

# Hướng dẫn sử dụng Zend – Framework

---

Nghiêm Đình Mừng

## Lời nói đầu

Tài liệu này không phải là tài liệu dùng để học cách sử dụng Zend Framework, cũng không phải do mình viết ra. Đây chỉ là những điều mình note lại khi đọc cuốn sách Beginning Zend Framework của tác giả Armando Padilla. Mục đích chính của mình chỉ là note lại một số chú ý để khi sử dụng Zend thì xem lại và làm theo các ví dụ, đây cũng là một cách học khá hay đối với tài liệu nước ngoài. Do là note lại nên có nhiều chỗ dịch không chính xác, nhiều chỗ bỏ qua, cũng có nhiều chỗ không giải thích cụ thể, hoặc có nhưng bổ sung thêm. Vì vậy các bạn nếu dùng tài liệu này chỉ mang tính chất tham khảo và nên đọc kèm sách gốc.

Bắc Ninh 06/02/2014

## Contents

I.	Cài đặt ứng dụng đầu tiên.....	4
II.	Hướng dẫn sử dụng Zend Framework .....	7
1.	Tạo Controller sử dụng Zend Controller .....	9
	Zend_Controller_Router_Route class.....	11
	Request Object.....	14
	Zend Controller Error .....	15
2.	Views, Forms, Filters, and Validators.....	18
	Views.....	18
	Làm việc với biến trong view .....	26
	Vòng lặp trong view .....	30
	Câu lệnh if else .....	32
	Escaping User Input .....	34
	Tạo Forms sử dụng Zend_Form .....	39
	Thêm các phần tử vào form.....	41
	Formatting Form .....	44
	Processing the Form .....	46
	Thêm phần xác nhận và lọc dữ liệu .....	50
	Creating Form Element Objects.....	54
	Tạo Textarea Fields .....	55
	Tạo Password Fields.....	60
	Tạo Radio Buttons.....	60
	Tạo Check Boxes .....	60
	File Uploading .....	65
	Tạo CAPTCHA .....	69
3.	Database Communication, Manipulation, and Display .....	72
	Kết nối với Database .....	72
	Và khi gọi các hàm của Zend_DB với phương thức ví dụ \$this->select().....	74
	Thêm dữ liệu .....	74
	Sử dụng SQL .....	74
	Không sử dụng lệnh SQL .....	76
	Database Expressions .....	78

Escaping Values.....	79
SQL Injection .....	79
Escaping User Data.....	79
Last Inserted ID .....	81
LoundBite Sign-up page .....	82
LoudBite Add Artist.....	84
Hướng đối tượng với câu lệnh SELECT .....	86
Querying Specific Columns .....	87
Thực thi truy vấn.....	88
Creating Column and Table Aliases.....	88
Điều kiện với WHERE .....	89
Truy vấn 2 hay nhiều bảng với câu lệnh JOIN .....	91
Limiting and Order the Result Set.....	93
Database Expressions .....	95
Phân trang.....	97
Sử dụng Zend_Paginator.....	97
Adding Control to the Paginator .....	100
Phân trang bản ghi cơ sở dữ liệu .....	103
<b>Zend Framework: Tương tác cơ sở dữ liệu với Zend_Db_Table .....</b>	<b>104</b>
Zend Framework: Tương tác cơ sở dữ liệu với Zend_Db.....	108
<b>Zend Framework: Hướng dẫn sử dụng layout trong ứng dụng.....</b>	<b>110</b>
<b>Zend Framework: Hướng dẫn cấu hình ứng dụng theo mô hình module.....</b>	<b>112</b>
<b>Zend Framework: Tìm hiểu cơ bản về Zend_Form .....</b>	<b>114</b>

## I. Cài đặt ứng dụng đầu tiên

Sau khi download Zend Framework về các bạn giải nén tất cả các file trong thư mục Zend Framework vào thư mục trong webroot như sau :

APACHE\_HOME/htdocs/helloworld/library/Zend

APACHE\_HOME/htdocs/helloworld/bin

Trong đó helloworld là thư mục chứa website chúng ta tạo trong webroot của apache ở đây là htdoc.

Để tạo được ứng dụng đầu tiên chúng ta có thể cài đặt bằng tay nhưng cách đó khá phức tạp, mình sẽ hướng dẫn các bạn cài đặt ứng dụng bằng Zend\_Tool.

Zend\_Tool là một bộ công cụ cho phép chúng ta sử dụng Zend một cách đơn giản hơn dưới dạng dòng lệnh.

Zend\_Tool hỗ trợ tự động sinh code:

- Project directory structure*
- Controllers*
- Actions*
- Views*
- Bootstrap file*
- Project details*

Để sử dụng Zend\_Tool các bạn vào cmd, mở file zf.bat (đối với windows) hoặc zf.sh (đối với linux) trong thư mục helloworld/bin.

```
C:\Windows\system32\cmd.exe
Volume Serial Number is 4E85-6C99

Directory of C:\xampp\htdocs\helloworld\bin

29/01/2014  08:24  CH      <DIR>          .
29/01/2014  08:24  CH      <DIR>          ..
03/05/2011  12:47  SA      1.258  zf.bat
23/07/2011  06:36  SA     19.616  zf.php
03/05/2011  12:47  SA      1.409  zf.sh
               3 File(s)      22.283 bytes
               2 Dir(s)  41.503.404.032 bytes free

C:\xampp\htdocs\helloworld\bin>zf.bat
An Error Has Occurred
An action and provider is required.

Zend Framework Command Line Console Tool v1.11.11
Usage:
    zf [--global-opts] action-name [--action-opts] provider-name [--provider-opt
s] [provider parameters ...]
    Note: You may use "?" in any place of the above usage string to ask for more
specific help information.
    Example: "zf ? version" will list all available actions for the version prov
ider.
```

Một số lệnh cơ bản:

Để hiển thị Version của zend

```
zf show version
```

Kết quả

```
Zend Framework Version: 1.8.0
```

Để tạo Project đầu tiên , các bạn dùng lệnh sau:

```
zf create project APACHE_HOME/htdocs/helloworld
```



Bây giờ các bạn vào trình duyệt

Sau khi tạo project các bạn vào trình duyệt gõ đường link:

<http://localhost/helloworld/public>

sẽ cho ra kết quả là một trang web và zend đã tạo sẵn cho chúng ta, vậy là ok phần cài đặt.



Như các bạn thấy thư mục public là nơi mà máy chủ sẽ tìm các file khi có yêu cầu từ client. Do vậy tất cả các file như css, js, images... chúng ta nên đặt trong thư mục public.

Nhưng sẽ có một điều bất tiện là mỗi khi truy cập trang web chúng ta lại phải thêm vào sau tên miền /public như vậy không tiện. Nếu chúng ta upload lên host hoặc chỉ có 1 project trên localhost thì chúng ta cấu hình tập tin:

`APACHE_HOME/conf/httpd.conf`

Như sau:

```
DocumentRoot "APACHE_HOME/htdocs/helloworld/public"
```

Set the DirectoryIndex to this:

```
DirectoryIndex index.php
```

Bây giờ các bạn chỉ cần gõ <http://localhost> là xong.

Có 1 cách khác là chúng ta sẽ đưa file .htaccess và index.php từ thư mục public ra ngoài, ngang cấp với thư mục application. Khi đó ta phải sửa lại file index như sau

```
defined('APPLICATION_PATH')
||
define('APPLICATION_PATH',
realpath(dirname(__FILE__) . '/../application'));
```

Thành

```
defined('APPLICATION_PATH')
||
define('APPLICATION_PATH',
realpath(dirname(__FILE__) . '/application'));
```

Và

```
set_include_path(implode(PATH_SEPARATOR, array(
    realpath(APPLICATION_PATH . '/library'),
    get_include_path(),
)));
```

Thành

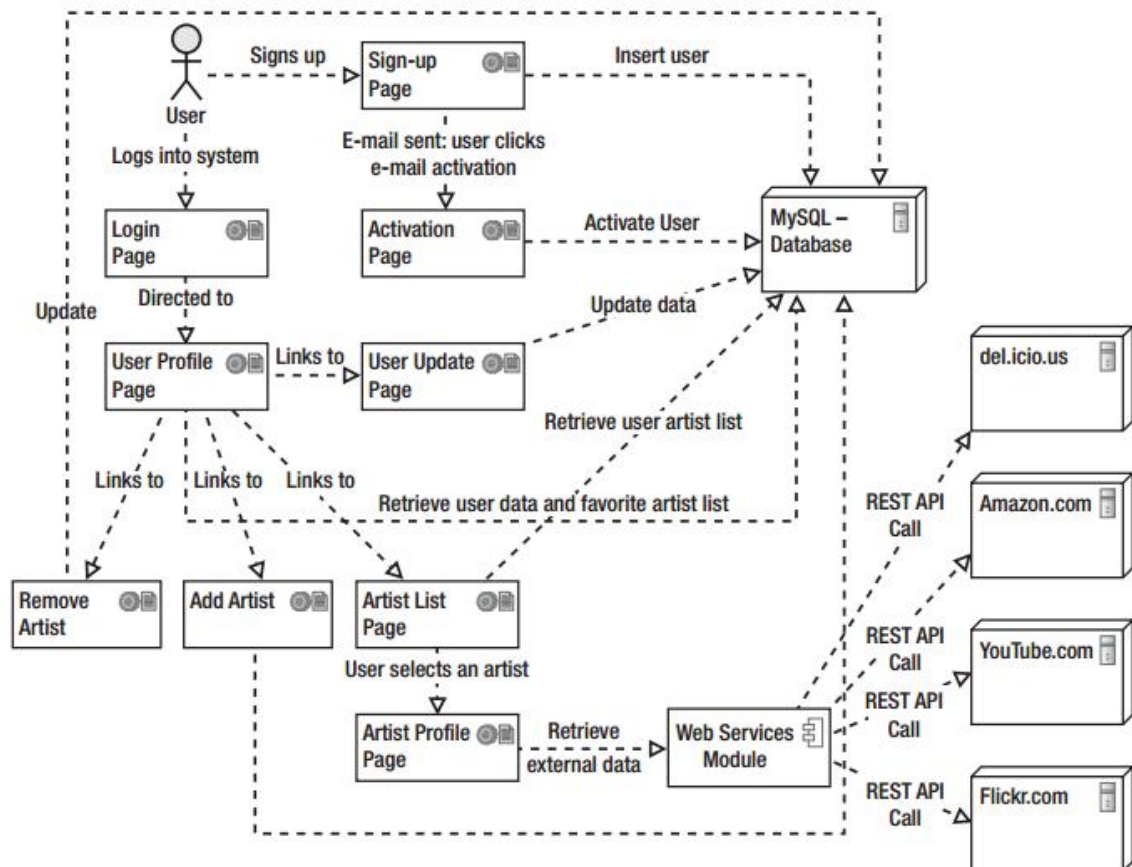
```
set_include_path(implode(PATH_SEPARATOR, array(
    realpath(APPLICATION_PATH . '/../library'),
    get_include_path(),
)));
```

Chúng ta phải sửa lại như vậy là do đường dẫn tới thư mục application và thư mục library đã thay đổi. Bây giờ các bạn chỉ cần vào trình duyệt và gõ <http://localhost/helloworld> là trang web của chúng ta sẽ hiện ra.

## **II. Hướng dẫn sử dụng Zend Framework**

Để minh họa cho việc sử dụng Zend chúng ta sẽ thực hiện với 1 ví dụ về website nghe nhạc mp3.

Website bao gồm 3 modul chính là : Account, Artists và Web services Module



Về phần thiết kế cấu trúc của website mình sẽ không nói nhiều mà sẽ chủ yếu nói về vận dụng Zend trong việc xây dựng website này.

## Tạo cơ sở dữ liệu

```

# Database creation
CREATE DATABASE loudbite;
# Change into the database to run commands
USE loudbite;

# accounts Table Creation
CREATE TABLE accounts (
id int(11) AUTO_INCREMENT PRIMARY KEY,
username varchar(20) NOT NULL UNIQUE,
email varchar(200) NOT NULL UNIQUE,
password varchar(20) NOT NULL,
status varchar(10) DEFAULT 'pending',
email_newsletter_status varchar(3) DEFAULT 'out',
email_type varchar(4) DEFAULT 'text',
email_favorite_artists_status varchar(3) DEFAULT 'out',

```



```
created_date DATETIME
);

# artists Table Creation
CREATE TABLE artists (
id int(11) AUTO_INCREMENT PRIMARY KEY,
artist_name varchar(200) NOT NULL,
genre varchar(100) NOT NULL,
created_date DATETIME
);

# accounts_artists Table Creation
CREATE TABLE accounts_artists (
id int(11) AUTO_INCREMENT PRIMARY KEY,
account_id int(11) NOT NULL,
artist_id int(11) NOT NULL,
created_date DATETIME,
rating int(1),
is_fav int(1)
);
```

## 1. Tạo Controller sử dụng Zend Controller

Để tạo một controller mới các bạn dùng lệnh:

```
zf create controller account
```

Ok, Bây giờ các bạn có thể thấy trong Controller sẽ xuất hiện 1 file AccountController.php và trong Views/Scripts/Account có file index.phtml

Mở file AccountController.php có nội dung:

```
<?php

class AccountController extends Zend_Controller_Action
{
    public function init()
    {
        /* Initialize action controller here */
    }
}
```

```

}

public function indexAction()

{
// action body
}

}

```

Ta có thể thấy 1 controller mới là 1 clas được kết thừa từ lớp Zend\_Controller\_Action và được đặt tên là TênControllerController đó là kỹ thuật lazy loading của Zend. Các bạn có thể tự tạo 1 controller theo quy tắc tương tự.

Trong Class các bạn có thể thấy có 2 function:

Init() là function được gọi đến đầu tiên trong tất cả các action của controller

IndexAction() là một action của Controller. Một cotronller có thể có nhiều action, tên của action cũng được đặt tên theo quy tắc lazyloading.

Để gọi tới 1 action các bạn gõ trên trình duyệt:

<http://localhost/tênContronller/tênAction>

Ví dụ <http://locahost/index/index>

Tất nhiên index controller và index action là những controller và action mặc định lên các bạn chỉ cần gõ <http://localhost> cũng ra trang web như trên.

Để tạo một action mới bằng Zend Tool các bạn gõ dòng lệnh:

```

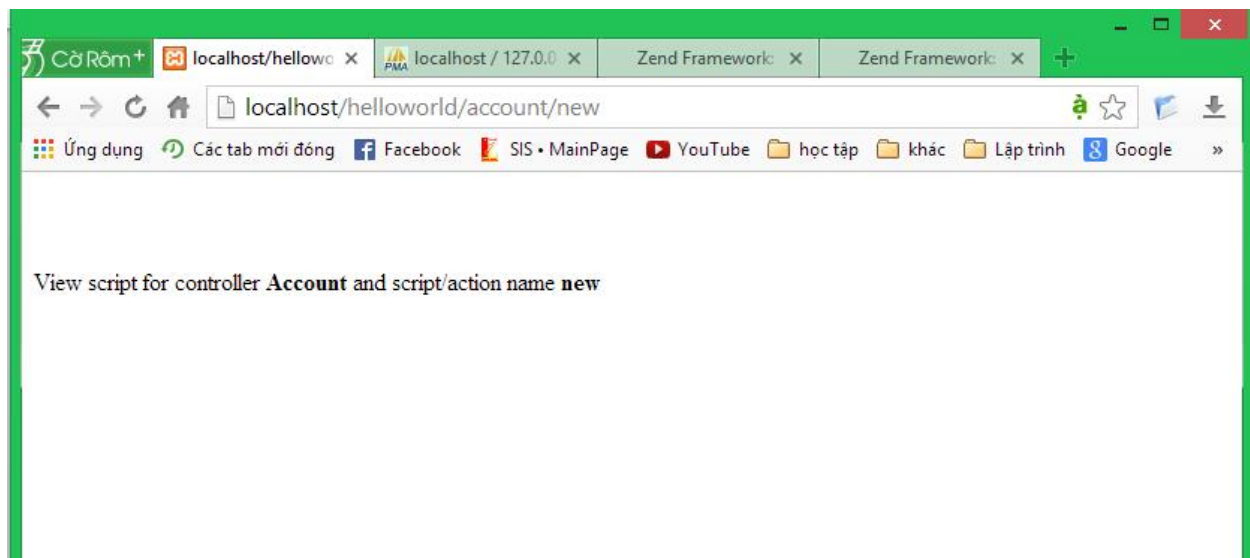
zf create action success account

zf create action new account

zf create action activate account

```

Zend Tool sẽ tự động sinh các action và view tương ứng cho chúng ta:



Một điều lưu ý là với kỹ thuật lazy loading nếu controller của các bạn có tên là `adminUserController` thì khi gọi tới controller này sẽ là `http://localhost/adminuser`

Còn nếu controller là `AdminUserController` thì khi gọi tới controller này sẽ là <http://localhost/admin-user>.

Quy tắc tương tự đối với action.

Như vậy vấn đề đặt ra là muốn thay đổi các đường dẫn này theo ý mình thì phải làm thế nào? Vấn đề này sẽ được giải quyết nhờ :

### **Zend\_Controller\_Router\_Route class**

Để sử dụng Class này các bạn thêm vào `index.php` nội dung sau:

```
/** Routing Info **/
```

```
$FrontController = Zend_Controller_Front::getInstance();
```

```
$Router = $FrontController->getRouter();
```

```
$Router->addRoute("artiststore",
```

```
new Zend_Controller_Router_Route
```

```
(
```

```
"artist/store",
```

```
array
```

```
("controller" => "artist",  
"action" => "artistaffiliatecontent"  
)  
));
```

Như vậy khi các bạn truy cập đường dẫn

<http://localhost/artist/store>

thực chất là đến đường dẫn :

<http://localhost/artist/artistaffiliatecontent>

Còn với đường dẫn kiểu như sau:

<http://localhost/artist/profile?artist=metallica>

thì phải rewrite url như thế nào?

