环境要求：

1. WAMP版本（PHP5.3.7以上）

wampserver2.2e-php5.4.3-httpd2.2.22-mysql5.5.24-x64.exe

（PHP5.4.3 apache2.2.22 mysql5.5.24或沿用上一版本5.1.36）

1. Windows补丁vcredist\_x64.exe；
2. ZendFramework-2.0.5.zip
3. PHPUnit3.7.10
4. PhpDocumentor2.0.0a12

Zend Framework2简介

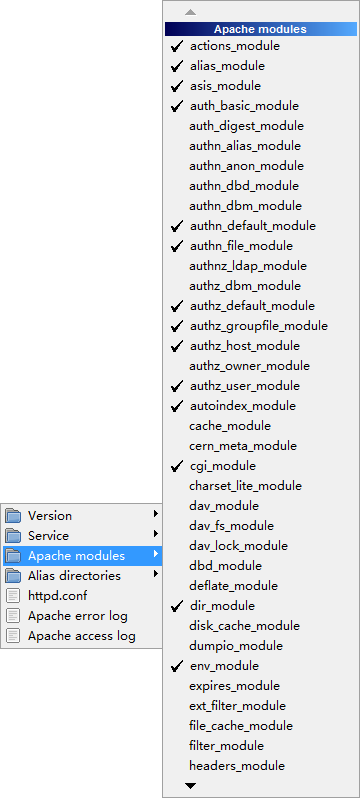
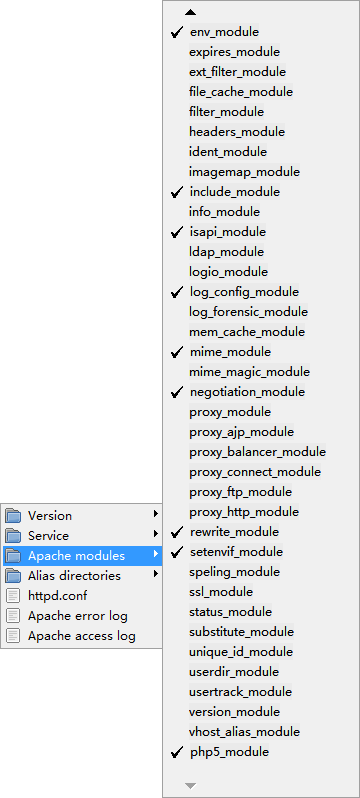
Zend Framework2（简称ZF2）是一个用于开发web应用和web Service的开源框架，要求PHP5.3及以上版本。ZF2使用100%面向对象的编码方式，应用了PHP5.3的大部分新特性，例如命名空间、晚绑定、lambda函数和封装包。

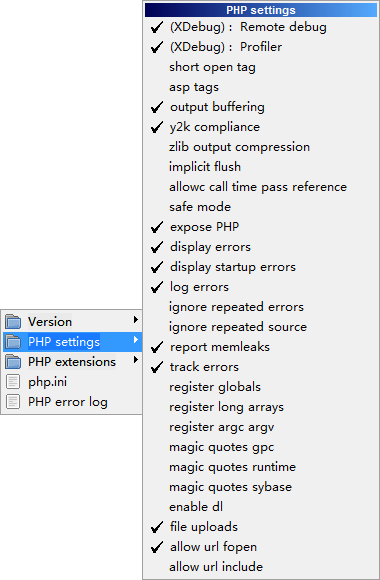
注意：ZF2不向下兼容ZF1，因为在框架中应用了PHP5.3及以后版本的新特性，同时重写了很多组件。

