse;
	border-spacing: 0px;
}

table.editor td.editor_title
{
	background-color: #666;
	color: #fff;
	padding: 4px;
	text-align: left;
	font-weight: bold;
	font-size: 16px;
}
```

```
table.editor td
{
	padding: 4px;
}

table.list
{
	text-align: center;
	font-family: Verdana, Geneva, Arial, Helvetica, sans-serif ;
	font-weight: normal;
	font-size: 11px;
	color: #fff;
	width: 280px;
	background-color: #666;
	border: 0px;
	border-collapse: collapse;
	border-spacing: 0px;
}

table.list td.item
{
	background-color: #CCC;
	color: #000;
	padding: 4px;
	text-align: left;
	border: 1px #fff solid;
}

table.list td.list_title,table.list td.headers
{
	background-color: #666;
	color: #fff;
	padding: 4px;
	text-align: left;
	border-bottom: 2px #fff solid;
	font-weight: bold;
}

table.list td.list_title
{
	font-size: 16px;
}
```

```
table.list td.headers
{
	font-size: 12px;
}
```

把  拷贝到 images 文件夹。

就是这样了。打开你的浏览器，访问 http://localhost/kohana/index.php/album 你会看到以下类似的界面：



如果你试着添加一个新的专辑或是编辑一个已有专辑，你会看到下面的界面：

Step 10：最后的想法

当然这个程序还有需要改进的地方，但是你已经用很少的代码完成了一个 web 应用程序。现在你应该了解 Kohana 的 MVC 架构。并知道如何使用它的数据库类和 helper。更多资料请参考官方文档，或中文手册。