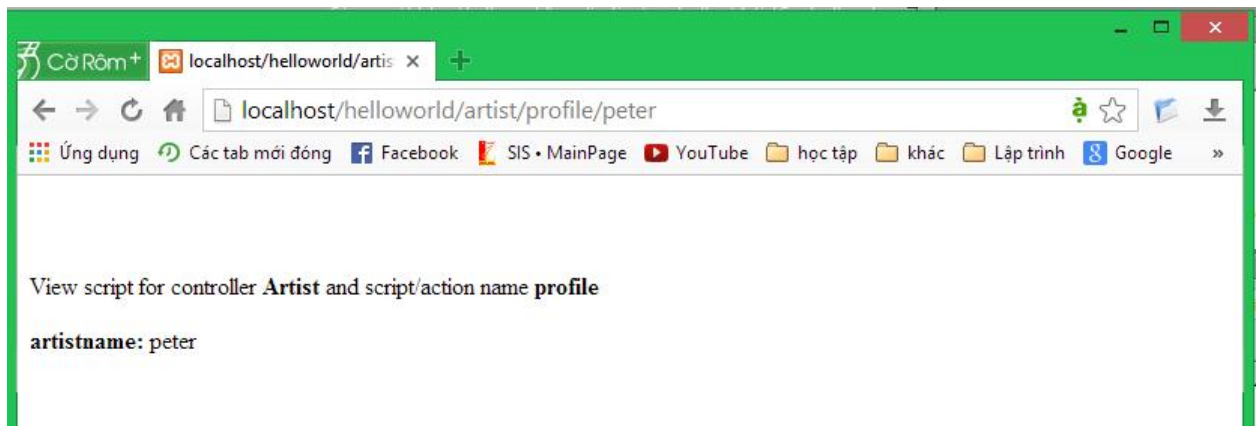
Đầu tiên chúng ta tạo 1 action profile trong controller artist. Sau đó ta thêm vào index.php đoạn code sau:

```
Zend_Controller_Router_Route("artist/:artistname",  
array  
("artistname" => "The Smiths",  
"controller" => "artist",  
"action" => "profile"  
));
```

Trong đó The Smiths là tên artistname mặc định

Trong action profile ta sửa lại như sau:

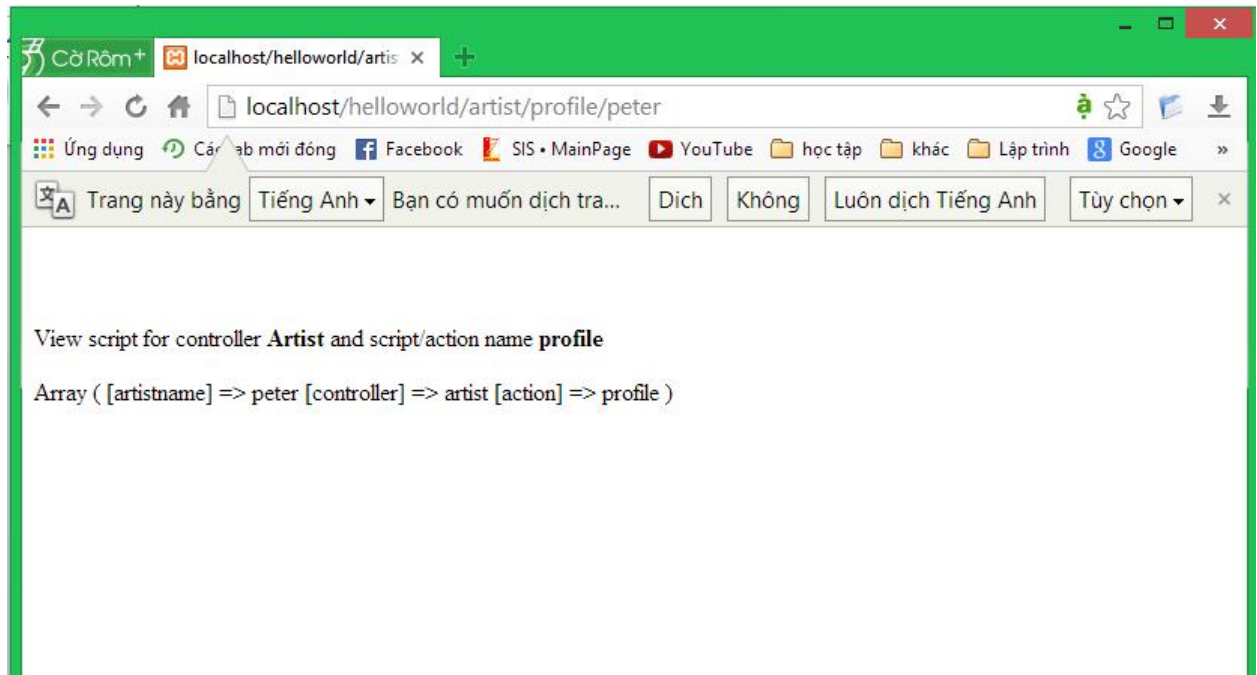
```
public function profileAction()  
{  
    // action body  
    echo "artistname:". $this->_request->getParam('artistname');  
}
```



Ngoài phương thức `getParam` ta còn có phương thức `getParams`

```
public function profileAction()
{
    // action body

    print_r($this->_request->getParams());
}
```



## Request Object

Có 2 loại Request chính là POST và GET.

POST: Dữ liệu được gửi tới từ form

GET: Dữ liệu yêu cầu từ trên đường dẫn ví dụ

<http://localhost/account?u=armando> thì u=armando là get request

Ta sửa nội dung successAction như sau:

```
public function successAction()
{
    $email = $this->_request->getParam('email');
    $username = $this->_request->getParam('username');
    $password = $this->_request->getParam('password');
    //Save the user into the system.
}
```

Có bốn phương thức chính trong Request Object:

```
getParam(String key)
getParams()
getPost(String key)
getQuery(String key)
$this->_request->isPost() //nếu là post request
$this->_request->isGet() //nếu là get request
$this->_request->isXmlHttpRequest() // ajax request object
```

### Zend Controller Error

Zend sẽ gọi tới ErrorController.php mặc định khi có lỗi xảy ra:

```
<?php
class ErrorController extends Zend_Controller_Action {
    public function errorAction(){}
}
```

Để tạo một Controller mới xử lý bắt lỗi ta làm như sau. Thêm một controller mới để xử lý lỗi, ví dụ ApplicationError.

```
<?php
class ApplicationErrorController extends Zend_Controller_Action
{
    public function indexAction()
    {

    }
}
?>
```

Để Zend gọi tới controller này khi có lỗi xảy ra ta phải update lại file index.php

```
$FrontController = Zend_Controller_Front::getInstance();  
$plugin = new Zend_Controller_Plugin_ErrorHandler();  
$plugin->setErrorHandlerController("ApplicationError");  
$plugin->setErrorHandlerAction("index");  
$FrontController->registerPlugin($plugin);
```

Bây giờ sửa lại file index.phtml trong thư mục scripts/application-error

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
  
<html xmlns="http://www.w3.org/1999/xhtml">  
  
<head>  
  
<meta http-equiv="Content-Type" content="text/html;  
charset=utf-8" />  
  
<title>Lỗi</title>  
  
</head>  
  
<body>  
  
<h1>Xin lỗi, đã có lỗi gì đó xảy ra</h1>  
  
</body>  
  
</html>
```

Bây giờ vào trình duyệt gõ 1 đường link lỗi bất kì



Để lấy được exception ta thêm vào indexAction:



//Get the controller's errors.

```
$errors = $this->_getParam('error_handler');
```

```
$exception = $errors->exception;
```

Và truyền ra view

```
$this->view->exception = $exception;
```

Ta sử dụng một số hàm sau để lấy thông tin về ngoại lệ:

getMessage      Returns Zend Framework–created error message.      String

getTrace      Returns a complete flow of the way Zend Framework  
arrived at the error.

Array

getLine      Returns the line where the error was located.      String

getTraceAsString      Returns a formatted string of the trace.      String

getCode      Returns the code for the exception.      String

getFile      Returns the absolute file path that threw the error.  
String

Ví dụ ta thêm vào view dòng lệnh sau:

```
<?php echo $this->exception->getMessage();?>
```



## 2. Views, Forms, Filters, and Validators

### Views

Để truyền dữ liệu từ Controller ra view ta sử dụng phương thức

*\$this->view->tênBiến*

Trong view ta sẽ sử dụng biến này bằng phương thức

*\$this->tênBiến*

Một số phương thức trong View Helper Placeholders

<code>doctype(String)</code>	Creates the <doctype>element for the markup. Acceptable values: XHTML11, XHTML1_STRICT, XHTML1_TRANSITIONAL, XHTML1_FRAMESET, XHTML_BASIC1, HTML4_STRICT, HTML4_LOOSE, HTML4_FRAMESET, HTML5
<code>headLink()</code>	Creates the <link>element within your markup. Available methods: <code>appendStylesheet(\$href, \$media, \$conditionalStyleSheet, \$extras)</code> <code>setStylesheet((\$href, \$media, \$conditionalStyleSheet, \$extras)</code> Example: <code>\$this-&gt;headLink()-&gt;appendStylesheet();</code>
<code>headMeta()</code>	Creates the <meta>element within your markup. Available methods: <code>appendName(\$keyvalue, \$content, \$conditionalName)</code> <code>setName(\$key, \$content, \$conditionalName)</code> <code>setHttpEquiv(\$key, \$content, \$modifiers)</code> Example:

	<code>\$this-&gt;headMeta()-&gt;setHttpEquiv();</code>
<code>headScript()</code>	Creates the <script> element within your markup. Available methods: <code>appendFile(\$src, \$type, \$attributes)</code> <code>setFile(\$src, \$type, \$attributes)</code> <code>appendScript(\$script, \$type, \$attributes)</code> <code>setScript(\$script, \$type, \$attributes)</code>
<code>headStyle()</code>	Creates the <style>element within your markup. Available methods: <code>appendStyle(\$content, \$attributes)</code> <code>setStyle(\$content, \$attributes)</code> Example: <code>\$this-&gt;headStyle()-&gt;setStyle();</code>
<code>headTitle(String)</code>	Creates the <title>element within your markup.

Ta tạo thêm action new trong controller artist

```
/**
 * Add a new artist to the user's profile
 *
 */
public function newAction()
{
    //Get all the genres
    $genres = array("Electronic",
    "Country",
    "Rock",
    "R & B",
    "Hip-Hop",
```

```

    "Heavy-Metal",
    "Alternative Rock",
    "Christian",
    "Jazz",
    "Pop");

//Set the view variables
$this->view->genres = $genres;
}

```

### Tạo file view của newAction

```

<?php echo $this->doctype('XHTML1_STRICT'); ?>

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">

<head>

<?php echo $this->headTitle('LoudBite.com - Add Artist'); ?>

</head>

<body>

<?php echo $this->render('includes/header.phtml') ?>

<h2>Add Artist</h2>

<form method="post" action="save-artist">

<table width="500" border="0" cellpadding="5">

<tr>

<td>Artist Name:</td>

<td><input type="text" name="artistName" /></td>

</tr>

```

```

<tr>

<td>Genre:</td>

<td>

<select name="genre">

<?php foreach ($this->genres as $genre){ ?>

<option value="<?php echo $genre ?>"><?php echo $genre
?></option>

<?php } ?>

</select>

</td> </tr>

<tr>

<td>Add to Favorite List</td>

<td>Yes <input type="radio" value="yes" name="isFav"/> |
No <input type="radio" value="no" name="isFav" /></td>

</tr>

<tr>

<td>Rate:</td>

<td>1 <input type="radio" name="rating" value="1"/> |
2 <input type="radio" name="rating" value="2"/> |
3 <input type="radio" name="rating" value="3"/> |
4 <input type="radio" name="rating" value="4"/> |
5 <input type="radio" name="rating" value="5"/></td>

</tr>

<tr>

<td colspan="2"><input type="submit" value="Add Artist"
/></td>

</tr>

</table>

```

```
</form>
```

```
</body>
```

```
</html>
```

Các bạn có để ý thấy dòng lệnh

```
<?php echo $this->render('includes/header.phtml') ?>
```

Vì header này cố định đối với website nên ta tạo nó ra 1 file riêng để gọi lại trong các view, tiện cho việc sửa đổi và đỡ phải viết code. Các bạn tạo file header.phtml trong thư mục scripts/includes

```
<table width="100%" cellpadding="0" cellspacing="0">
```

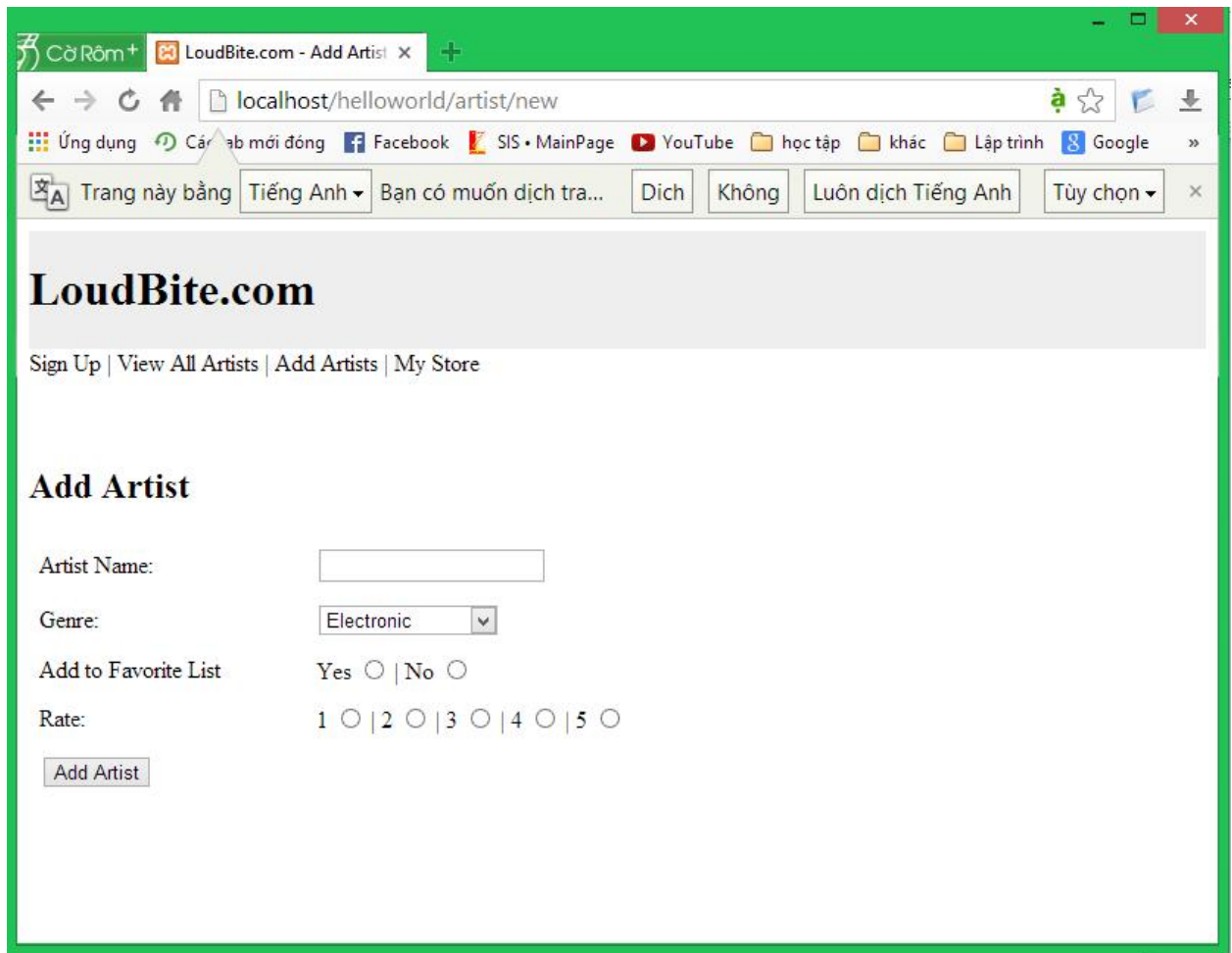
```
<tr
```

```
bgcolor="#eeeeee"><td><h1>LoudBite.com</h1></td></tr>
```

```
<tr><td>Sign Up | View All Artists | Add Artists | My  
Store</td></tr>
```

```
</table>
```

```
<br /><br />
```



Sau khi đã có form để nhập thông tin về artist mới thì cần phải có action saveArtist để lưu thông tin về artist mới này

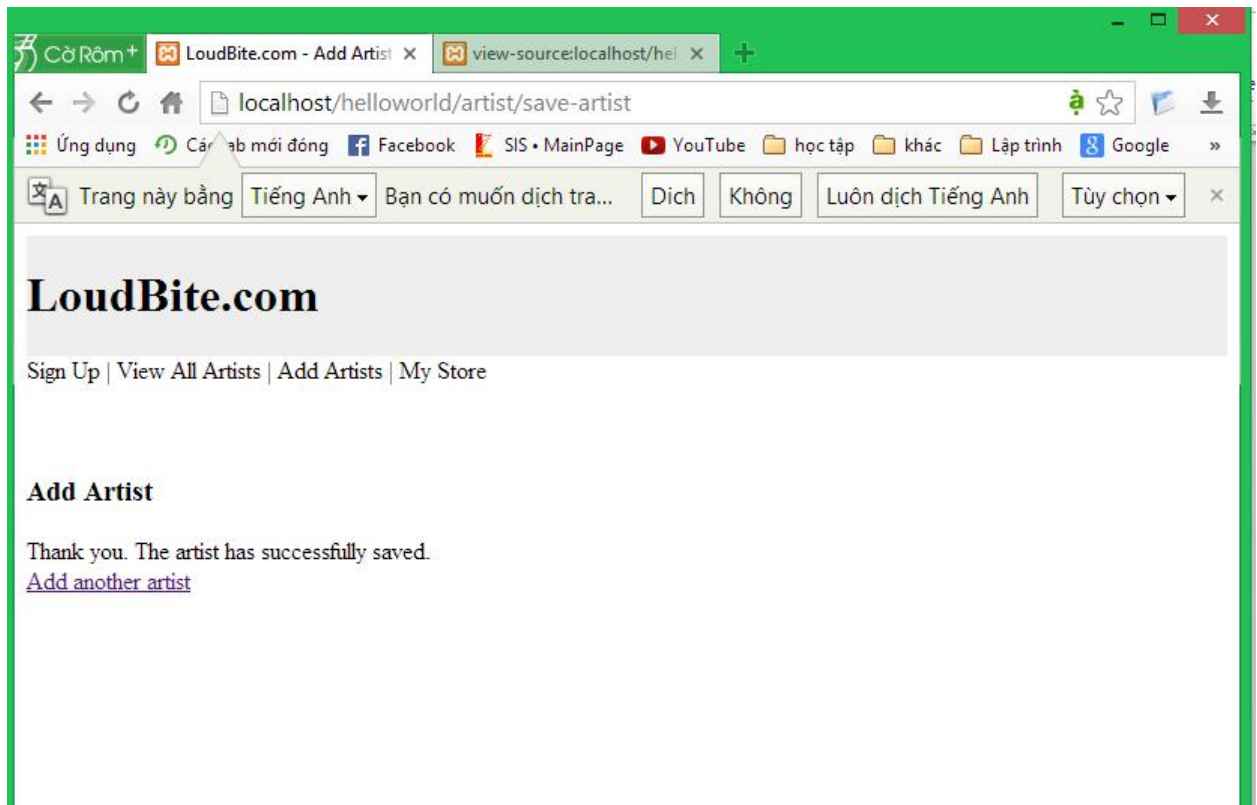
```
/**
 * Save the Artist entered by the user.
 */
public function saveArtistAction() {
    //Initialize variables
    $artistName = $this->_request->getPost('artistName');
    $genre = $this->_request->getPost('genre');
    $rating = $this->_request->getPost('rating');
```

```
$isFav = $this->_request->getPost('isFav');  
  
//Validate  
  
//Save the input into the DB  
  
}
```

Và bây giờ phải thêm view cho action này nữa

```
<?php echo $this->doctype('XHTML1_STRICT'); ?>  
  
<html xmlns="http://www.w3.org/1999/xhtml"  
xml:lang="en" lang="en">  
  
<head>  
  
<?php echo $this->headTitle('LoudBite.com - Add  
Artist'); ?>  
  
</head>  
  
<body>  
  
<?php echo $this->render('includes/header.phtml') ?>  
  
<h3>Add Artist</h3>  
  
<p>  
  
Thank you. The artist has successfully saved.<br/>  
  
<a href='new'>Add another artist</a>  
  
</p>  
  
</body>  
  
</html>
```





Bình thường thì Zend sẽ tự động load view tương ứng với controller nhưng chúng ta cũng có thể hoàn toàn tạo một đối tượng View theo ý của chúng ta. Xét ví dụ tạo thêm 1 action update account

```
/**
 * Update the user's data.
 */
public function updateAction()
{
    //Check if the user is logged in
    //Get the user's id
    //Get the user's information
    //Create the Zend_View object
    $view = new Zend_View();
    //Assign variables if any
```