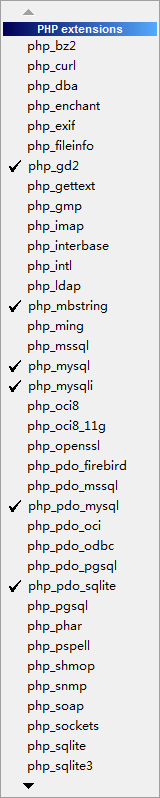
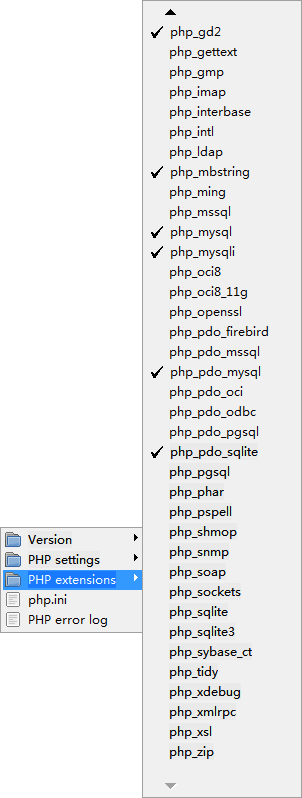
英文帮助：<http://framework.zend.com/manual/2.0/en/index.html>

中文翻译：<http://blog.csdn.net/hunkxia/article/details/8151310>

1. WAMP2.0的安装及设置
2. Apache



1. PHP



1. MySQL

修改root用户的密码，如’1234’，注意使用password算法；

1. phpMyAdmin设置

wamp\apps\phpmyadmin3.5.1\config.inc.php

$cfg['Servers'][$i]['password'] = '1234';

wamp\apps\phpmyadmin3.5.1\libraries\config.default.php

$cfg['Servers'][$i]['password'] = '1234';

1. 下载ZendSkeleonApplication项目

从<https://github.com/zendframework/ZendSkeletonApplication>下载zip包

ZendSkeletonApplication-master.zip

解压缩到wamp的www目录下，根目录改为zendapp或其他名称。

三、ZF2的安装

1. 下载ZF2.05。

方式一：直接下载Zip或tgz包：<https://packages.zendframework.com/>

方式二：使用Git客户端下载：https://github.com/zendframework/zf2

1. 配置ZF2（**推荐方式四**）

方式一：在系统的环境变量中加入ZF2\_PATH，

值为C:/wamp/ZendFramework-2.0.5/library（该路径为ZF2解压后library的路径，注意后面不加/）

此方式对应project中init\_autoloader.php中的getenv('ZF2\_PATH')分支。

方式二：在httpd.conf（在apache的conf目录下）或.htaccess（在project的根目录）文件中添加一行：SetEnv ZF2\_PATH C:/wamp/ZendFramework-2.0.5/library（该路径为ZF2解压后library的路径，注意后面不加/）

此方式对应project中init\_autoloader.php中的getenv('ZF2\_PATH')分支。

方式三：在php.ini文件（apache的bin目录下）中加入

zf2\_path = "C:\wamp\ZendFramework-2.0.5\library"

此方式对应project中init\_autoloader.php中的get\_cfg\_var('zf2\_path')分支。

方式四：将ZF2解压缩后的目录（bin、demos、library、resources、vendor等一层的目录及文件）复制到项目的vendor/ZF2/路径下。

此方式对应project中init\_autoloader.php中的is\_dir('vendor/ZF2/library')分支。

方式五：用PHP的依赖管理工具自动安装。

用cmd到当前项目路径下（该路径下有一个composer.phar文件），依次执行以下命令：

php composer.phar self-update

php composer.phar install

执行后显示如下信息：

Loading composer repositories with package information

Installing dependencies from lock file

- Installing zendframework/zendframework (2.0.5)

Loading from cache

……

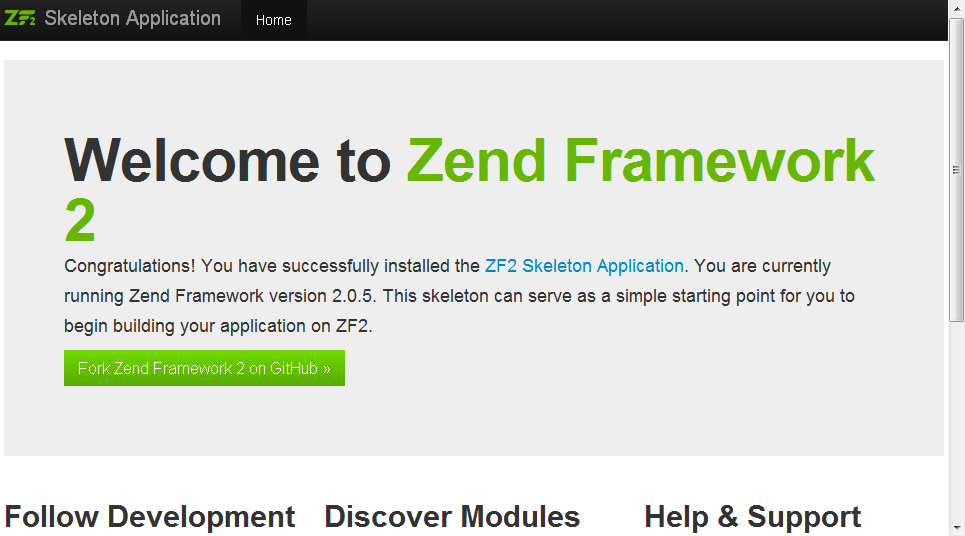
Generating autoload files

此方式为自动加载方式，执行后在project的vender目录中加入了zendframework\zendframework\（可以兼容当前项目的ZF最新版本的代码），\bin\和\composer\（配置文件）。

1. 验证是否配置成功

建Alias目录zendapp（或建立虚拟主机，**建议建虚拟主机，直接把httpd.conf中的**

**DocumentRoot改为"c:/wamp/www/zendapp/public/"，就可以用<http://localhost/>直接访问，与帮助文档中的方式一致。**），对应目录"c:/wamp/www/zendapp/public/"，浏览器输入http://localhost/zendapp/， 出现以下页面，证明ZF2配置成功。



1. 安装 PEAR、PHPUnit 及 phpDoc2（后面会用到PHPUnit测试）

<http://yp.oss.org.cn/blog/show_resource.php?resource_id=1522>

PHPUnit的使用参考Zend Framework的Unit Testing一节相关的内容。

PHPUnit的断言使用说明参考PHPUnit断言详解.doc。

运行unittest：

进入Cmd，将当前路径切换到c:wamp\www\zendapp\module\Application\test\，执行phpunit命令，出现以下结果证明测试通过，否则未通过。

Time: 1 second, Memory: 5.00Mb

OK (1 test, 1 assertion)

1. Album实例详解。

下面将在ZendSkeleonApplication基础上建立Album程序应用实例，一步一步告诉你如何在ZF2上建立应用。

1. 建立如下目录结构

zendapp/ ——————根目录

/module

/Album

/config

/src —————————存放控制层和模型层的PHP文件

/Album ———————用于区分不同的namespaces

/Controller

/Form

/Model

/view ——————————存放表示层代码

/album

/album ——————存放phtml文件

1. 为了加载和配置一个模块，ZF2提供了一个ModuleManager。它将在模块的根目录（module/Album）中寻找Module.php文件，并从该文件中找到一个Album/Module类。也就是说，模块中的类将拥有以模块的名字（模块的目录名）命名的namesapce。

在Album模块中创建Module.php文件，即zendapp/module/Album/Module.php，代码如下：

<?php

**namespace** Album;

**class** **Module**

{

**public** **function** getAutoloaderConfig()

{

**return** **array**(

'Zend\Loader\ClassMapAutoloader' => **array**(

\_\_DIR\_\_ . '/autoload\_classmap.php',

),

'Zend\Loader\StandardAutoloader' => **array**(

'namespaces' => **array**(

\_\_NAMESPACE\_\_ => \_\_DIR\_\_ . '/src/' . \_\_NAMESPACE\_\_,

),

),

);