```

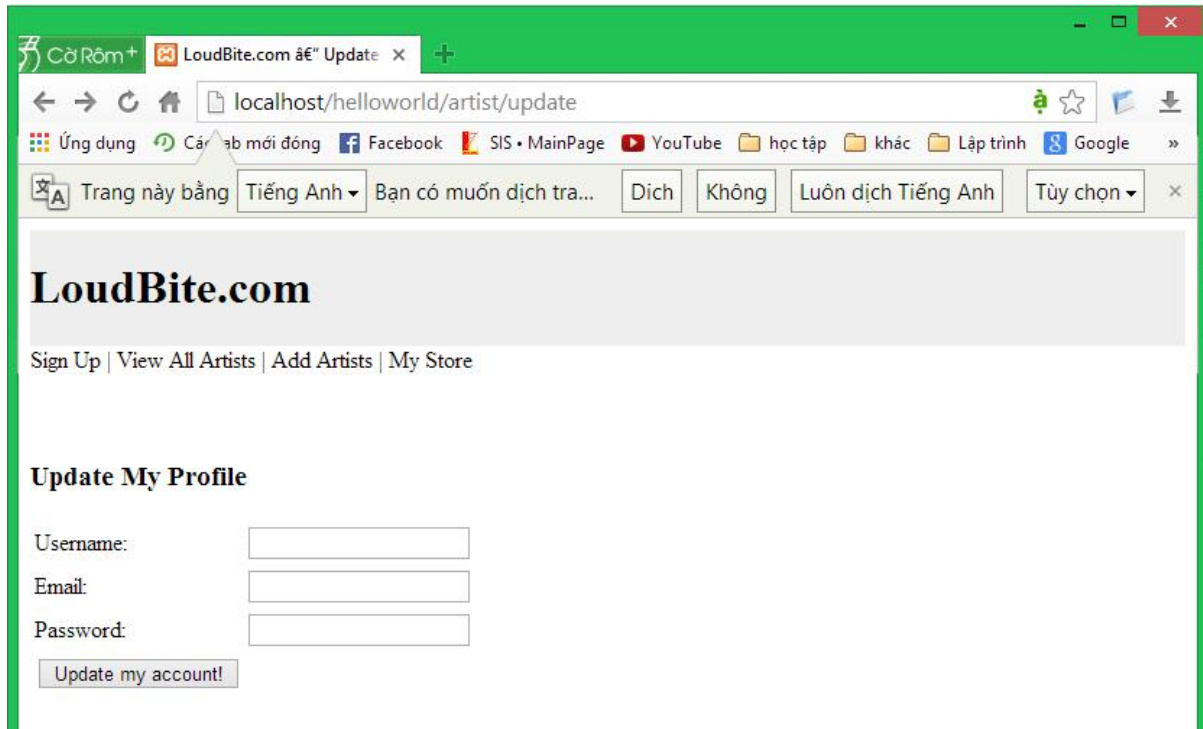
$tmpPath=APPLICATION_PATH.'/views/scripts';

$view->setScriptPath($tmpPath);

$view->render('artist/update.phtml');
}

```

Bây giờ ta tạo file update.phtml trong views/scripts/artist



### Làm việc với biến trong view

Zend\_View có rất nhiều cách để truyền tham số từ controller sang view. Tôi sẽ hướng dẫn tất cả các cách. Nhưng các bạn nên chọn một cách tốt nhất phù hợp với phong cách lập trình của các bạn. Để minh họa cho phần này, chúng ta tạo thêm một action mới trong ArtistController. Action mới cho phép user xóa artist khỏi một danh sách yêu thích nào đó. Chúng ta phải hiển thị 1 danh sách tất cả các artist và hiển thị check box cho tên của từng người, cho phép user xóa artist đó.

Ta tạo một action removeAction và remove.phtml

```

/**
 * Remove favorite artist..
 */

```

```
public function removeAction()
{
    //Check if the user is logged in
    //Get the user's Id
    //Get the user's artists.
    $artists = array("Thievery Corporation",
    "The Eagles",
    "Elton John");
    //Set the view variables
    $this->view->totalArtist = count($artists);
    $this->view->artists = $artists;
}
```

### ***Remove.phtml***

```
<?php echo $this->doctype('XHTML1_STRICT'); ?>
<html>
<head>
<?php echo $this->headTitle('LoudBite.com - Remove
Artist'); ?>
</head>
<body>
<?php echo $this->render('includes/header.phtml') ?>
<h3>Remove Artist </h3>
<table>
<tr>
```

```

<td>Artists - <i>Total Artists (<?php echo $this-
>totalArtist; ?>)</i></td>

</tr>

<tr><td><input type="checkbox" /><?php echo $this-
>artists[1]?></td></tr>

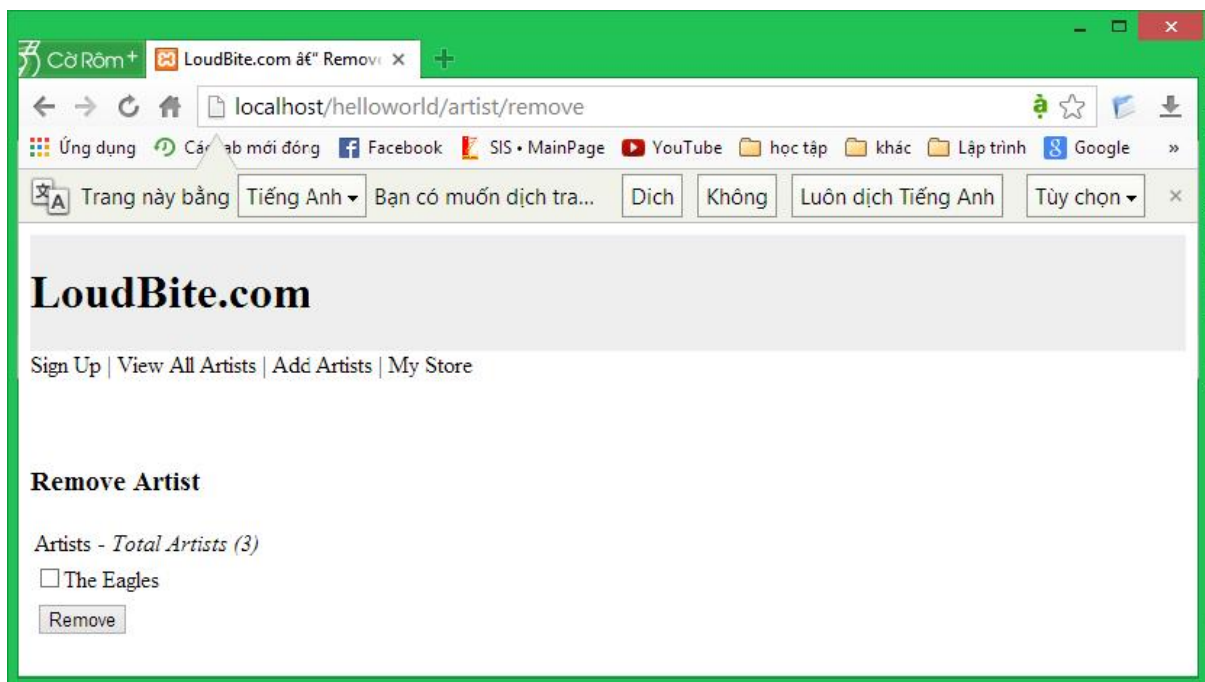
<tr><td><input type="submit"
value="Remove"/></td></tr>

</table>

</body>

</html>

```



Ok cách 1 là ta dùng phương thức `$this->view`. Cách hai chúng ta sẽ dùng hàm `Assign (<mixed value>,<optional value to assign it>)`

Phương thức này cho phép bạn chỉ định rõ tên của biến mà bạn muốn khởi tạo và các giá trị bao gồm.

```

/**
 * Remove favorite artist.
 */

```

```

public function removeAction()
{
    //Check if the user is logged in

    //Get the user's Id

    //Get the user's artists.

    $artists = array("Thievery Corporation", "The Eagles", "Elton
    John");

    //Set the total number of artists in the array.

    //Demonstrates the use of a key-value array assignment.

    $totalNumberOfArtists      =      array("totalArtist"      =>
    count($artists));

    //Set the view variables

    $this->view->assign("artists", $artists);

    $this->view->assign($totalNumberOfArtists);

}

```

Ở cách 2 chúng ta sử dụng phương thức assign của Zend\_View. Còn ở cách thức 3 ta sẽ tạo 1 đối tượng thuộc lớp stdClass và coi nó như 1 thuộc tính.

```

/**
 * Remove favorite artist.
 */

public function removeAction()
{
    //Check if the user is logged in

    //Get the user's Id

    //Get the user's artists.

    $artists = array("Thievery Corporation", "The Eagles", "Elton
    John");

    //Set the total number of artists in the array.

    //Demonstrates the use of a key-value array assignment.

```

```

$totalNumberOfArtists      =      array("totalArtist"      =>
count($artists));

//Create the class

$artistObj = new stdClass();

$artistObj->artists = $artists;

//Set the total number of artists in the array.

//Demonstrates the use of a key-value array assignment.

$totalNumberOfArtists      =      array("totalArtist"      =>
count($artists));

//Set the view variables

$this->view->assign((array)$artistObj);

$this->view->assign($totalNumberOfArtists);

}

```

Cả ba cách trên đều cho ta một kết quả như nhau. Hãy thử cả ba cách và chọn cách mà bạn cảm thấy thích nhất.

### Vòng lặp trong view

Nói chung ta có thể sử dụng các phương thức lặp for, foreach và while tương tự như ở php thông thường.

#### ***remove.phtml***

```

<?php echo $this->doctype('XHTML1_STRICT'); ?>

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">

<head>

<?php echo $this->headTitle('LoudBite.com - Remove Artist');
?>

</head>

<body>

<?php echo $this->render('includes/header.phtml') ?>

<h3>Remove Artist </h3>

```

```

<table>

<tr>

<td>Artists - <i>Total Artists (<?php echo $this->totalArtist;
?>)</i></td>

</tr>

<?php foreach($this->artists as $artist){ ?>

<tr><td><input type="checkbox" value="<?php echo $artist?>"
name="remove[]" /><?php echo $artist?></td>

</tr>

<?php } ?>

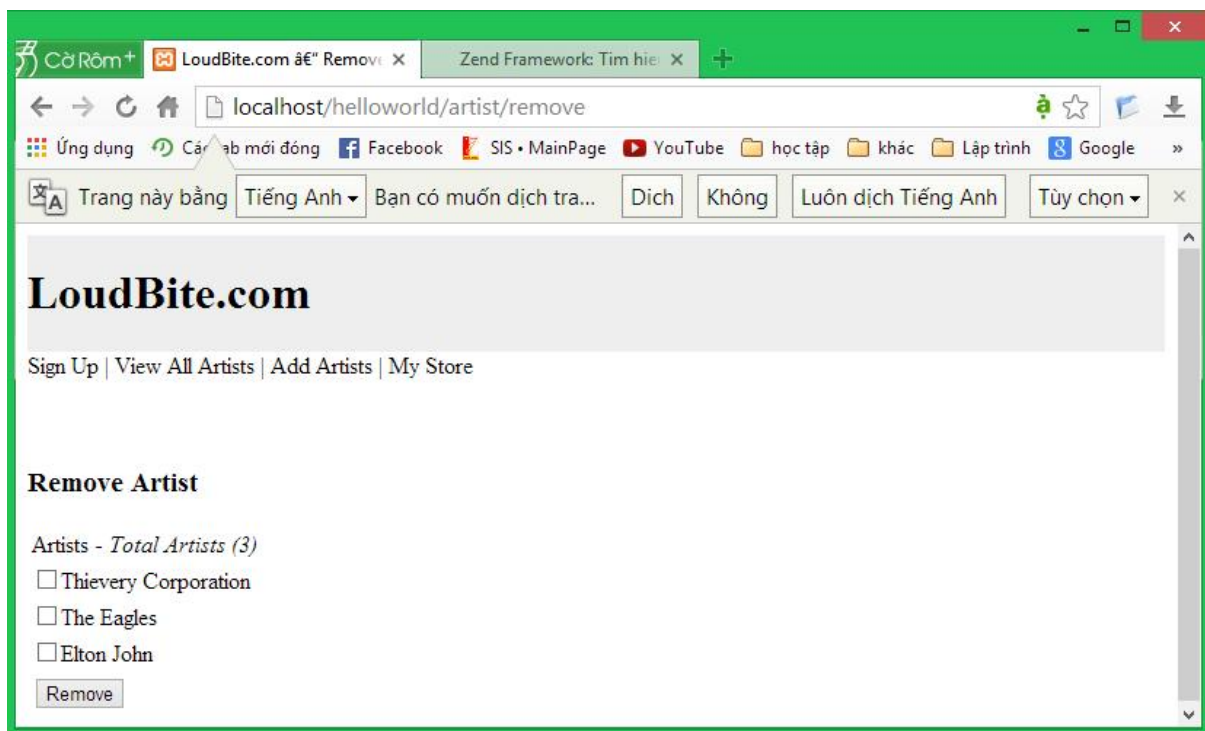
<tr><td><input type="submit" value="Remove"/></td></tr>

</table>

</body>

</html>

```



## Câu lệnh if else

ở phần trên mình đã minh họa các bạn sử dụng vòng lặp trong Zend , ở phần này mình giới thiệu cách dùng if else. Nếu user có rated là 5 ( mức độ yêu thích cao nhất) , thêm một dấu \* sau tên của user.

Listing 4-18. ArtistController.php

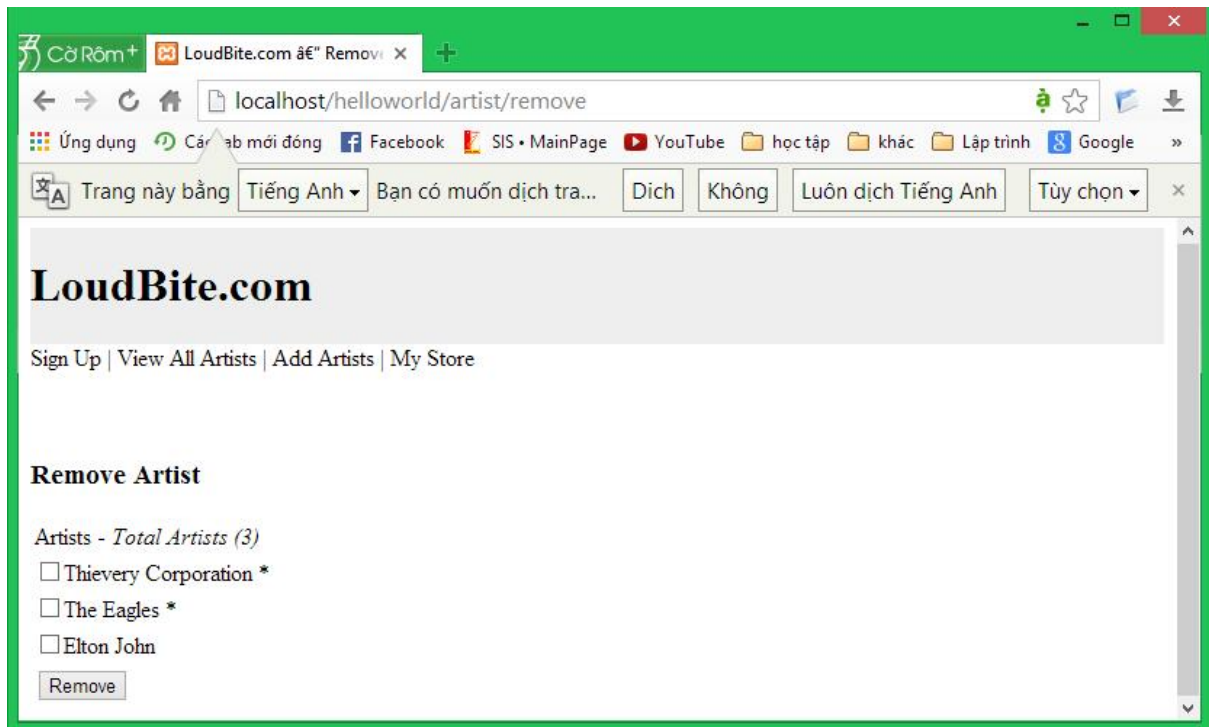
```
/**
 * Remove favorite artist..
 */
public function removeAction()
{
    //Check if the user is logged in
    //Get the user's Id
    //Get the user's artist with rating.
    $artists = array(
        array( "name" => "Thievery Corporation", "rating" => 5),
        array("name" => "The Eagles", "rating" => 5),
        array("name" => "Elton John", "rating" => 4)
    );
    //Create the class
    $artistObj = new stdClass();
    $artistObj->artists = $artists;
    //Set the view variables
    $this->view->assign((array)$artistObj);
    //Set the total number of artists in the array.
    //Demonstrates the use of a key-value array assignment.
    $totalNumberOfArtists = array("totalArtist" =>
        count($artists));
    //Set the view variables
```



```
$this->view->assign((array)$artistObj);  
$this->view->assign($totalNumberOfArtists);  
}
```

**Listing 4-19. remove.phtml**

```
<?php echo $this->doctype('XHTML1_STRICT'); ?>  
  
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"  
lang="en">  
  
<head>  
  
<?php echo $this->headTitle('LoudBite.com - Remove Artist');  
?>  
  
</head>  
  
<body>  
  
<?php echo $this->render('includes/header.phtml') ?>  
  
<h3>Remove Artist </h3>  
  
<table>  
  
<tr>  
  
<td>Artists - <i>Total Artists (<?php echo $this->totalArtist;  
?>)</i></td>  
  
</tr>  
  
<?php foreach($this->artists as $artist){ ?>  
  
<tr><td><input type="checkbox" value="<?php echo  
$artist['name']?>"  
  
name="remove" /><?php echo $artist['name']?>  
  
<?php if($artist['rating'] == 5){ ?> * <?php } ?></td></tr>  
  
<?php } ?>  
  
<tr><td><input type="submit" value="Remove"/></td></tr>  
  
</table>  
  
</body>  
  
</html>
```



## Escaping User Input

Nhiều trường hợp dữ liệu đầu vào của chúng ta có thể chứa các ký tự đặc biệt ví dụ như html tag. Để có thể hiển thị các ký tự này chúng ta cần sử dụng 1 hàm `escape()`.

Hàm `escape()` sẽ sử dụng các hàm cơ bản của php như `htmlspecialchars()` để thay thế các ký tự `<`, `>`, `&`, `"`, và `'` thành các ký tự đặc biệt của html như `&lt;`, `&gt;`, `&quot;`, và `&apos;`.

Ví dụ các bạn chạy đoạn code sau:

```
<tag>PHP& Zend Framework</tag>
```

After passing the string into `htmlspecialchars()`, it would become the following:

```
&lt;tag&gt;PHP &amp; Zend Framework&lt;/tag&gt;
```

Bây giờ các bạn sửa lại file `remove.phtml`

```
<?php echo $this->doctype('XHTML1_STRICT'); ?>
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
```

```
<head>
```

```

<?php echo $this->headTitle('LoudBite.com - Remove Artist');
?>

</head>

<body>

<?php echo $this->render('includes/header.phtml') ?>

<h3>Remove Artist </h3>

<table>

<tr>

<td>Artists - <i>Total Artists (<?php echo $this->totalArtist;
?>)</i></td>

</tr>

<?php foreach($this->artists as $artist){ ?>

<tr>

<td>

<input type="checkbox" value="<?php echo $this-
>escape($artist['name']) ?>"

name="remove" /><?php echo $this->escape($artist['name']) ?>

<?php if($this->escape($artist['rating']) == 5){ ?> * <?php }
?>

</td>

</tr>

<?php } ?>

<tr><td><input type="submit" value="Remove"/></td></tr>

</table>

</body>

</html>

```

**Hàm `escape()` thường dùng khi ta lấy dữ liệu từ form**

```

public function saveArtistAction() {

//Initialize variables

```

```

$artistName = $this->_request->getPost('artistName');
$genre = $this->_request->getPost('genre');
$rating = $this->_request->getPost('rating');
$isFav = $this->_request->getPost('isFav');
//Clean up inputs
$artistName = $this->view->escape($artistName);
$genre = $this->view->escape($genre);
$rating = $this->view->escape($rating);
$isFav = $this->view->escape($isFav);
//Save the input
}

```

Nếu các bạn muốn loại bỏ hẳn các thẻ html có thể dùng thêm lệnh sau:

```

$this->view->setEscape('strip_tags');

```

Nếu hàm `escape()` chưa thực sự tối ưu theo yêu cầu bạn có thể tự tạo 1 hàm `escape` mới như sau.

Các bạn tạo file `Escape.php` trong thư mục `models/utlis`

```

<?php
class Escape {
public function doEnhancedEscape ($string){
$stringToEscape = $string;
//Clean
$stringToEscape = strip_tags($stringToEscape);
$stringToEscape = htmlentities($stringToEscape, ENT_QUOTES,
"UTF-8");
return $stringToEscape;
}
}
?>

```

Để sử dụng được hàm này thì các bạn phải sửa lại file index.php để hàm của chúng ta có thể load vào controller.