}

**public** **function** getConfig()

{

**return** **include** \_\_DIR\_\_ . '/config/module.config.php';

}

}

ModuleManager将会为我们自动调用getAutoloaderConfig（）和getConfig（）方法。

1. 文件的自动加载

getAutoloaderConfig()方法返回一个和ZF2的AutoloaderFactory相符的array，我们进行如下配置：向ClassMapAutoloader添加一个class map文件，并将该模块的namespace添加到StandardAutoloader上。StandardAutoloader需要一个namespace和为该namesapce找到文件的路径。这就是PSR-0规则，按照此规则类直接映射到文件。

由于我们在开发过程中不需要通过classmap加载文件，我们提供给class autoloader一个空的array。在zendapp/module/Album下面创建一个autoload\_classmap.php文件。

<?php

**return** **array**();

当autoloader在Album命名空间中找到一个类后，就会将其返回。

注意：不能用getAutoloaderConfig方法代替将module/Application/src添加到composer.json的psr-0 key，否则需要运行php composer.phar update来更新composer自动加载的文件。（此段不理解）

1. 配置

Album\Module中的getConfig()方法加载了config/module.config.php文件，在zendapp/module/Album/config路径下创建module.config.php文件，代码如下：

<?php

**return** **array**(

'controllers' => **array**( //module提供的所有controllers的列表

'invokables' => **array**(

'Album\Controller\Album' => 'Album\Controller\AlbumController',

), // key必须唯一，所以加上module名作为前缀

),

'view\_manager' => **array**( //view/路径下的表示层代码

'template\_path\_stack' => **array**(

'album' => \_\_DIR\_\_ . '/../view',

),

),

);

1. 将新的模块信息通知给application模块

更新application下的config/application.config.php，将Album模块包含进入。代码如下：

<?php

**return** **array**(

'modules' => **array**(

'Application',

'Album', *// <-- 添加此行*

),

'module\_listener\_options' => **array**(

'config\_glob\_paths' => **array**(

'config/autoload/{,\*.}{global,local}.php',

),

'module\_paths' => **array**(

'./module',

'./vendor',

),

),

);

1. 加入Album的程序代码
2. 程序结构

该程序包含以下页面：

| **Page** | **Description** |
| --- | --- |
| Home | 显示albums列表，提供编辑和删除的链接和添加新album的链接。 |
| Add new album | 提供添加新album的表单 |
| Edit album | 提供编辑album的表单 |
| Delete album | 确认删除album，然后删除 |

每个页面可以看成一个action，actions构成模块的controllers。因此，一般将相关的actions放到到一个controller，例如新闻controller可能包含当前、归档和查看的actions。

我们将4个页面的action放到AlbumController中：

| **Page** | **Controller** | **Action** |
| --- | --- | --- |
| Home | AlbumController | index |
| Add new album | AlbumController | add |
| Edit album | AlbumController | edit |
| Delete album | AlbumController | delete |

URL到实际动作的映射在module的module.config.php文件中定义，需要为album的动作添加入口，参考下面代码的高亮部分。

s<?php

**return** **array**(

'controllers' => **array**(

'invokables' => **array**(

'Album\Controller\Album' => 'Album\Controller\AlbumController',

),

),

*// The following section is new and should be added to your file*

'router' => **array**(

'routes' => **array**(

'album' => **array**(

'type' => 'segment',

'options' => **array**(

'route' => '/album[/:action][/:id]',

'constraints' => **array**(

'action' => '[a-zA-Z][a-zA-Z0-9\_-]\*',

'id' => '[0-9]+',

),

'defaults' => **array**(

'controller' => 'Album\Controller\Album',

'action' => 'index',

),

),

),

),

),

'view\_manager' => **array**(

'template\_path\_stack' => **array**(

'album' => \_\_DIR\_\_ . '/../view',

),

),

);

代码说明：入口的name为album（**如果设置的虚拟目录zendapp，则改为zendapp/album**），类型segment。Segment route允许在URL模式（route）中指定占位符（placehoders），在匹配的入口中占位符将被映射成命名的参数。在上述代码中，入口” **/album[/:action][/:id]**”将会匹配所有以/album开头的URL，接下来的部分是一个可选的action名，最后的一部分将会被映射到一个可选想id。（方括号表示该部分为可选项）

Constraints section（正则表达式部分）用来限定segment中期望的字符集，我们限定actions以一个字母开头，随后的字符可以是字母、数字、\_和-，同时限定id为一个或多个数字。所以，该入口允许以下URLs：

| **URL** | **Page** | **Action** |
| --- | --- | --- |
| /album | Home (list of albums) | index |
| /album/add | Add new album | add |
| /album/edit/2 | Edit album with an id of 2 | edit |
| /album/delete/4 | Delete album with an id of 4 | delete |

1. 创建controller

在ZF2中，controller一般命名为{Controller name}Controller，其中{Controller name}必须以大写字母开头。该类定义在{Controller name}Controller.php文件中，存放到模块的Controller目录中，我们的程序中是

module/Album/src/Album/Controller。

在controller类中每一个action是一个public的方法，命名为{action name}Action，{action name}应该以小写字母开头。

注意：以上做法为惯例。ZF2没有提供很多关于controllers的限制规则，除了必须实现Zend\Stdlib\Dispatchable接口。ZF2提供了两个抽象类来帮我们实现这些：Zend\Mvc\Controller\AbstractActionController和

Zend\Mvc\Controller\AbstractActionController，在本程序中将使用AbstractActionController，不过如果你打算写一个RESTful web service（RESTful WebService是比基于SOAP消息的WebService简单的多的一种轻量级Web服务，RESTful WebService是没有状态的，发布和调用都非常的轻松容易。

http://club.topsage.com/thread-473468-1-1.html），AbstractActionController可能会用得到。

在zendapp/ module/Album/src/Album/Controller路径下创建

AlbumController.php，相关代码参考ZF2在线帮助的【Routing and controllers】一节的内容。

我们建立的4个action，每个action对应的URL如下：

| **URL** | **Method called** |
| --- | --- |
| <http://zf2-tutorial.localhost/album> | Album\Controller\AlbumController::indexAction |
| <http://zf2-tutorial.localhost/album/add> | Album\Controller\AlbumController::addAction |
| <http://zf2-tutorial.localhost/album/edit> | Album\Controller\AlbumController::editAction |
| <http://zf2-tutorial.localhost/album/delete> | Album\Controller\AlbumController::deleteAction |

1. 初始化表示层

表示层文件将会被DefaultViewStrategy执行，并且DefaultViewStrategy将会把从controller action方法返回的变量或表示模型传递给表示层。

创建以下4个空文件：

 module/Album/view/album/album/index.phtml

 module/Album/view/album/album/add.phtml

 module/Album/view/album/album/edit.phtml

 module/Album/view/album/album/delete.phtml

可以用PHPUnit测试Controller，具体步骤参考ZF2在线帮助的【Routing and controllers】中【Write the tests】一节的内容。

1. 创建数据库和模型层

创建数据库和数据库表的过程参考ZF2在线帮助的【Database and models】一节的内容。

ZF没有提供一个Zend\Model构件，由于model是业务逻辑并且是由你来决定希望它如何工作的。在此你可以根据需求使用多种构件，方法一是：在程序中用模型类代表每一个实体，然后用映射对象加载和保存实体到数据库；另外一个方法是使用Doctrine或Propel之类的ORM（需要进一步调查）。

在本教程中，我们将创建一个AlbumTable类，它使用了

Zen\DB\TableGeteway\TableGateway类，每个album对象是一个Album 对象（实体）。这是一个实现Table Data Gateway设计模式，以便和数据库表中的数据建立接口。（Table Data Gateway设计模式在大型系统中存在局限性）

很容易将访问数据库的代码放到controller的action方法中，因为这些代码被Zend\Db\TableGateway\AbstractTableGateway暴露出来了，不要这样做。