```
<?php

// Define path to application directory
defined('APPLICATION_PATH')

|| define('APPLICATION_PATH', realpath(dirname(__FILE__) .
'../application'));

// Define application environment
defined('APPLICATION_ENV')

|| define('APPLICATION_ENV', (getenv('APPLICATION_ENV') ?
getenv('APPLICATION_ENV') : 'production'));

// Ensure library/ is on include_path
set_include_path(implode(PATH_SEPARATOR, array(
realpath(APPLICATION_PATH . '/../library'),
get_include_path(),
)).".";".realpath(APPLICATION_PATH . '/models'));

/** Zend_Application */
require_once 'Zend/Application.php';

// Create application, bootstrap, and run
$application = new Zend_Application(
APPLICATION_ENV,
APPLICATION_PATH . '/configs/application.ini'
);

/** Routing Info */
$FrontController = Zend_Controller_Front::getInstance();
$Router = $FrontController->getRouter();
$Router->addRoute("artiststore",
new Zend_Controller_Router_Route(
```

```

"artist/store",
array
("controller" => "artist",
"action" => "artistaffiliatecontent"
));
$app->bootstrap()
->run();

```

**Và bây giờ thử sử dụng hàm escape() mới**

```

/**
 * Save the artist into the system.
 *
 */
public function saveArtistAction()
{
    //Initialize variables
    $artistName = $this->_request->getPost('artistName');
    $genre = $this->_request->getPost('genre');
    $rating = $this->_request->getPost('rating');
    $isFav = $this->_request->getPost('isFav');
    //Set new escape function to use.
    require "utils/Escape.php";
    $escapeObj = new Escape();
    $this->view->setEscape(array($escapeObj, "doEnhancedEscape"));
    //Clean up inputs
    $artistName = $this->view->escape($artistName);
    $genre = $this->view->escape($genre);
    $rating = $this->view->escape($rating);

```

```

$isFav = $this->view->escape($isFav);

//Save the input

}

```

## Tạo Forms sử dụng Zend\_Form

Một phần quan trọng của một website cho phép kết nối giữa người dùng với hệ thống là form. Sử dụng Zend Framework với Zend\_Form, chúng ta có thể tạo form 1 cách hoàn toàn đơn giản.

**Table 4-2.** *Zend\_Form Setters*

Setter	Description
setAction()	Sets the action attribute in the form tag. <form action='<value>'> Accepts single String value.
setMethod()	Sets the method attribute in the form tag. <form method='<value>'> Accepts single String value. (delete, get, post, put). Default is post.
setName()	Sets the name of the form. Cannot be empty. <form name='<value>'>
setEnctype()	Sets the form encoding type. <form enctype='<value>'> By default, the form encoding is set to application/x-www-form-urlencoded.
setAttrib()	Sets a single form attribute setAttrib(key, value). Can be used to add custom attributes to form tag <form key='<value>'>.
setDecorators()	Sets decorators that govern how the form is rendered. By default, the form is made up of <dl> elements that wrap each input.
setAttribs()	Sets multiple form attribute in one call. setAttribs(array(key, value)) Can be used to add custom attributes to form tag. <form key='<value>'>
setDescription()	Sets the description of the form.

Để hiểu hơn các hàm trong zend\_Form chúng ta xét ví dụ sau:

Listing 4-26. AccountController.php: Updates

```

/**
 * Account Sign Up.
 */

public function newAction() {

//Create Form

```

```

$form = new Zend_Form();

$form->setAction('success');

$form->setMethod('post');

$form->setDescription("sign up form");

$form->setAttrib('sitename', 'loudbite');

//Add the form to the view

$this->view->form = $form;

}

```

Chúng ta tạo 1 form mới bằng 1 đối tượng \$form, và set thuộc tính, action, method và mô tả cho nó. Sau đó truyền form cho view

#### Listing 4-27. new.phtml

```

<?php echo $this->doctype('XHTML1_STRICT'); ?>

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">

<head>

<?php echo $this->headTitle('LoudBite.com - Account Sign up');
?>

</head>

<body>

<?php echo $this->render("includes/header.phtml"); ?>

<h3>Account Sign up</h3>

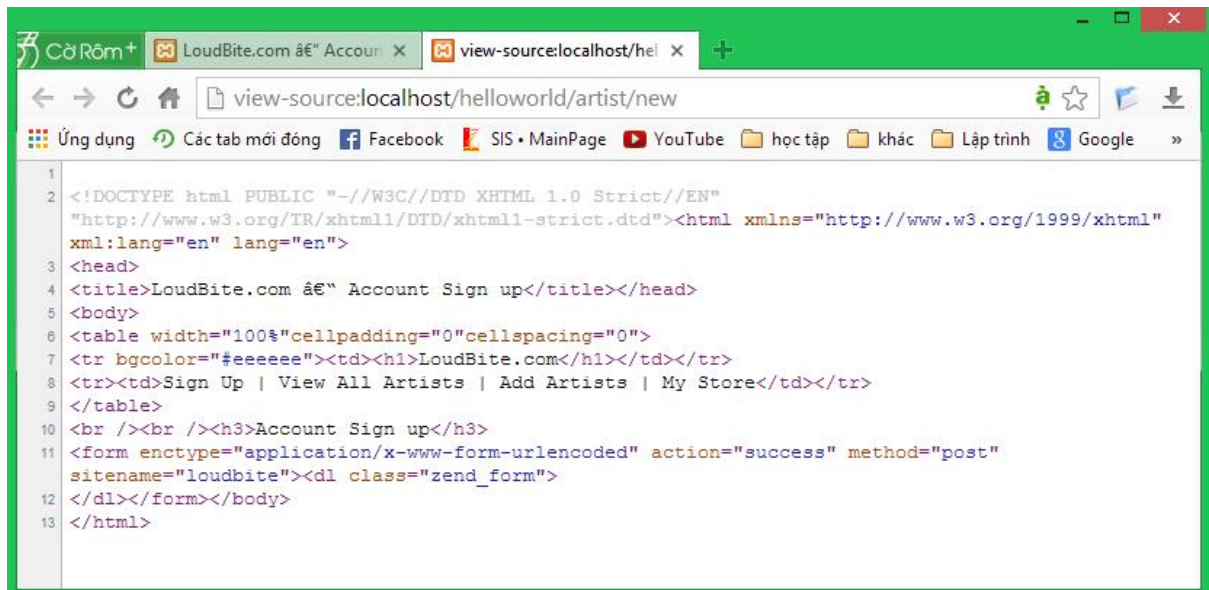
<?php echo $this->form; ?>

</body>

</html>

```





The screenshot shows a web browser window with the address bar displaying 'view-source:localhost/helloworld/artist/new'. The page content is the source code of an HTML document. The code includes a DOCTYPE declaration, a title 'LoudBite.com Account Sign up', and a body containing a table with navigation links and a form for account sign-up. The form has an action of 'success' and a method of 'post'.

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"><html xmlns="http://www.w3.org/1999/xhtml"
3 xml:lang="en" lang="en">
4 <head>
5 <title>LoudBite.com Account Sign up</title></head>
6 <body>
7 <table width="100%" cellpadding="0" cellspacing="0">
8 <tr bgcolor="#eeeeee"><td><h1>LoudBite.com</h1></td></tr>
9 <tr><td>Sign Up | View All Artists | Add Artists | My Store</td></tr>
10 </table>
11 <br /><br /><h3>Account Sign up</h3>
12 <form enctype="application/x-www-form-urlencoded" action="success" method="post"
13 sitename="loudbite"><dl class="zend_form">
```

## Thêm các phần tử vào form

Để sử dụng text fields, menu dọc, check box hoặc các phần tử khác của form, chúng ta phải thêm các phần tử đó vào trong form

Để thêm các phần tử vào form Zend cung cấp cho chúng ta phương thức `addElement()`. Tham số truyền vào có thể là một String hoặc một `Zend_Form_Element` object và String thứ 2 là tên của thuộc tính phần tử và cuối cùng là `Zend_Config` object nếu cần.

**Table 4-3.** Acceptable addElement Values

Value	Element Result	Zend_Form_Element Object
text	Text input field	Zend_Form_Element_Text
password	Password input field	Zend_Form_Element_Password
radio	Radio button	Zend_Form_Element_Radio
hidden	Hidden text field	Zend_Form_Element_Hidden
button	Form button	Zend_Form_Element_Button
checkbox	Check box	Zend_Form_Element_Checkbox
file	File upload field	Zend_Form_Element_File
hash	Hidden hash field that protects from cross site request forgery attacks	Zend_Form_Element_Hash
image	Image input type	Zend_Form_Element_Image
multicheckbox	Multicheck boxes	Zend_Form_Element_Multicheckbox
multiselect	Multiselect menu	Zend_Form_Element_MultiSelect
reset	Reset button	Zend_Form_Element_Reset
select	Select drop-down menu	Zend_Form_Element_Select
submit	Submit button	Zend_Form_Element_Submit
textarea	Text area	Zend_Form_Element_TextArea

Để minh họa ta thêm các phần tử vào form như sau:

Listing 4-29. AccountController.php

```
/**
 * Account Sign Up.
 *
 */
public function newAction(){
    //Create Form
    $form = new Zend_Form();
    $form->setAction('success');
    $form->setMethod('post');
```

```
$form->setDescription("sign up form");

$form->setAttrib('sitename', 'loudbite');

//Add Elements

//Create Username Field.

$form->addElement('text', 'username');

$usernameElement = $form->getElement('username');

$usernameElement->setLabel('Username:');

//Create Email Field.

$form->addElement('text', 'email');

$emailElement = $form->getElement('email');

$emailElement->setLabel('Email:');

//Create Password Field.

$form->addElement('password', 'password');

$passwordElement = $form->getElement('password');

$passwordElement->setLabel('Password:');

$form->addElement('submit', 'submit');

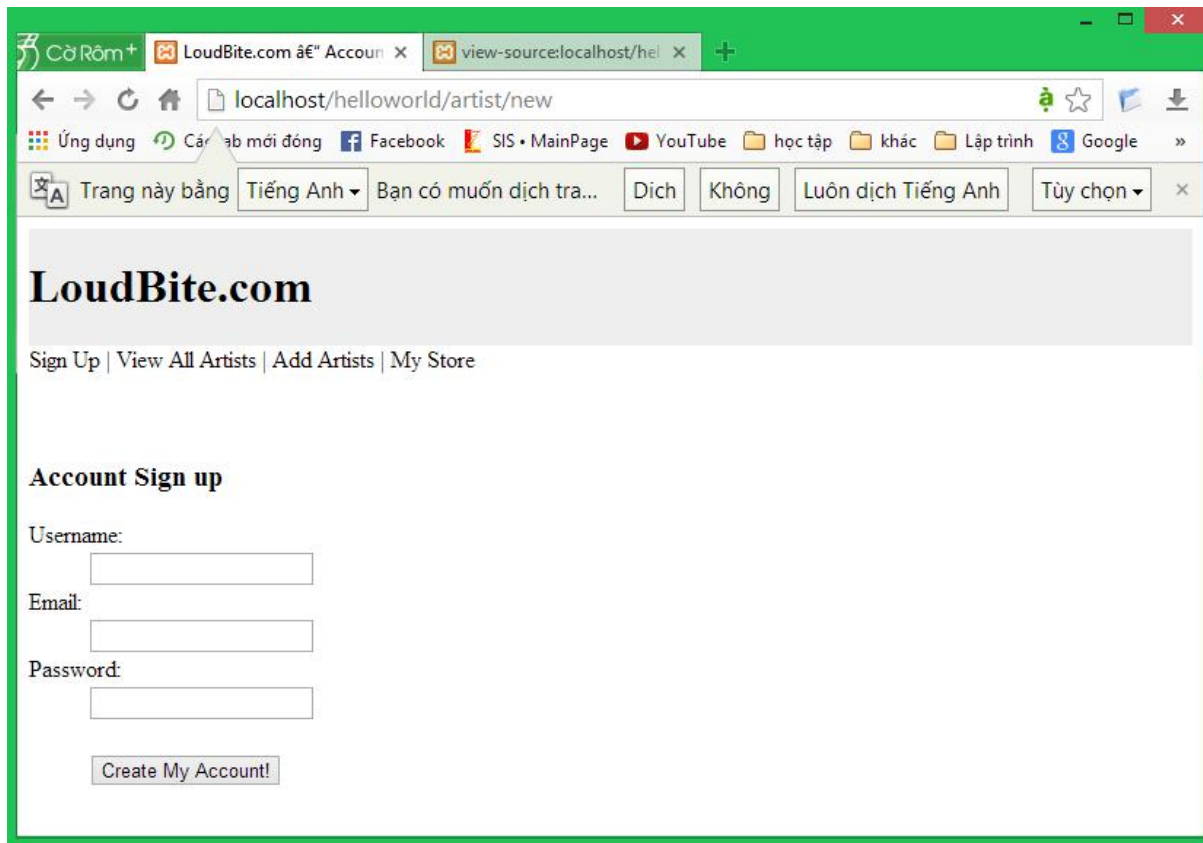
$submitButton = $form->getElement('submit');

$submitButton->setLabel('Create My Account!');

//Add the form to the view

$this->view->form = $form;

}
```



Như chúng ta thấy , khi thêm một phần tử mới, ta có thể dùng hàm `getElement()` để lấy 1 đối tượng `Zend_Form_Element_*` và có thể dùng các hàm với đối tượng này, ví dụ như `setLabel()`

## Formatting Form

Theo mặc định thứ tự của các phần tử sẽ theo thứ tự khởi tạo, để thay đổi thứ tự này chúng ta có thể sử dụng phương thức `Zend_Form_Element` `setOrder` với tham số đầu vào là thứ tự của phần tử:

Listing 4-30. Setting the Order in the AccountController

```
/**
 * Account Sign Up.
 *
 */
public function newAction() {
    //Create Form
    $form = new Zend_Form();
```

```
$form->setAction('success');

$form->setMethod('post');

$form->setDescription("sign up form");

$form->setAttrib('sitename', 'loudbite');

//Add Elements

//Create Username Field.

$form->addElement('text', 'username');

$usernameElement = $form->getElement('username');

$usernameElement->setLabel('Username:');

$usernameElement->setOrder(1);

//Create Email Field.

$form->addElement('text', 'email');

$emailElement = $form->getElement('email');

$emailElement->setLabel('Email:');

$emailElement->setOrder(3);

//Create Password Field.

$form->addElement('password', 'password');

$passwordElement = $form->getElement('password');

$passwordElement->setLabel('Password:');

$passwordElement->setOrder(2);

$form->addElement('submit', 'submit');

$submitButton = $form->getElement('submit');

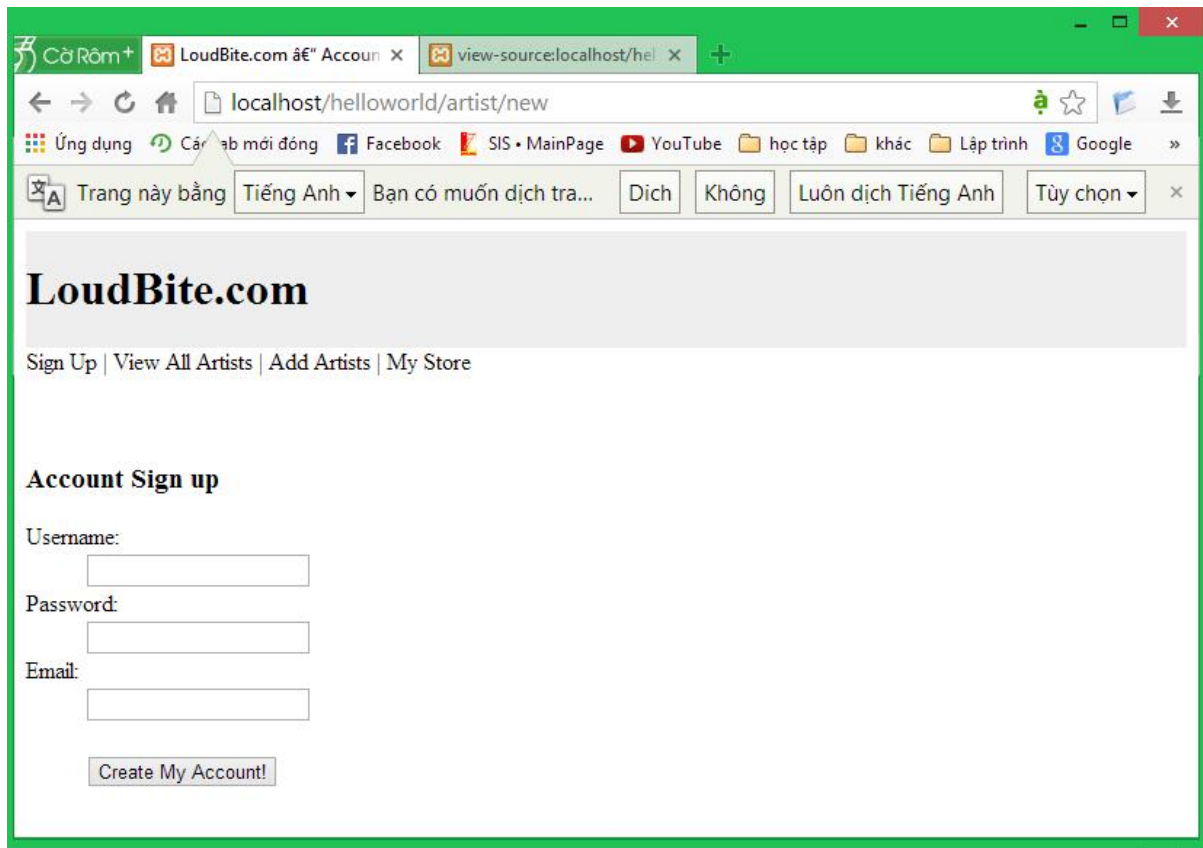
$submitButton->setLabel('Create My Account!');

$submitButton->setOrder(4);

//Add the form to the view

$this->view->form = $form;

}
```



## Processing the Form

Để kiểm tra xem có dữ liệu gửi đến từ form hay không ta sẽ sử dụng hàm `isValid()`. Phương thức này chỉ chấp nhận 2 giá trị `$_POST` or `$_GET`. Để lấy giá trị từ form thì ta sẽ sử dụng phương thức `getValue()` hoặc `getUnfilteredValue()`.

Xét ví dụ:

```
/**
 * Create the sign up form.
 */
private function getSignupForm()
{
    //Create Form
    $form = new Zend_Form();
    $form->setAction('success');
    $form->setMethod('post');
```

```

$form->setAttrib('sitename', 'loudbite');

//Add Elements

//Create Username Field.

$form->addElement('text', 'username');

$usernameElement = $form->getElement('username');

$usernameElement->setLabel('Username:');

$usernameElement->setOrder(1)->setRequired(true);

//Create Email Field.

$form->addElement('text', 'email');

$emailElement = $form->getElement('email');

$emailElement->setLabel('Email:');

$emailElement->setOrder(3)->setRequired(true);

//Create Password Field.

$form->addElement('password', 'password');

$passwordElement = $form->getElement('password');

$passwordElement->setLabel('Password:');

$passwordElement->setOrder(2)->setRequired(true);

$form->addElement('submit', 'submit');

$submitButton = $form->getElement('submit');

$submitButton->setLabel('Create My Account!');

$submitButton->setOrder(4);

return $form;

}

/**
 * Process new account form.
 *
 */

```

```

public function successAction()
{
    $form = $this->getSignupForm();
    //Check if the submitted data is POST type
    if($form->isValid($_POST)){
        $email = $form->getValue("email");
        $username = $form->getValue("username");
        $password = $form->getValue("password");
        //Save the user into the system.
    }else{
        throw new Exception("Whoops. Wrong way of submitting your
        information.");
    }
}

/**
 * Account Sign Up.
 */

public function newAction()
{
    //Get the form.
    $form = $this->getSignupForm();
    //Add the form to the view
    $this->view->form = $form;
}

```

Nếu có lỗi xảy ra chúng ta có thể hiển thị thông tin lỗi bằng cách dùng phương thức `getMessages()` hoặc `getErrors()`

Listing 4-32. AccountController.php: Updates



```

/**
 * Process Sign up Form.
 *
 */
public function successAction()
{
    $form = $this->getSignupForm();

    //Check if the submitted data is POST type
    if($form->isValid($_POST)){
        $email = $form->getValue("email");
        $username = $form->getValue("username");
        $password = $form->getValue("password");
        //Save the user into the system.
    }else{
        $this->view->errors = $form->getMessages();
        $this->view->form = $form;
    }
}

```

Và bây giờ hiển thị lỗi ở view

```

<?php echo $this->doctype('XHTML1_STRICT'); ?>

<html>

<head>

<?php echo $this->headTitle('LoudBite.com - Account Sign up
success'); ?>

</head>

<body>

<?php echo $this->render("includes/header.phtml"); ?>

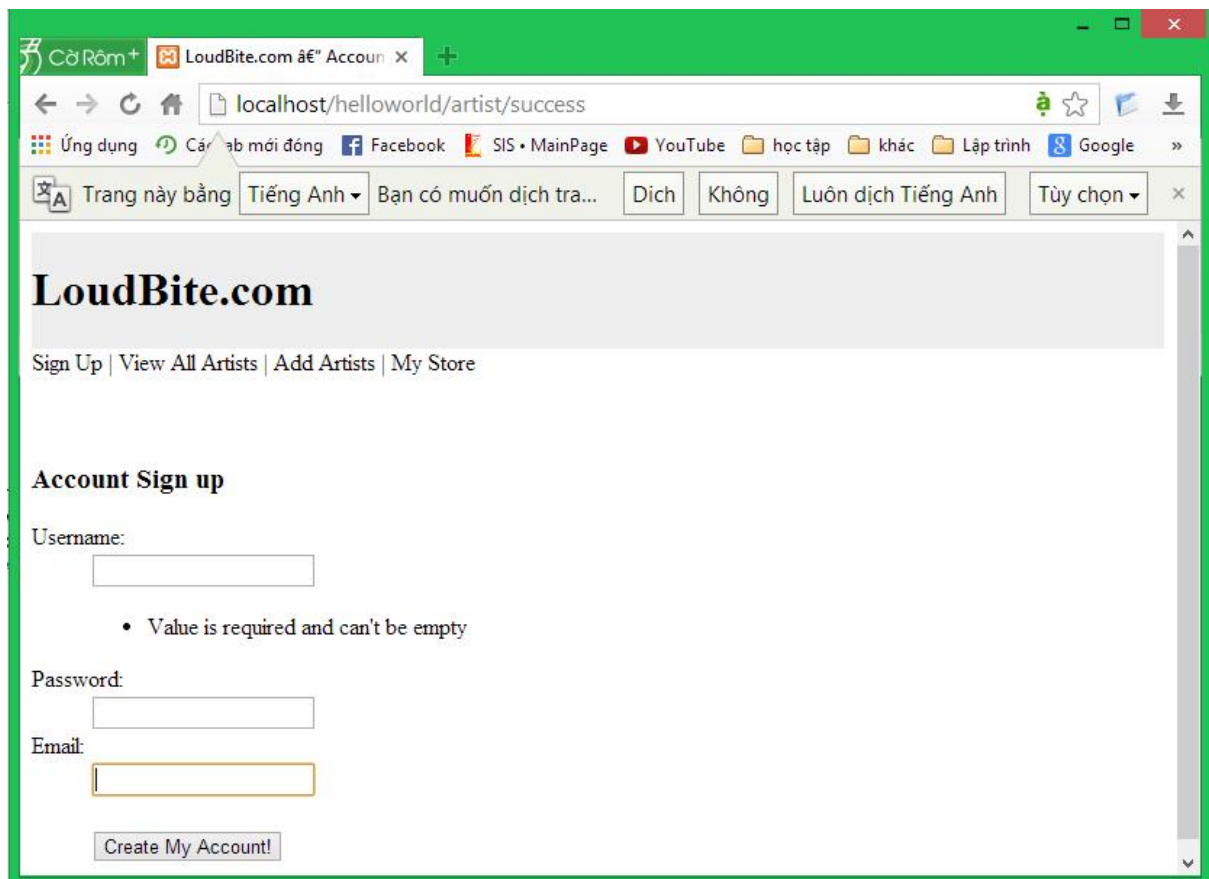
<h3>Account Sign up</h3>

```

```

<?php
if($this->errors){
echo $this->form;
}else{
echo "Thank you! You are one step closer
to becoming a full member.";
}
?>
</body>
</html>

```



## Thêm phần xác nhận và lọc dữ liệu

Làm thế nào để chúng ta kiểm tra xem người dùng có nhập đúng loại dữ liệu yêu cầu hay không? Điều này sẽ được thực hiện đơn giản với Zend\_Vvalidate

Sau đây là bảng Zend\_Validator() object

**Table 4-4. Built-in Validators**

Validator	Description
Zend_Validate_Alnum	Allows only alphanumeric characters.
Zend_Validate_Alpha	Allows only alphabetic characters.
Zend_Validate_Barcode	Validated barcode.
Zend_Validate_Between	Validates ranges: min/max values.
Zend_Validate_Ccnum	Validates entered credit card value to match Luhn algorithm.
Zend_Validate_Date	Allows valid dates in 0000-00-00 format.
Zend_Validate_Digits	Allows only digits.
Zend_Validate_EmailAddress	Allows valid e-mail address formats.
Zend_Validate_Float	Allows only float values.
Zend_Validate_GreaterThan	Allows values greater than a certain value.
Zend_Validate_Hex	Allows only hex values.
Zend_Validate_Hostname	Checks for valid host.
Zend_Validate_Int	Allows only integer values.
Zend_Validate_Ip	Checks for valid IP address.
Zend_Validate_LessThan	Allows the value to be less than the given value.
Zend_Validate_NotEmpty	Checks that the value is not empty.
Zend_Validate_Regex	Checks value against specified regular expression.
Zend_Validate_StringLength	Checks that the string is given between min and max length.

Còn sau đây là bảng Zend\_Filter Object

**Table 4-5.** *Zend\_Filter Objects*

Filter Object	Description
Zend_Filter_Alnum	Returns only alphanumeric characters.
Zend_Filter_Alpha	Returns only alphabetic characters.
Zend_Filter_Basename	Returns the base name of the file.
Zend_Filter_Digits	Returns only the digits in a string.
Zend_Filter_Dir	Returns the directory name.
Zend_Filter_HtmlEntities	Encodes the string and returns the encoded value of the string.
Zend_Filter_Int	Returns only int values.
Zend_Filter_StripNewlines	Removes \n markers from the string.
Zend_Filter_StringToLower	Converts the string to lowercase values.
Zend_Filter_StringToUpper	Converts the string to uppercase values.
Zend_Filter_StringTrim	Strips whitespace from the string.
Zend_Filter_StripTags	Strips HTML tags from the string.

Để hiểu hơn về cách dùng 2 lớp này ta xem ví dụ:

```
/**
 * Create the sign up form.
 */
private function getSignupForm() {
    //Create Form
    $form = new Zend_Form();
    $form->setAction('success');
    $form->setMethod('post');
    $form->setAttrib('sitename', 'loubdbite');
    //Add Elements
    //Create Username Field.
    $form->addElement('text', 'username');
    $usernameElement = $form->getElement('username');
```

```
$usernameElement->setLabel('Username:');

$usernameElement->setOrder(1)->setRequired(true);

//Add validator

$usernameElement->addValidator(
new Zend_Validate_StringLength(6, 20)
);

//Add Filter

$usernameElement->addFilter(new Zend_Filter_StringToLower());
$usernameElement->addFilter(new Zend_Filter_StripTags());

//Create Email Field.

$form->addElement('text', 'email');

$emailElement = $form->getElement('email');

$emailElement->setLabel('Email:');

$emailElement->setOrder(3)->setRequired(true);

//Add Validator

$emailElement->addValidator(new Zend_Validate_EmailAddress());

//Add Filter

$emailElement->addFilter(new Zend_Filter_StripTags());

//Create Password Field.

$form->addElement('password', 'password');

$passwordElement = $form->getElement('password');

$passwordElement->setLabel('Password:');

$passwordElement->setOrder(2)->setRequired(true);

//Add Validator

$passwordElement->addValidator(
new Zend_Validate_StringLength(6,20)
);
```

```
//Add Filter

$passwordElement->addFilter(new Zend_Filter_StripTags());

$form->addElement('submit', 'submit');

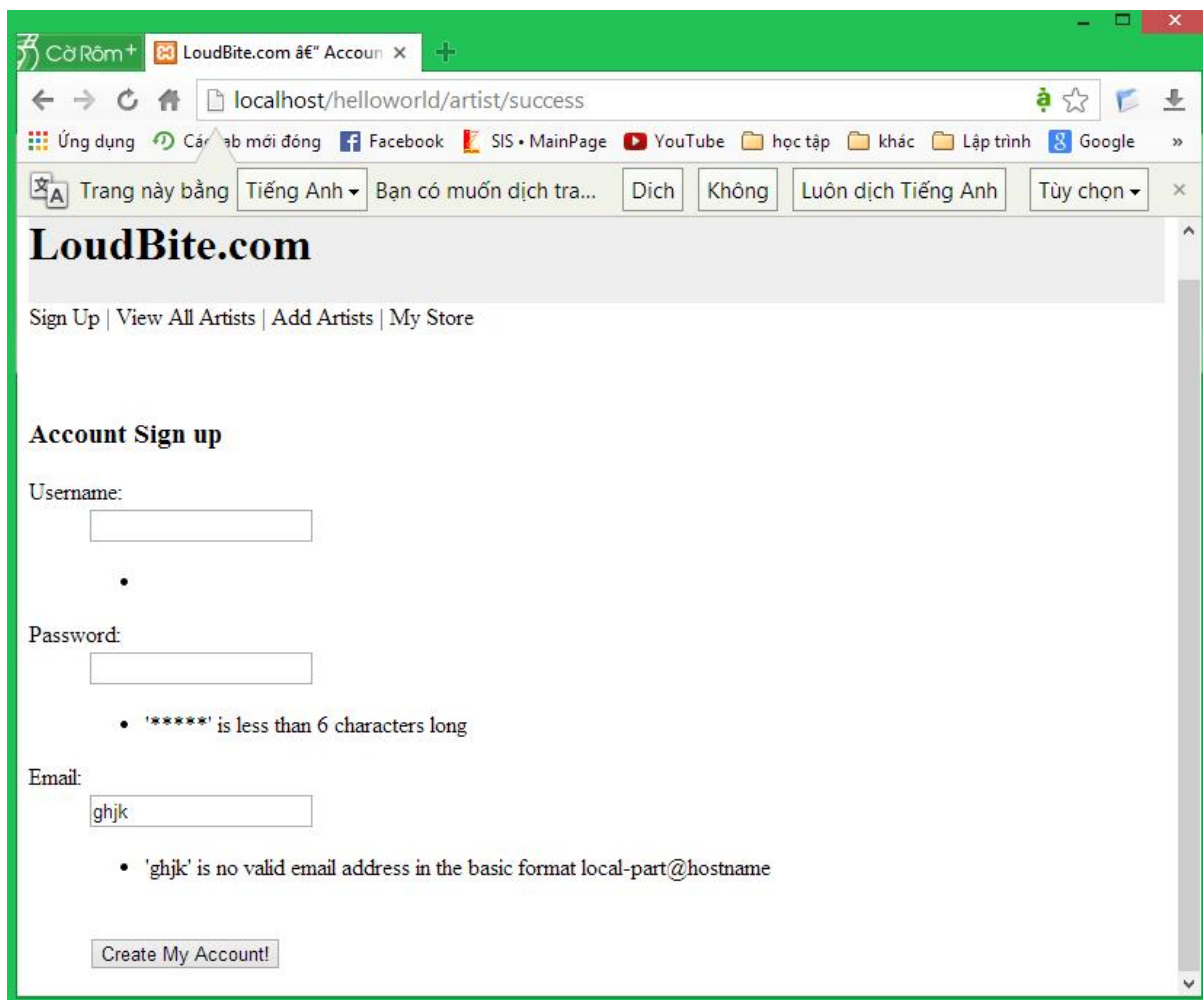
$submitButton = $form->getElement('submit');

$submitButton->setLabel('Create My Account!');

$submitButton->setOrder(4);

return $form;

}
```



## Creating Form Element Objects

Ở phần trước chúng ta đã tạo form trực tiếp, ở phần này chúng ta sẽ tạo form bằng cách thêm các element có sẵn.

Dưới đây là bảng Zend\_Form\_Element Object Setter

**Table 4-6.** *Zend\_Form\_Element* Setters

Setter	Description
setName()	Sets the name of the element.
setLabel()	Sets the label of the element.
setOrder()	Sets the order the element will be shown on the form.
setValue()	Sets the default value that will be shown on the element when the form renders.
setDescription()	Sets the description of the element.
setRequired()	Identifies whether the element is required.
setAllowEmpty()	Identifies whether the element can be empty.
setAutoInsertNotEmptyValidator()	Sets whether the element will be marked as not empty when it is required. If it is marked as not empty, no other validators will run, and the user will be told.
setAttrib()	Sets any attribute on the element.
setAttribs()	Sets multiple attributes for the element.

## Tạo Textarea Fields

Đầu tiên ta tạo trong models thư mục Form

### Listing 4-35. Form/Elements.php class

```
<?php
/**
 * Loudbite.com Form Elements.
 *
 */
class Elements
{
/**
 * Create email text field
 *
 * @returnZend_Form_Element_Text
 */
}
```

```

public function getEmailTextField()
{
    //Create Text Field Object.
    $emailElement = new Zend_Form_Element_Text('email');
    $emailElement->setLabel('Email:');
    $emailElement->setRequired(true);
    //Add Validator
    $emailElement->addValidator(new Zend_Validate_EmailAddress());
    //Add Filter
    $emailElement->addFilter(new Zend_Filter_HtmlEntities());
    $emailElement->addFilter(new Zend_Filter_StripTags());
    return $emailElement;
}

/**
 * Create password text field
 *
 * @return Zend_Form_Element_Password
 */
public function getPasswordTextField()
{
    //Create Password Object.
    $passwordElement = new Zend_Form_Element_Password('password');
    $passwordElement->setLabel('Password:');
    $passwordElement->setRequired(true);
    //Add Validator
    $passwordElement->addValidator
    (
        new Zend_Validate_StringLength(6,20)
    );
}

```



```

//Add Filter

$passwordElement->addFilter(new Zend_Filter_HtmlEntities());

$passwordElement->addFilter(new Zend_Filter_StripTags());

return $passwordElement;

}

/**
 * Create username text field.
 *
 * @return Zend_Form_Element_Text
 */
public function getUsernameTextField()
{
    $usernameElement = new Zend_Form_Element_Text('username');
    $usernameElement->setLabel('Username:');
    $usernameElement->setRequired(true);

    //Add validator
    $usernameElement->addValidator
    (
        new Zend_Validate_StringLength(6, 20)
    );

    //Add Filter
    $usernameElement->addFilter(new Zend_Filter_StripTags());
    $usernameElement->addFilter(new Zend_Filter_HtmlEntities());
    $usernameElement->addFilter(new Zend_Filter_StringToLower());

    return $usernameElement;
}
}

```

**Bây giờ mở file AccountController.php**

\* Update Form

```

*/

private function getUpdateForm(){

//Create Form

$form = new Zend_Form();

$form->setAction('update');

$form->setMethod('post');

$form->setAttrib('sitename', 'loubbite');

//Load Elements class

require "Form/Elements.php";

$LoubbiteElements = new Elements();

//Create Username Field.

$form->addElement($LoubbiteElements->getUsernameTextField());

//Create Email Field.

$form->addElement($LoubbiteElements->getEmailTextField());

//Create Password Field.

$form->addElement($LoubbiteElements->getPasswordTextField());

//Create Text Area for About me.

$textAreaElement = new Zend_Form_Element_TextArea('aboutme');

$textAreaElement->setLabel('About Me:');

$textAreaElement->setAttribs(array('cols' => 15,

'rows' => 5));

$form->addElement($textAreaElement);

//Create a submit button.

$form->addElement('submit', 'submit');

$submitElement = $form->getElement('submit');

$submitElement->setLabel('Update My Account');

return $form;

}

/**

```

```

* Update the User's data.
*
*/
public function updateAction()
{
    //Check if the user is logged in
    //Fetch the user's id
    //Fetch the users information
    //Create the form.
    $form = $this->getUpdateForm();
    //Check if the form has been submitted.
    //If so validate and process.
    if($_POST){
        //Check if the form is valid.
        if($form->isValid($_POST)){
            //Get the values
            $username = $form->getValue('username');
            $password = $form->getValue('password');
            $email = $form->getValue('email');
            $aboutMe = $form->getValue('aboutme');
            //Save.
        }
        //Otherwise redisplay the form.
    }else{
        $this->view->form = $form;
    }
}

//Otherwise display the form.
else{

```

```
$this->view->form = $form;  
  
}  
  
}
```

### Tạo Password Fields

Cách đơn giản nhất là ta dùng `Zend_Form_Element_Password` object. Ta chú ý tới phương thức `setObscure(boolean)` :true nếu ta muốn ẩn dữ liệu và false là ngược lại

### Tạo Radio Buttons

Radio buttons cho phép người dùng lựa chọn một trong tập các giá trị. Chúng ta sử dụng `Zend_Form_Element_Radio` với các phương thức:

*`addMultiOption()` Adds an option and value of the option to display as a radio option.*

*`addMultiOptions()` Adds a set of options and values to display as radio options.*

*`setMultiOptions()` Overwrites existing options.*

### Tạo Check Boxes

Check boxes cho phép người dùng lựa chọn nhiều giá trị từ nhiều lựa chọn. Chúng ta sẽ sử dụng `Zend_Form_Element_Checkbox` hoặc `Zend_Form_Element_MultiCheckBox` object. Đối với `Zend_Form_Element_Checkbox` giá trị mặc định khi bị check là 1 và không bị check là

**Table 4-7.** *Zend\_Form\_Element\_Radio* Setters

Function	Description
<code>addMultiOption()</code>	Adds an option and value of the option to display as a radio option.
<code>addMultiOptions()</code>	Adds a set of options and values to display as radio options.
<code>setMultiOptions()</code>	Overwrites existing options.

## Creating Check Boxes

Check boxes allow users to select multiple values from many options. They are represented as `Zend_Form_Element_Checkbox` objects or `Zend_Form_Element_MultiCheckbox` objects if you want the user to have the option to select two or more values. For the `Zend_Form_Element_Checkbox`, the default value for a checked item is 1, and the default value for an unchecked item is 0.

Table 4-8 shows the key methods of `Zend_Form_Element_Checkbox`.

**Table 4-8.** *Zend\_Form\_Element\_Checkbox* Methods

Method	Description
<code>setCheckedValue()</code>	Sets the value for the checked value.
<code>setUncheckedValue()</code>	Sets the value for the unchecked value.

The `Zend_Form_Element_MultiCheckbox` contains the setters shown in Table 4-9.

**Table 4-9.** *Zend\_Form\_Element\_MultiCheckbox* Setters

Method	Description
<code>addMultiOption()</code>	Adds an option and value of the option to display as a radio option.
<code>addMultiOptions()</code>	Adds a set of options and values of the options to display as radio options.
<code>setMultiOptions()</code>	Overwrites existing options.

## Tạo Select Menus và Multiselect Menus

Menu bình thường cho phép người dùng một lựa chọn, đa menu cho phép người dùng nhiều lựa chọn hơn. Để tạo menu ta dùng `Zend_Form_Element_Select` và `Zend_Form_Element_Multiselect`.