在module/Album/src/Album/Model目录下创建Album.php，为了和Zend\Db中的TableGateway类一起工作，需要在Album.php中实现exchangeArray方法，该方法只是简单将通过array传过来的数据复制到当前实体的属性中。

详细代码参考ZF2在线帮助的【Database and models】一节的内容。

创建Album.php之后，可以对exchangeArray方法用PHPUnit进行测试，详细参考在线帮助的【Database and models】一节的内容。

下面在module/Album/src/Album/Model目录下创建AlbumTable.php文件，代码参考在线帮助的【Database and models】一节的内容。

在AlbumTable.php中设定protected属性$tableGateway用来接收从constructor传过来的TableGateway的实例，我们将用该属性执行对albums数据库表的操作。还创建了一些程序中将会用到的方法。

1. 用ServiceManager配置table gateway以及注入到AlbumTable中

为了一直使用AlbumTable的同一个实例，我们将用ServiceManager定义如何创建它。最简单的方法是在Module类中创建getServiceConfig方法，此方法将会被ModuleManager自动调用然后应用到ServiceManager。这样我们就能够在需要时在controller中调用它。

为了配置ServiceManager，我们即可以使得这个类成为单例或者作为一个工厂（closure or call back，参考

http://php.net/manual/zh/functions.anonymous.php），当这个ServiceManager需要调用的时候来调用它。我们从实现getServiceConfig()这个方法到提供一个能够创建AlbumTable的工厂类。然后将这个方法添加到Module类的底部。代码参考在线帮助的【Database and models】一节的内容。

注意：代码中匿名函数的参数的定义在

在传给ServiceManager之前，getServiceConfig方法返回一个由ModuleManager合并好的工厂数组。Album\Model\AlbumTable工厂用ServiceManager创建一个AlbumTableGateway然后传递给AlbumTable。同时告诉ServiceManager：通过取得一个Zend\Db\Adapter\Adapter并使用它创建一个TableGateway对象。TableGateway在创建一个新的结果时被告知使用一个Album对象，TableGateway类会使用原型模式创建结果集及实体。这就意味着需要的时候不是实例化，而是系统会克隆之前已经实例化的对象。

最后，我们需要配置ServiceManager以便它能够知道如何取得一个Zend\Db\

Adapter\Adapter。这件事已经由一个可以在合并好的配置系统中进行配置的工厂Zend\Db\Adapter\AdapterServiceFactory完成。ZF2的ModuleManager把来自每个module的module.config.php文件中的配置进行合并，合并到config/autoload(\*.global.php以及\*.local.php文件)目录下的文件中。我们把数据库配置信息添加到global.php文件中，并将该文件提交到版本控制系统中。你可以用local.php（不要放在版本控制系统中）来存储数据库的认证信息。

Config/autoload/global.php的代码如下：

<?php

**return** **array**(

'db' => **array**(

'driver' => 'Pdo',

'dsn' => 'mysql:dbname=zf2tutorial;host=localhost',

'driver\_options' => **array**(

PDO::MYSQL\_ATTR\_INIT\_COMMAND => 'SET NAMES \'UTF8\''

),

),

'service\_manager' => **array**(

'factories' => **array**(

'Zend\Db\Adapter\Adapter'

=> 'Zend\Db\Adapter\AdapterServiceFactory',

),

),

);

数据库认证信息放在Config/autoload/local.php文件中，该文件不在git资源库中。代码如下：

<?php

**return** **array**(

'db' => **array**(

'username' => 'YOUR USERNAME HERE',

'password' => 'YOUR PASSWORD HERE',

),

);

以上代码的测试参考在线帮助的【Database and models】一节的内容。

1. 向controller中添加getAlbumTable方法。

*// module/Album/src/Album/Controller/AlbumController.php:*

**public** **function** getAlbumTable()

{

**if** (!$this->albumTable) {

$sm = $this->getServiceLocator();

$this->albumTable = $sm->get('Album\Model\AlbumTable');

}

**return** $this->albumTable;

}

同时添加属性：protected $albumTable;

测试代码参考在线帮助的【Database and models】一节的内容。

1. 列出Albums的列表

为了列出列表，我们需要将数据检索出来并将数据传给视图。因此，向indexAction方法中加入以下代码。

*// module/Album/src/Album/Controller/AlbumController.php:*

*// ...*

**public** **function** indexAction()

{

**return** **new** ViewModel(**array**(

'**albums**' => $this->getAlbumTable()->fetchAll(),

));

}

*// ...*

在ZF2框架中，为了设置表示层的变量，我们返回一个ViewModel的实例，ViewModel的constructor的第一个参数是包含我们需要的数据的Array。这些变量将自动被传给表示层的代码。ViewModel对象允许我们修改表示层的代码，默认使用{controller name}/{action name}结构。

向index.phtml中添加表示脚本。

<?php

*// module/Album/view/album/album/index.phtml:*

$title = 'My albums';

$this->headTitle($title);

?>

<h1><?php **echo** $this->escapeHtml($title); ?></h1>

<p>

<a href="<?php **echo** $this->url('album', **array**('action'=>'add'));?>">Add new album</a>

</p>

<table class="table">

<tr>

<th>Title</th>

<th>Artist</th>

<th>&nbsp;</th>

</tr>

<?php **foreach** (**$albums** **as** $album) : ?>

<tr>

<td><?php **echo** $this->escapeHtml($album->title);?></td>

<td><?php **echo** $this->escapeHtml($album->artist);?></td>

<td>

<a href="<?php **echo** $this->url('album',

**array**('action'=>'edit', 'id' => $album->id));?>">Edit</a>

<a href="<?php **echo** $this->url('album',

**array**('action'=>'delete', 'id' => $album->id));?>">Delete</a>

</td>

</tr>

<?php **endforeach**; ?>

</table>

<head>部分用headTitle()将页面Title显示到浏览器的Title bar。

url()方法是ZF2提供的用来创建我们需要的链接，其中第一个参数是我们期望的URL的入口名，第二个参数是一个由所有要被填充到占位符的变量组成的数组。在本程序中，我们用album入口，接收两个占位符变量：action和id。

接下来遍历在controller action中制定的**$albums（注意代码中红色标注的地方）**，ZF2自动确保这些变量被提取到了表示层代码的范围，所以我们不必关心在ZF1中需要给变量加$this->前缀，如果想加上也可以。

**注意：我们通常使用escapeHtml()帮助我们避免XSS漏洞。**

至此，我们便可以通过浏览器看到album的列表。

Http://localhost/zendapp/album

1. 布局和多语言翻译

如果想修改SkeletonApplication的Title和copyright信息，需要修改application/language/目录中的\*.po文件，编辑需要用到poedit。

<http://www.poedit.net/download.php>

通过修改Application/config中的module.config.php文件中的translator的local修改当前显示的语言。

版权信息在Application/view/lauout/layout.phtml中。

1. 参考在线帮助的【Database and models】一节的内容给程序添加表单和其他动作。
2. AlbumForm继承Zend\Form\Form，为每一个项目（显示项目、隐藏标签、按钮等）设定变量属性（attributes）和选项（options），包括要显示的标签（label）等；
3. Album实先InputFilterAwareInterface，实现setInputFilter和getInputFilter方法设定表单校验；
4. 在AlbumController中添加addAction方法显示页面和进行提交。

页面重定向：return $this->redirect()->toRoute('album');

页面不跳转：return array('form' => $form);

1. add.phtml

ZF2提供了很多用于显示的简单标签：如this->headTitle(“”)，this->form，this->form()->openTag()，this->form()->closeTag()，this->formRow，this->formHidden()，this->formSubmit()等；还提供了用于显示页面的页面绑定方式this->formCollection($form)；

5）$form->bind($album);可以实现Form的表示和项目的初始化，通过合法性校验后保存到model中；