Listing 4-38. Updating `newAction()` with `Zend_Form_Element` Objects

```
/**
 * Create Add Artist Form.
 *
 * @return Zend_Form
 */
private function getAddArtistForm()
```

```

{
$form = new Zend_Form();
$form->setAction("saveartist");
$form->setMethod("post");
$form->setName("addartist");
//Create artist name text field.
$artistNameElement = new Zend_Form_Element_Text('artistName');
$artistNameElement->setLabel("Artist Name:");
//Create genres select menu
$genres = array("multiOptions" => array
(
"electronic" => "Electronic",
"country" => "Country",
"rock" => "Rock",
"r_n_b" => "R & B",
"hip_hop" => "Hip-Hop",
"heavy_metal" => "Heavy-Metal",
"alternative_rock" => "Alternative Rock",
"christian" => "Christian",
"jazz" => "Jazz",
"pop" => "Pop"
));
$genreElement = new Zend_Form_Element_Select('genre',
$genres);
$genreElement->setLabel("Genre:");
$genreElement->setRequired(true);
//Create favorite radio buttons.
$favoriteOptions = array("multiOptions" => array

```

```

(
"1" => "yes",
"0" => "no"
));

$isFavoriteListElement = new
Zend_Form_Element_Radio('isFavorite',
$favoriteOptions);

$isFavoriteListElement->setLabel("Add to Favorite List:");
$isFavoriteListElement->setRequired(true);

//Create Rating raio button

$ratingOptions = array("multiOptions" => array
(
"1" => "1",
"2" => "2",
"3" => "3",
"4" => "4",
"5" => "5"
));

$ratingElement = new Zend_Form_Element_Radio('rating',
$ratingOptions);

$ratingElement->setLabel("Rating:");

$ratingElement->setRequired(true)->addValidator(new
Zend_Validate_Alnum(false));

//Create submit button

$submitButton = new Zend_Form_Element_Submit("submit");

$submitButton->setLabel("Add Artist");

//Add Elements to form

$form->addElement($artistNameElement);

```

```
$form->addElement($genreElement);
$form->addElement($isFavoriteListElement);
$form->addElement($ratingElement);
$form->addElement($submitButton);
return $form;
}

/**
 * Add an artist to the database.
 *
 */
public function newAction()
{
    //Check if the user is logged in.
    //Set the view variables
    $this->view->form = $this->getAddArtistForm();
}
```



## File Uploading

File Uploading trong form là điều hoàn toàn có thể. Với `Zend_Form_Element_File` class bạn có thể upload mọi file mp3 hoặc ảnh lên trên server.

**Table 4-10.** *Zend\_Form\_Element\_File Methods*

Function	Description
<code>setDestination()</code>	Sets the path to the destination where the files will be saved.
<code>setMultiFile()</code>	Sets the total number of files to upload.
<code>setMaxFileSize()</code>	Sets the maximum file size.
<code>isUploaded()</code>	Determines whether the file has been uploaded.
<code>isReceived()</code>	Determines whether the file has been received by the server.
<code>isFiltered()</code>	Determines whether any filtering has been done to the file.
<code>addValidator()</code>	Adds constraints to the file upload. Acceptable values: count, size, extension.

Để minh họa cách sử dụng cách phương thức trên, chúng ta sẽ thêm chức năng cho phép thêm ảnh avatar cho trang thông cá nhân. Đầu tiên ta phải tạo thư mục đựng ảnh trong thư mục public. Thư mục chỉ chứa một bức ảnh cho 1 user, mỗi ảnh được đổi tên theo tên của user sau khi upload.

Listing 4-39. Updated getUpdateForm()

```
/**
 * Update Form
 */
private function getUpdateForm()
{
    //Create Form
    $form = new Zend_Form();
    $form->setAction('update');
    $form->setMethod('post');
    $form->setAttrib('sitename', 'loubbite');
    $form->setAttrib('enctype', 'multipart/form-data');
    //Load Elements class
    require "Form/Elements.php";
    $LoubbiteElements = new Elements();
    //Create Username Field.
    $form->addElement($LoubbiteElements->getUsernameTextField());
    //Create Email Field.
    $form->addElement($LoubbiteElements->getEmailTextField());
    //Create Password Field.
    $form->addElement($LoubbiteElements->getPasswordTextField());
    //Create Text Area for About me.
    $textAreaElement = new Zend_Form_Element_TextArea('aboutme');
    $textAreaElement->setLabel('About Me:');
```

```

$textAreaElement->setAttribs(array('cols' => 15,
    'rows' => 5));
$form->addElement($textAreaElement);
//Add File Upload
$fileUploadElement = new Zend_Form_Element_File('avatar');
$fileUploadElement->setLabel('Your Avatar:');
$fileUploadElement->setDestination('../public/users');
$fileUploadElement->addValidator('Count', false, 1);
$fileUploadElement->addValidator('Extension', false,
    'jpg,gif');
$form->addElement($fileUploadElement);
//Create a submit button.
$form->addElement('submit', 'submit');
$submitElement = $form->getElement('submit');
$submitElement->setLabel('Update My Account');
return $form;
}

```

#### **Listing 4-40. Updating Form with File Upload**

```

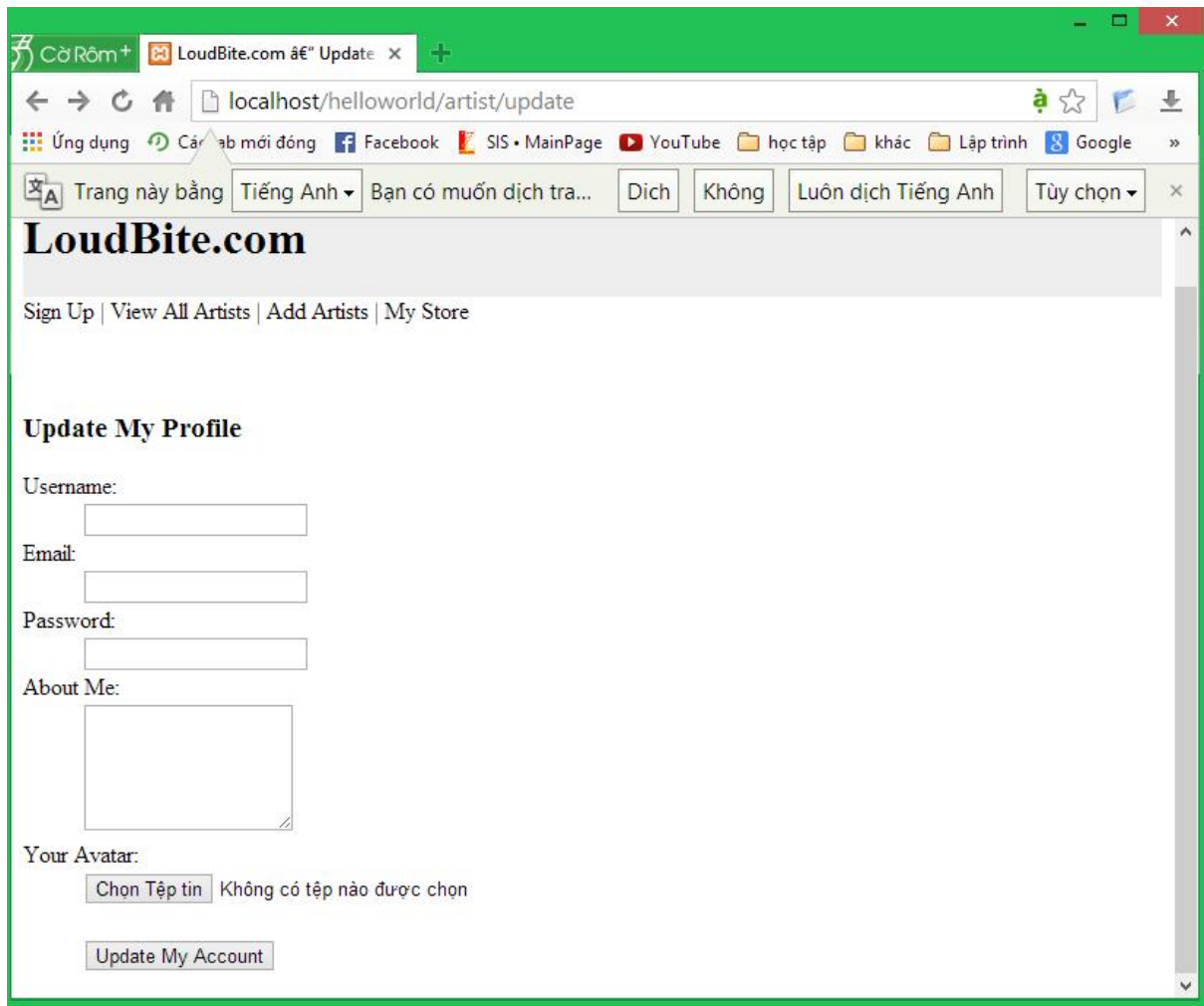
/**
 * Update the User's data.
 *
 */
public function updateAction()
{
    //Check if the user is logged in
    //Fetch the user's id
    //Fetch the users information
    //Create the form.

```

```
$form = $this->getUpdateForm();

//Check if the form has been submitted.
//If so validate and process.
if($_POST){
    //Check if the form is valid.
    if($form->isValid($_POST)){
        //Get the values
        $username = $form->getValue('username');
        $password = $form->getValue('password');
        $email = $form->getValue('email');
        $aboutMe = $form->getValue('aboutme');
        //Save the file
        $form->avatar->receive();
        //Save.
    }
    //Otherwise redisplay the form.
    else{
        $this->view->form = $form;
    }
}

//Otherwise display the form.
else{
    $this->view->form = $form;
}
}
```



Để upload nhiều file chúng ta có thể sử dụng `Zend_Form_Element_File::setMultiFile()`, với tham số là số lượng file cần upload.

### Tạo CAPTCHA

CAPTCHA là phương thức căn bản nhất để kiểm tra xem dữ liệu đầu vào là do máy tính nhập hay con người nhập.

CAPTCHAS có rất nhiều kiểu khác nhau. Ví dụ như sử dụng ảnh và một câu hỏi đơn giản để người dùng trả lời. Zend\_Form cung cấp một cách thức đơn giản để tạo ra captcha sử dụng `Zend_Form_Element_Captcha`. Class này cung cấp 4 kiểu captchas:

- Images-based
- Dumb (user types word displayed backward)
- Figlet
- ReCaptcha

Dưới đây là một số phương thức:

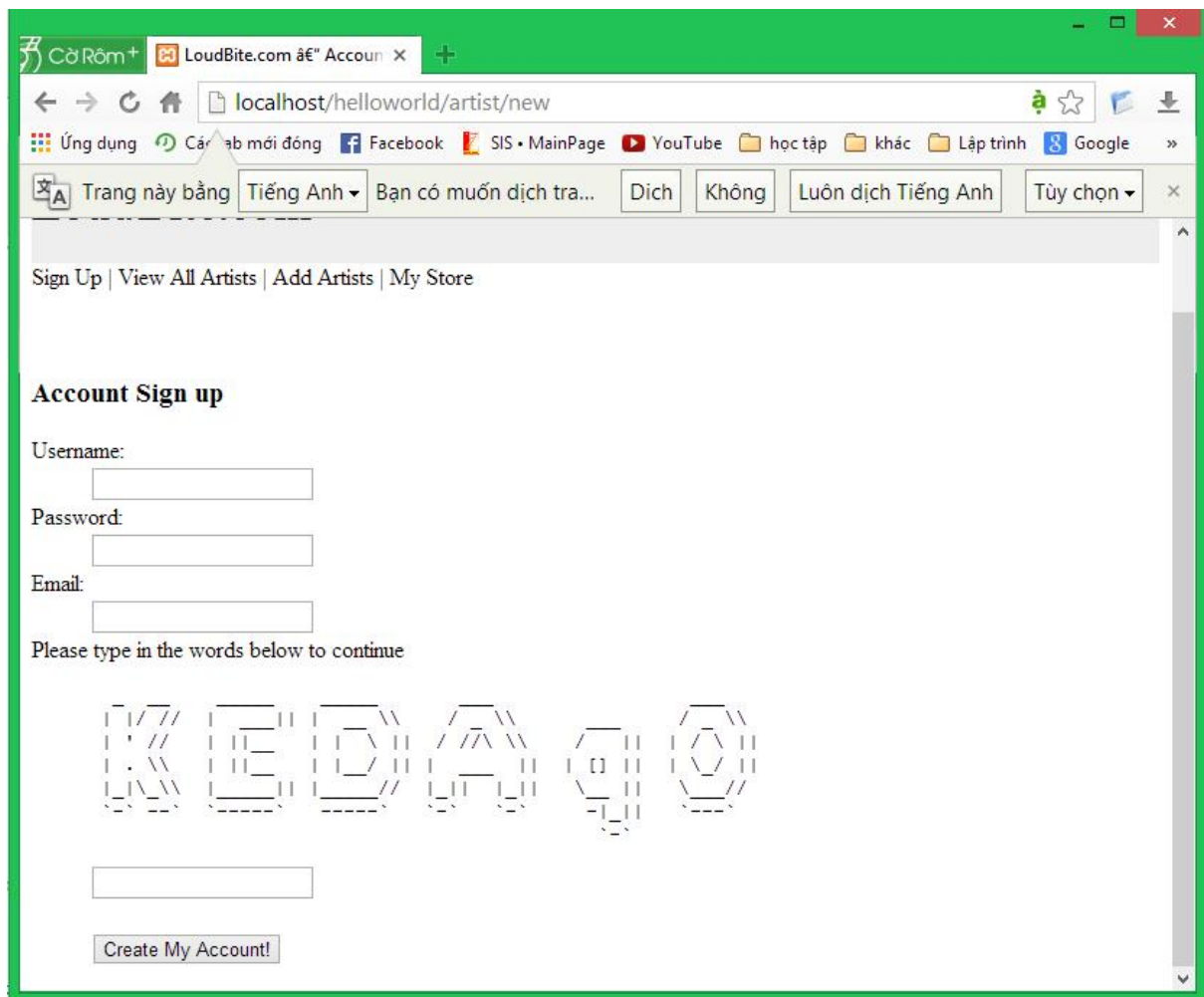
**Table 4-11.** *Zend\_Form\_Element\_Captcha* Setters

Function	Description
<code>setExpiration()</code>	Determines how long a CAPTCHA image should reside in the server (accepts time in seconds).
<code>setGcFreq()</code>	Determines how often garbage collection is run (the default is <code>1/&lt;value you set&gt;</code> ).
<code>setFont()</code>	Sets the font to use.
<code>setFontSize()</code>	Sets the font size to use.
<code>setHeight()</code>	Sets the image height used for CAPTCHA.
<code>setWidth()</code>	Sets the width of the image used for CAPTCHA.
<code>setImgDir()</code>	Sets the image directory that holds the images to use for CAPTCHA.
<code>setImgUrl()</code>	Sets the image path to use for the CAPTCHA.
<code>setSuffix()</code>	Sets the file name suffix for the images (the default is <code>.png</code> ).

Để hiểu hơn cách tạo captcha các bạn tham khảo đoạn code sau:

```
/**
 * Create the sign up form.
 */
private function getSignupForm()
{
    //Create Form
    $form = new Zend_Form();
    $form->setAction('success');
    $form->setMethod('post');
    $form->setAttrib('sitename', 'loudbite');
    //Add Elements
    require "Form/Elements.php";
    $LoudbiteElements = new Elements();
    //Create Username Field.
    $form->addElement($LoudbiteElements->getUsernameTextField());
    //Create Email Field.
```

```
$form->addElement($LoudbiteElements->getEmailTextField());  
  
//Create Password Field.  
  
$form->addElement($LoudbiteElements->getPasswordTextField());  
  
//Add Captcha  
  
$captchaElement = new Zend_Form_Element_Captcha  
(  
    'signup',  
    array('captcha' => array(  
        'captcha' => 'Figlet',  
        'wordLen' => 6,  
        'timeout' => 600))  
);  
  
$captchaElement->setLabel('Please type in the  
words below to continue');  
  
$form->addElement($captchaElement);  
  
$form->addElement('submit', 'submit');  
  
$submitButton = $form->getElement('submit');  
$submitButton->setLabel('Create My Account!');  
  
return $form;  
}
```



### 3. Database Communication, Manipulation, and Display

#### Kết nối với Database

Để thử nghiệm việc kết nối với cơ sở dữ liệu, ta tạo một action `testConnAction` trong `AccountController`. Trong action này ta có sử dụng phương thức `setNoRender()` để không sử dụng view.

Listing 5-1. `AccountController.php:testConnAction`

```
/**  
 * Test our connection  
 */  
public function testConnAction()  
{
```



```

try{
$connParams = array("host" => "localhost",
"port" => "<Your Port Number>",
"username" => "<Your username>",
"password" => "<Your password>",
"dbname" => "loubd bite");

$db = new Zend_Db_Adapter_Pdo_Mysql($connParams);
}catch(Zend_Db_Exception $e){
echo $e->getMessage();
}

echo "Database object created.";

//Turn off View Rendering.
$this->_helper->viewRenderer->setNoRender();
}

```

Với những hệ quản trị cơ sở dữ liệu khác nhau chúng ta sẽ sử dụng các `Zend_Db_Adapter` khác nhau:

- `Zend_Db_Adapter_Pdo_Mysql`
- `Zend_Db_Adapter_Pdo_Ibm`
- `Zend_Db_Adapter_Pdo_Mssql`
- `Zend_Db_Adapter_Pdo_Oci`
- `Zend_Db_Adapter_Pdo_Pgsql`
- `Zend_Db_Adapter_Pdo_Sqlite`

Ta tạo một file `Db_Db.php` trong thư mục `application/models/Db`.

Listing 5-2. `Db_Db.php`

```

<?php

/**

 * Database handler.

```

```

*
*/

class Db_Db
{
public static function conn(){
$connParams = array("host" => "localhost",
"port" => "<Your Port Number>",
"username" => "<Your username>",
"password" => "<Your password>",
"dbname" => "loubbite");
$db = new Zend_Db_Adapter_Pdo_Mysql($connParams);
return $db;
}
}

```

Ngoài cách trên ta có thể kết nối với cơ sở dữ liệu bằng cách thêm vào file application.ini của chúng ta những đoạn lệnh sau:

```

resources.db.adapter = "Pdo_mysql"
resources.db.params.host = "localhost"
resources.db.params.username = "root"
resources.db.params.password = ""
resources.db.params.dbname = "loubbite"

```

**Và khi gọi các ham của Zend\_DB với phương thức ví dụ \$this->select()**

## Thêm dữ liệu

### Sử dụng SQL

Listing 5-3. Using Full SQL INSERT Statements: testInsertAction()

```

/**
 * Test Insert
 */

public function testInsertAction()

```

```

{
try {
//Create a DB object
require_once "Db/Db_Db.php";
$db = Db_Db::conn();
//DDL for initial 3 users
$stmt = "INSERT INTO accounts(
username, email, password,status, created_date
)
VALUES(
'test_1', 'test@loubdbite.com', 'password',
'active', NOW()
)";
$stmt2 = "INSERT INTO accounts(
username,email,password,status,created_date
)
VALUES(
'test_2', 'test2@loubdbite.com', 'password',
'active', NOW()
)";
$stmt3 = "INSERT INTO accounts(
username,email,password,status,created_date
)
VALUES (
?, ?, ?, ?, NOW()
)";
//Insert the above statements into the accounts.

```

```

$db->query($statement);

$db->query($statement2);

//Insert the statement using ? flags.

$db->query($statement3, array('test_3', 'test3@loubdbite.com',
'password', 'active'));

//Close Connection

$db->closeConnection();

echo "Completed Inserting";

}catch(Zend_Db_Exception $e){

echo $e->getMessage();

}

//Supress the View.

$this->_helper->viewRenderer->setNoRender();

}

```

Với cách này chúng ta sẽ viết các câu lệnh SQL và sử dụng phương thức Query để thực hiện truy vấn cơ sở dữ liệu.

### Không sử dụng lệnh SQL

```

/**

* Test Insert Method

* Insert data into table using insert()

*/

public function testInsertMethodAction()

{

try{

//Create a DB object

require_once "Db/Db_Db.php";

$db = Db_Db::conn();

//Data to save.

```

```
$userData1 = array("username" => 'test_4',
"email" => 'test4@loubdbite.com',
"password" => 'password',
"status" => 'active',
"created_date" => '0000-00-00 00:00:00');
$userData2 = array("username" => 'test_5',
"email" => 'test5@loubdbite.com',
"password" => 'password',
"status" => 'active',
"created_date"=> '0000-00-00 00:00:00');
$userData3 = array("username" => 'test_6',
"email" => 'test6@loubdbite.com',
"password" => 'password',
"status" => 'active',
"created_date"=> '0000-00-00 00:00:00');
//Insert into the Accounts.
$db->insert('accounts', $userData1);
$db->insert('accounts', $userData2);
$db->insert('accounts', $userData3);
//Close Connection
$db->closeConnection();
echo "Completed Inserting";
} catch (Zend_Db_Exception $e) {
echo $e->getMessage();
}
//Supress the View
$this->_helper->viewRenderer->setNoRender();
```

```
}
```

Ở cách thứ hai này chúng ta sử dụng hàm insert với hai tham số đầu vào là tên bảng và mảng dữ liệu.

## Database Expressions

Chúng ta sử dụng Zend\_Db\_Expr clas để lấy thông tin về cơ sở dữ liệu ví dụ như: NOW(), COUNT(), LOWER(),SUB()

Ví dụ để lấy thời gian khởi tạo user dùng NOW();

Listing 5-5. Zend\_Db\_Expr Usage: testExpressionAction

```
/**
 * Test Expression
 * Using Database Expressions.
 */
public function testExpressionAction()
{
    try{
        //Create a DB object
        require_once "Db/Db_Db.php";
        $db = Db_Db::conn();
        //Data to save.
        $userData = array("username" => 'test_7',
            "email" => 'test7@loubdbite.com',
            "password" => 'password',
            "status" => 'active',
            "created_date"=> new Zend_Db_Expr("NOW()"));
        //Insert into the accounts.
        $db->insert('accounts', $userData);
        //Close Connection
        $db->closeConnection();
    }
```

```

echo "Completed Inserting";

}catch(Zend_Db_Exception $e){

echo $e->getMessage();

}

//Supress the View

$this->_helper->viewRenderer->setNoRender();

}

```

## Escaping Values

Khi bạn thêm thông tin vào cơ sở dữ liệu. Bạn có thắc mắc rằng dữ liệu được lưu vào là chính xác và không gây hỏng hóc. Bạn phải cẩn thận rằng người dùng không thử sử dụng tất cả các công nghệ SQL injection khi lấy thông tin từ các bảng cơ sở dữ liệu.

## SQL Injection

Sql injection là những tổn thương gây ra bởi các hệ quản trị cơ sở dữ liệu, cho phép người dùng thêm các câu lệnh SQL và sẽ được truy vấn đến cơ sở dữ liệu. Để hiểu hơn chúng ta xét một ví dụ:

Giả sử người dùng quyết định đăng nhập vào ứng dụng và điền nội dung sau vào trường username mà không được lọc và làm sạch:

```
user'DELETE * FROM Accounts
```

Và khi thực hiện truy vấn câu lệnh INSERT vào cơ sở dữ liệu thì cũng đồng thời thực hiện câu lệnh DELETE được thêm vào code bởi người dùng. Điều này nguy hiểm với bảng account vì có khả năng toàn bộ bản ghi sẽ bị xóa. Điều này là vô cùng nguy hiểm đối với ứng dụng của chúng ta. Vì vậy Zend Framework cung cấp phương thức để chúng ta giải quyết vấn đề này.

## Escaping User Data

Để chống lại những tấn công không mong muốn Zend Framework cũng cấp hai phương thức để escape dấu nháy đơn. Sử dụng quote() và quoteInto()

Phương thức quote() chấp nhận hai tham số. Giá trị khởi tạo là 1 chuỗi, và mảng các giá trị muốn escape, Zend\_Db\_Expr object và Zend\_Db\_Select object.

Theo mặc định phương thức này sẽ trả về một chuỗi với nháy đơn. Ví dụ khi bạn nhập “this is a test” là tham số thì kết quả trả về sẽ là ‘’this is a test’’. Giá trị trả về là tạm chấp nhận được

vì mỗi hệ quản trị cơ sở dữ liệu có nhiều cách khác nhau để thêm hoặc cập nhật dữ liệu. Mỗi một giá trị được truy vấn nằm trong dấu nháy đơn.

Trong một số trường hợp cơ sở dữ liệu không cho phép lưu trữ dưới dạng xâu, ví dụ như khi lưu trữ số kiểu INTERGER. Lúc này bạn phải sử dụng tham số thứ hai . Theo mặc định tham số thứ 2 này là null, nhưng bạn phải ghi đề tham số này khi làm việc với các kiểu dữ liệu INTERGER, Float, bigint.

Xem ví dụ:

#### Listing 5-6. Quoting Strings: testQuoteAction()

```
/**
 * Test Quote
 */
public function testQuoteAction()
{
    try {
        //Create Db object
        require_once "Db/Db_Db.php";
        $db = Db_Db::conn();
        $username = "testing ' user";
        $usernameAfterQuote = $db->quote($username);
        echo "BEFORE QUOTE: $username<br>";
        echo "AFTER QUOTE: $usernameAfterQuote<br>";
        //DDL for initial 3 users
        echo $statement = "INSERT INTO accounts(
        username, email, password, status, created_date
        )
        VALUES (
        $usernameAfterQuote, 'test8@loubdbite.com', 'password',
        'active', NOW()
        ) ";
```



```

//Insert the above statements into the accounts.
$db->query($statement);
//Close Connection
$db->closeConnection();
echo "Successfully inserted.";
} catch (Zend_Db_Exception $e) {
echo $e->getMessage();
}
//Supress the View.
$this->_helper->viewRenderer->setNoRender();
}

```

Khi chạy thử thì chúng ta thấy kết quả trước khi dùng `quote()` là `testing' user` và sau khi dùng `quote()` là `'testing' user`

## Last Inserted ID

Nhiều trường hợp ta cần phải lấy id của account cuối cùng được chèn vào một bảng chẳng hạn, lúc này ta có thể dùng hàm `lastInsertId()` với hai tham số:

- Tên của bảng
- Tên của thuộc tính là khóa chính

### Listing 5-7. Retrieving the ID for the New Record

```

/**
 * Test Last Insert
 */
public function testLastInsertAction()
{
try {
//Create Db object
require_once "Db/Db_Db.php";
$db = Db_Db::conn();

```

```

//Data to save.

$userData = array("username" => 'testinguser9',
"email" => 'test9@loubdbite.com',
"password" => 'password',
"status" => 'active',
"created_date" => new Zend_Db_Expr("NOW()"));
$db->insert('accounts', $userData);

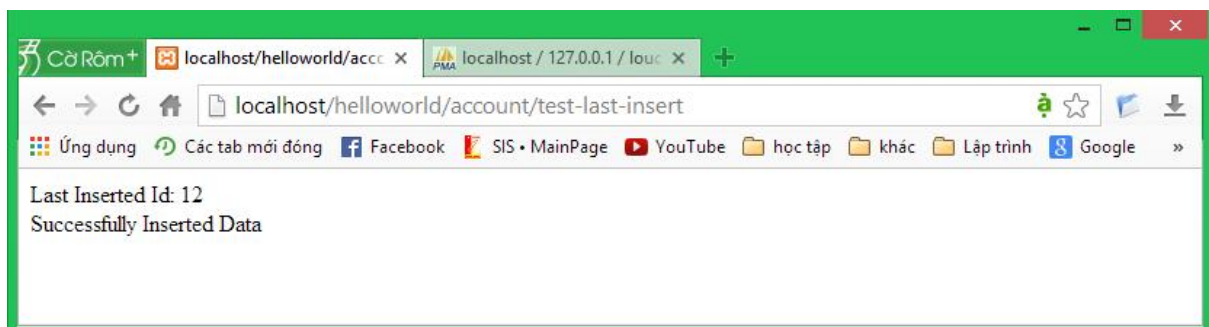
//Retrieve the id for the new record and echo
$id = $db->lastInsertId();
echo "Last Inserted Id: ".$id."<br>";

//Close Connection
$db->closeConnection();

echo "Successfully Inserted Data";
} catch (Zend_Db_Exception $e) {
echo $e->getMessage();
}

//Supress the View.
$this->_helper->viewRenderer->setNoRender();
}

```



## LoundBite Sign-up page

Mở file AccountController và xem successAction(). Action này lấy thông tin từ người dùng, làm sạch chúng và sau đó không làm gì cả. bạn cần phải hoàn thiện nốt phần còn lại bây giờ.

Sau khi dữ liệu người dùng được làm sạch, bạn cần phải tạo một hàm lưu trữ dữ liệu. Sử dụng kiến thức vừa học, chúng ta sẽ tạo hàm chèn dữ liệu vào cơ sở dữ liệu.

Listing 5-8. AccountController.php: SuccessAction

```
/**
 * Process Sign up Form.
 *
 */
public function successAction()
{
    $form = $this->getSignupForm();

    //Check if the submitted data is POST type
    if($form->isValid($_POST)){
        $email = $form->getValue("email");
        $username = $form->getValue("username");
        $password = $form->getValue("password");

        //Create Db object
        require_once "Db/Db_Db.php";
        $db = Db_Db::conn();

        //Create the record to save into the Db.
        $userData = array("username" => $username,
            "email" => $email,
            "password" => $password,
            "status" => 'pending',
            "created_date" => new Zend_Db_Expr("NOW()"));
        try{
            //Insert into the accounts.
            $db->insert('accounts', $userData);

            //Get the Id of the user
```

```

$userId = $db->lastInsertId();

//Send out thank you email. We'll get to this. Chapter 6.
}catch(Zend_Db_Exception $e){

$this->view->form = $form;

}

}else{

$this->view->errors = $form->getMessages();

$this->view->form = $form;

}

}

```

## LoudBite Add Artist

Bây giờ mở file `ArtistController.php`. Bạn sẽ sửa lại file `saveArtistAction` để lưu trữ một user's artist. Do trang login chưa được tạo vì vậy chúng ta phải tạo một account ID tĩnh.

Listing 5-9. `ArtistController.php: saveartistAction()`

```

/**

* Save Artist to Db

*

*/

public function saveArtistAction()

{

//Create instance of artist form.

$form = $this->getAddArtistForm();

//Check if there were no errors

if($form->isValid($_POST)){

//Initialize the variables

$artistName = $form->getValue('artistName');

$genre = $form->getValue('genre');

```

```
$rating = $form->getValue('rating');

$isFav = $form->getValue('isFavorite');

//Set the temporary account id to use.

$userId = 10;

try{

//Create a db object

require_once "Db/Db_Db.php";

$db = Db_Db::conn();

//Initialize data to save into DB

$artistData = array("artist_name" => $artistName,

"genre" => $genre,

"created_date" =>

new Zend_Db_Expr("NOW()"));

//Insert the artist into the Db

$db->insert('artists', $artistData);

//Fetch the artist id

$artistId = $db->lastInsertId();

//Initialize data for the account artists table

$accountArtistData = array("account_id" => $userId,

"artist_id" => $artistId,

"rating" => $rating,

"is_fav" => $isFav,

"created_date" =>

new Zend_Db_Expr("NOW()"));

//Insert the data.

$db->insert('accounts_artists', $accountArtistData);

} catch(Zend_Db_Exception $e){
```

```

echo $e->getMessage();

}

}else{

$this->view->errors = $form->getMessages();

$this->view->form = $form;

}

}

```

### Hướng đối tượng với câu lệnh SELECT

Zend\_Db\_Select là class cho phép tạo ra câu lệnh SELECT . Sử dụng lớp này dữ liệu đã được escapes và qua bộ lọc để tránh các tấn công SQL injection.

Listing 5-24. Simple Object-Oriented Statement: teststatementAction

```

/**
 * Test - Object Oriented Select Statement
 *
 */

public function teststatementAction() {

//Create DB object

require_once "Db/Db_Db.php";

$db = Db_Db::conn();

$select = new Zend_Db_Select($db);

$statement = $select->from('artists');

//Compare Statement

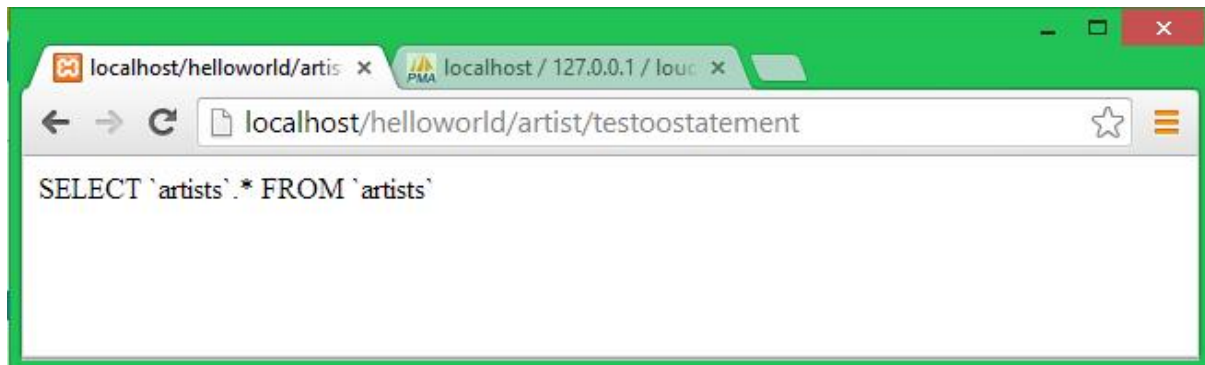
echo $statement->__toString();

//Supress the View

$this->_helper->viewRenderer->setNoRender();

}

```



### Querying Specific Columns

Nhiều khi bạn không cần chọn tất cả các cột của một bảng. Ví dụ câu lệnh truy vấn sau:

```
SELECT `artists`.`id`, `artists`.`artist_name`, `artists`.`genre` FROM `artists`
```

Sử dụng phương thức form() với tham số thứ hai là mảng các cột cần truy vấn:

Listing 5-26. Identifying Specific Columns to Fetch

```
/**
 * Test - Object Oriented Select Statement
 *
 */
public function testoostatementAction() {
    //Create DB object
    require_once "Db/Db_Db.php";
    $db = Db_Db::conn();

    //Create the statement
    //SELECT `artists`.`id`, `artists`.`artist_name`,
    `artists`.`genre`
    //FROM `artists`

    $select = new Zend_Db_Select($db);

    //Determine which columns to retrieve.
    $columns = array('id', 'artist_name', 'genre');
    $statement = $select->from('artists', $columns);
```

```
//Compare Statement
echo $statement->__toString();

//Supress the View
$this->_helper->viewRenderer->setNoRender();
}
```



## Thực thi truy vấn

Để thực thi truy vấn thì ta thêm đoạn code sau:

```
//Query the Database
$results = $db->query($statement);
$rows = $results->fetchAll();
```

## Creating Column and Table Aliases

Nếu bạn muốn hiện thị một cột với tên khác ví dụ:

```
SELECT `a`.`id` AS `artist id`, `a`.`artist_name` AS `name`, `a`.`genre`
FROM `artists` AS `a`
```

Xét ví dụ sau:

Listing 5-28. Implementing Table and Column Aliases

```
/**
 * Test - Object Oriented Select Statement
 *
 */
public function testoostatementAction() {
```



```

//Create DB object
require_once "Db/Db_Db.php";
$db = Db_Db::conn();

//Create the statement
//SELECT `a`.`id` AS `artist id`, `a`.`artist_name` AS `name`,
//`a`.`genre` FROM `artists` AS `a`
$select = new Zend_Db_Select($db);

//Determine which columns to retrieve.
//Determine which table to retrieve data from.
$columns = array("artist id" => 'id',
"name" => 'artist_name',
"genre" => 'genre');
$tableInfo = array("a" => "artists");
$statement = $select->from($tableInfo, $columns);

//Query the Database
$results = $db->query($statement);
$rows = $results->fetchAll();

//Compare Statement
echo $statement->__toString();

//Supress the View
$this->_helper->viewRenderer->setNoRender();
}

```

## Điều kiện với WHERE

Xét các ví dụ sau và câu lệnh tương ứng

```

SELECT `a`.`id`, `a`.`artist_name` AS `name`, `a`.`genre` FROM `artists`
AS `a` WHERE (artist_name='Groove Armada')

```

```

$columns = array("id" => 'id',
"name" => 'artist_name',
"genre" => 'genre');
$tableInfo = array("a" => 'artists');
$statement = $select->from($tableInfo, $columns)
->where("artist_name=?", 'Groove Armada');

```

```

SELECT `a`.`id`, `a`.`artist_name` AS `name`, `a`.`genre` FROM `artists`
AS `a` WHERE (artist_name='Groove Armada') AND (genre='electronic')

```

//Determine which columns to retrieve.

//Determine which table to retrieve data from.

```

$column = array("id" => 'id',
"name" => 'artist_name',
"genre" => 'genre');
$tableInfo = array("a" => 'artists');
$statement = $select->from($tableInfo, $column)
->where("artist_name=?", 'Groove Armada')
->where('genre=?', 'electronic');

```

```

SELECT `a`.`id`, `a`.`artist_name` AS `name`, `a`.`genre` FROM `artists` AS `a`
WHERE (artist_name='Groove Armada') AND (genre='electronic')
OR (genre='house')

```

//Determine which columns to retrieve.

//Determine which table to retrieve data from.

```

$columns = array("id" => 'id',
"name" => 'artist_name',
"genre" => 'genre');
$tableInfo = array("a" => 'artists');
$statement = $select->from($tableInfo, $columns)

```

->where("artist\_name=?", 'Groove Armada')

->where('genre=?', 'electronic')

->orWhere('genre=?', 'house');

## Truy vấn 2 hay nhiều bảng với câu lệnh JOIN

```
SELECT `a`.`id` AS `artist id`, `a`.`artist_name` AS `name`, `a`.`genre`,  
`aa`.`account_id` AS `user_id`, `aa`.`created_date` AS `date_became_fan` FROM  
`artists` AS `a` INNER JOIN `accounts_artists` AS `aa` ON aa.artist_id = a.id
```

**Table 5-5. Join Methods**

Method	Description	Parameters
join() OR joinInner()	INNER JOIN SQL call	join(table name, join condition, columns to retrieve) Example: join(array("tableAlias" => "tableName"), "table1.id = table2.id", array("column_alias" => "column"));
joinLeft()	LEFT JOIN SQL call	joinLeft(table name, join condition, columns to retrieve) Example: join(array("tableAlias" => "tableName"), "table1.id = table2.id", array("column_alias" => "column"));
joinRight()	RIGHT JOIN SQL call	joinRight(table name, join condition, columns to retrieve) Example: joinRight(array("tableAlias" => "tableName"), "table1.id = table2.id", array("column_alias" => "column"));
joinFull()	FULL JOIN SQL call	joinFull(table name, join condition, columns to retrieve) Example: joinFull(array("tableAlias" => "tableName"), "table1.id = table2.id", array("column_alias" => "column"));
joinCross()	CROSS JOIN SQL call	joinCross(table name, columns to retrieve) Example: joinCross(array("tableAlias" => "tableName"), array("column_alias" => "column"));
joinNatural()	NATURAL JOIN SQL call	joinNatural(table name, columns to retrieve) Example: joinNatural(array("tableAlias" => "tableName"), array("column_alias" => "column"));

Xét ví dụ:

```
/**
```

```
* Test - Get All Fans
```

```
*
```

```

*/

public function testtoofansAction(){

//Create DB object

require_once "Db/Db_Db.php";

$db = Db_Db::conn();

//Create the statement

//SELECT `a`.`id` AS `artist id`, `a`.`artist_name` AS `name`,
//`a`.`genre`,aa`.`account_id` AS `user_id`,
//`aa`.`created_date` AS `date_became_fan`
//FROM `artists` AS `a`
//INNER JOIN `accounts_artists` AS `aa` ON aa.artist_id = a.id

$select = new Zend_Db_Select($db);

//Determine which columns to retrieve.

//Determine which table to retrieve data from.

$columns = array("artist id" => 'a.id',
"name" => 'a.artist_name',
"genre" => 'a.genre');

$tableInfo = array("a" => 'artists');

$statement = $select->from($tableInfo, $columns)
->join(array("aa" => 'accounts_artists'),
'aa.artist_id = a.id',
array("user_id" => 'aa.account_id',
"date_became_fan" =>
'aa.created_date'));

$results = $db->query($statement);

$rows = $results->fetchAll();

//Compare Statement

```

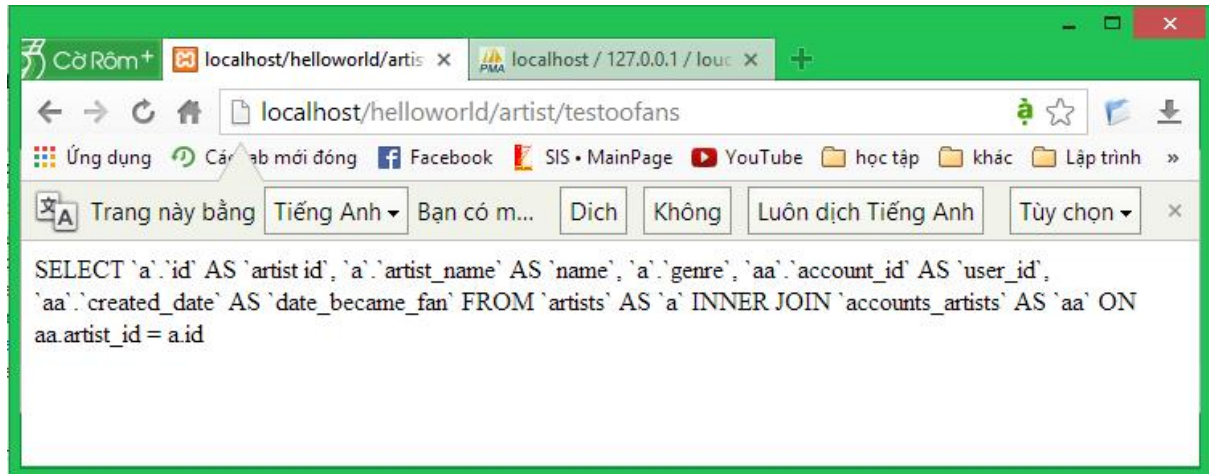
```

echo $statement->__toString();

//Supress the View

$this->_helper->viewRenderer->setNoRender();
}

```



## Limiting and Order the Result Set

Với câu lệnh limit ta dùng phương thức limit() còn với sắp xếp kết quả thì dùng phương thức order():

Listing 5-33. testoofansAction Using order()

```

/**
 * Test - Get All Fans
 *
 */

public function testoofansAction() {

    //Create DB object
    require_once "Db/Db_Db.php";
    $db = Db_Db::conn();

    //Create the statement
    //SELECT `a`.`id` AS `artist id`, `a`.`artist_name` AS `name`,
    //`a`.`genre`, `aa`.`account_id` AS `user_id`,
    //`aa`.`created_date` AS `date_became_fan`

```

```

//FROM `artists` AS `a`

//INNER JOIN `accounts_artists` AS `aa` ON aa.artist_id = a.id

//ORDER BY `date_became_fan` DESC LIMIT 10

$select = new Zend_Db_Select($db);

//Determine which columns to retrieve.

//Determine which table to retrieve data from.

$columns = array("artist id" => 'a.id',

"name" => 'a.artist_name',

"genre" => 'a.genre');

$tableInfo = array("a" => 'artists');

$statement = $select->from($tableInfo, $columns)

->join(array("aa" => 'accounts_artists'),

'aa.artist_id = a.id',

array("user_id" => 'aa.account_id',

"date_became_fan" =>

'aa.created_date'))

->order("date_became_fan DESC")

->limit(10);

$results = $db->query($statement);

$rows = $results->fetchAll();

//Compare Statement

echo $statement->__toString();

//Supress the View

$this->_helper->viewRenderer->setNoRender();

}

```



## Database Expressions

```
SELECT COUNT(id) AS `total_fans` FROM `accounts_artists` AS
`aa`
```

Listing 5-34. Implementing the count() Method

```
/**
 * Test - Database expression.
 *
 */

public function testoocountAction(){
    //Create Db object
    require_once "Db/Db_Db.php";
    $db = Db_Db::conn();

    //Create the statement

    // SELECT COUNT(id) AS `total_fans` FROM `accounts_artists` AS
    `aa`

    $select = new Zend_Db_Select($db);

    //Determine which columns to retrieve.

    //Determine which table to retrieve data from.

    $columns = array("total_fans" => 'COUNT(id)');
    $tableInfo = array("aa" => 'accounts_artists');
    $statement = $select->from($tableInfo, $columns);
```

```

$results = $db->query($statement);
$rows = $results->fetchAll();

//Compare Statement
echo $statement->__toString();

//Supress the View
$this->_helper->viewRenderer->setNoRender();
}

```

```

SELECT DISTINCT `a`.`genre` FROM `artists` AS `a`

```

Let's go ahead and create the object-oriented equivalent of this statement (see Listing 5-35).

Listing 5-35. Implementing the distinct() Method

```

/**
 * Test - Return distinct genres
 *
 */

public function testoogenrelistAction(){
    //Create Db object
    require_once "Db/Db_Db.php";
    $db = Db_Db::conn();

    //Create the statement
    //SELECT DISTINCT `a`.`genre` FROM `artists` AS `a`
    $select = new Zend_Db_Select($db);

    //Determine which columns to retrieve.
    //Determine which table to retrieve data from.
    $columns = array("genre" => 'a.genre');
    $tableInfo = array("a" => 'artists');
    $statement = $select->from($tableInfo, $columns)
    ->distinct();
}

```



```

$results = $db->query($statement);

$rows = $results->fetchAll();

//Compare Statement

echo $statement->__toString();

//Supress the View

$this->_helper->viewRenderer->setNoRender();

}

```

## Phân trang

Ứng dụng của chúng ta có thể bao gồm hàng nghìn hoặc hàng triệu bản ghi. Nhưng ở mỗi trang chúng ta chỉ có thể hiển thị một số lượng bản ghi nhất định. Đó là lý do mà chúng ta phải phân trang.

### Sử dụng Zend\_Paginator

Chúng ta có thể dùng Zend\_Paginator để phân trang cho các đối tượng sau

- Zend\_Paginator\_Adaptor\_Array
- Zend\_Db\_Select
- Zend\_Db\_Table\_Select
- Iterator

Để có thể phân trang cho các đối tượng trên các bạn dùng hàm `Zend_Paginator::factory()`

Zend\_Paginator cung cấp một số phương thức sau:

**Table 5-6. Zend\_Paginator Setters**

Method	Description
<code>setCurrentPageNumber()</code>	Sets the current page number; the default is 1.
<code>setItemCountPerPage()</code>	Sets the number of records to display per page; the default is 10.
<code>setPageRange()</code>	Sets the total number of pages to display in the pagination control; the default is 10.
<code>setView()</code>	Sets the view that you want to associate with this pagination.

Để hiểu hơn chúng ta xét ví dụ phân trang cho một mảng user:

Listing 5-36. ArtistController.php: listAction()

```

/ **

```

```

* Display all the Artists in the system.
*/

public function listAction(){
    //Create a sample array of artist
    $artist = array("Underworld", "Groove Armada", "Daft Punk",
    "Paul Oakenfold", "MC Chris", "Ramones",
    "The Beatles", "The Mamas and the Papas",
    "Jimi Hendrix");

    //Initialize the Zend_Paginator
    $paginator = Zend_Paginator::factory($artist);
    $currentPage = 1;

    //Check if the user is not on page 1
    $i = $this->_request->getQuery('i');
    if(!empty($i)){ //Where i is the current page
        $currentPage = $this->_request->getQuery('i');
    }

    //Set the properties for the pagination
    $paginator->setItemCountPerPage(2);
    $paginator->setPageRange(3);
    $paginator->setCurrentPageNumber($currentPage);
    $this->view->paginator = $paginator;
}

```

Listing 5-37. list.phtml

```

<?php echo $this->doctype('XHTML1_STRICT'); ?>

<html
xmlns="http://www.w3.org/1999/xhtml"xml:lang="en"lang="en">

<head>

```

```
<?php echo $this->headTitle('LoudBite.com - Artist Listing');
?>

</head>

<body>

<?php echo $this->render("includes/header.phtml"?>

<table border='0' width='600'>

<?php

if ($this->paginator){

foreach($this->paginator as $item){

echo "<tr><td>".$item."</td></tr>";

}

}else{?>

<tr><td>There were no artists present in the system.
Add one now!</td></tr>

<?php

}

?>

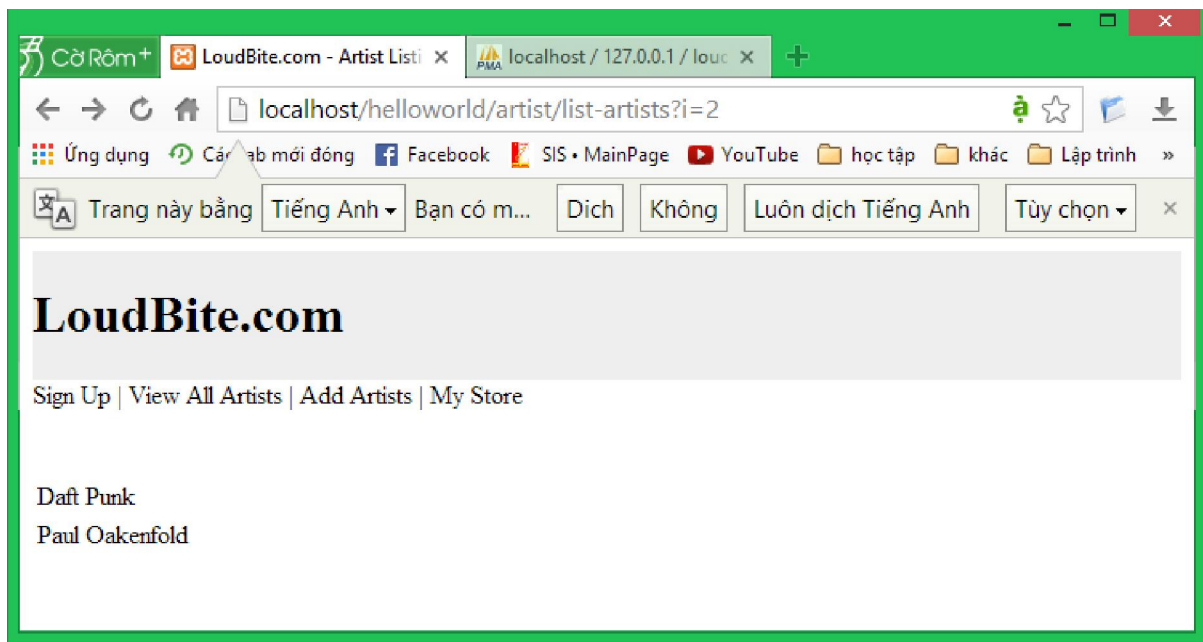
</table>

</body>

</html>
```

Chúng ta vào trình duyệt gõ đường link sau:

<http://localhost/helloworld/artist/list?i=<pagenumber>>



## Adding Control to the Paginator

Ở trên đã trình bày cho chúng ta cách phân trang nhưng mà không có hiện số trang và nút next hoặc prev. Để thêm thanh này chúng ta có thể sử dụng `paginationControl()`.

**Table 5-7.** *Pagination Controller Styles*

Scrolling Style	Description
All	Displays all the pages available to the user regardless of whether you use <code>setPageRange()</code> .
Elastic	Displays an initial set of pages. As the user moves from page to page, the number of pages displayed on the paginator control will expand.
Jumping	Displays a given range; moves on to the next range when the user gets to the end of the first range.
Sliding	Displays the page range set using <code>setPageRange()</code> . Places the current page number as close to the center as possible.

**Table 5-8.** *Zend\_Paginator Properties*

Property	Description	Return Type
first	Returns the initial page number in the set of pages.	String
firstItemNumber	Returns the index of the first record in the result set on the current page.	String
firstPageInRange	Returns the first page in the scrolling range for the current page.	String
current	Returns the current page number in the scrolling page range.	String
currentItemCount	Returns the current number of records displayed in the current page.	String
itemCountPerPage	Returns the total number of records that can be displayed per page.	String
last	Returns the last page number in the total set of pages.	String
lastItemNumber	Returns the index of the last record on the current page.	String
lastPageInRange	Returns the last page number in the current scrolling page range.	String
next	Returns the next page number to be fetched.	String
pageCount	Returns the total number of pages.	String
pagesInRange	Returns all the page numbers to display in the current scrolling page range. Key-value array: Value represents the page number.	Array
previous	Returns the previous page number.	String
totalItemCount	Returns the total number of records to paginate.	String

Chúng ta tạo file 'includes/paginationcontrol.phtml'

```
<?php
if ($this->pageCount) {
?>

<table>

<tr>

<!-- ADD PREVIOUS BUTTON -->

<?php
if (isset ($this->previous)) {
?>
```

```

<td><a href="list?i=<?php echo $this->previous?>"> &lt;-Previous</a>
|
</td>

<?php } ?>

<!-- SHOW THE AVAILABLE PAGES -->

<?php
foreach($this->pagesInRange as $page){
if($page != $this->current){
?>
<td>
<a href="list?i=<?php echo $page?>"><?php echo $page; ?></a> |
</td>
<?
}else{
?>
<td><?php echo $this->current; ?> |</td>
<?
}
}
?>

<!-- ADD NEXT BUTTON -->

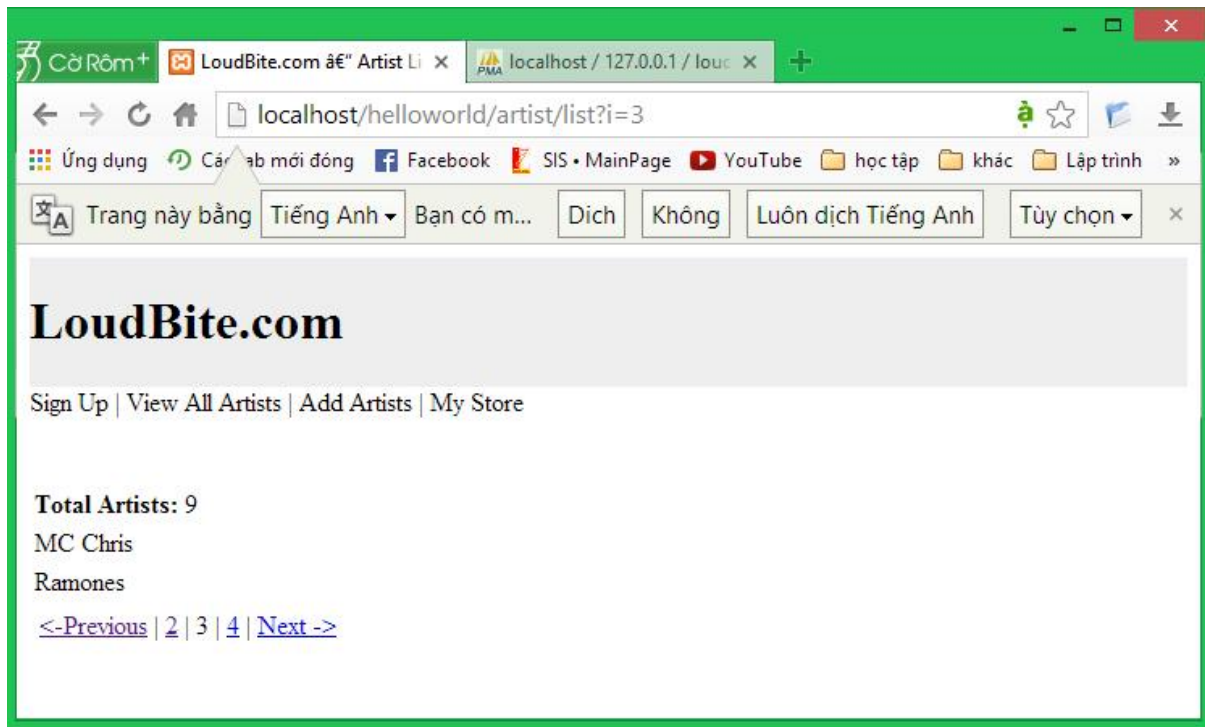
<?php
if(isset($this->next)){
?>
<td><a href="list?i=<?php echo $this->next?>">Next -&gt;</a></td>
<?}?>
</tr>
</table>

<?php

```

}

?>



### Phân trang bản ghi cơ sở dữ liệu

Listing 5-40. Using Zend\_Db\_Select with Zend\_Paginator

```
/**
```

```
* Display all the Artists in the system.
```

```
*/
```

```
public function listAction(){
```

```
    $currentPage = 1;
```

```
    //Check if the user is not on page 1
```

```
    $i = $this->_request->getQuery('i');
```

```
    if(!empty($i)){ //Where i is the current page
```

```
        $currentPage = $this->_request->getQuery('i');
```

```
    }
```

```
    //Create Db object
```

```
    require_once "Db/Db_Db.php";
```

```

$db = Db_Db::conn();

//Create a Zend_Db_Select object

$sql = new Zend_Db_Select($db);

//Define columns to retrieve as well as the table.

$columns = array("id", "artist_name");

$table = array("artists");

//SELECT `artists`.`id`, `artists`.`artist_name` FROM `artists`

$stmt = $sql->from($table, $columns);

//Initialize the Zend_Paginator

$paginator = Zend_Paginator::factory($stmt);

//Set the properties for the pagination

$paginator->setItemCountPerPage(10);

$paginator->setPageRange(3);

$paginator->setCurrentPageNumber($currentPage);

$this->view->paginator = $paginator;

}

```

Sử phần view

```

foreach($this->paginator as $item){

    echo "<tr><td>".$item["artist_name"]."</td></tr>"; }

```

Bổ sung

### Zend Framework: Tương tác cơ sở dữ liệu với Zend\_Db\_Table

Kết thúc bài vừa rồi, chúng ta đã **tim hiểu về quy trình làm việc trên view và cơ bản về layout trong zend framework**. Tiếp tục ở bài này, tôi sẽ hướng dẫn các bạn tìm hiểu về quy trình tương tác cơ sở dữ liệu trên model như thế nào. **Zend Framework** cho ta nhiều phương pháp để tương tác với cơ sở dữ liệu. Và một trong những lớp tương tác với model mà tôi lựa chọn ở đây chính là lớp **Zend\_Db\_Table**.

Vậy trước hết, ta tìm hiểu xem **model** là gì ?. Và tại sao phải sử dụng **model** ?. Model là tầng xử lý những tác vụ liên quan đến tương tác cơ sở dữ liệu từ những yêu cầu của **controller**. **Model** xử lý và trả về kết quả dưới dạng một mảng dữ liệu, khi đó thông qua **view** ta sẽ đẩy nội dung của mảng dữ liệu ấy ra bên ngoài. Việc tách biệt tầng **model** có rất nhiều thuận lợi, trước là dễ quản lý sau là dễ nâng cấp và phát triển trong tương lai của mã nguồn.

Để tương tác được với **Model** thì trước tiên ta phải kết nối được với cơ sở dữ liệu. Vậy ta tạo 1 bảng user với các cú pháp như sau:

```

01 CREATE TABLE user (

```



```

02 id int(10) unsigned NOT NULL AUTO_INCREMENT,
03 username varchar(50) NOT NULL,
04 password char(32) NOT NULL,
05 level int(1) NOT NULL DEFAULT '1',
06 PRIMARY KEY (id)
07 );
08 INSERT INTO 'user' (username,password,level) VALUES('admin', '12345', 2);
09 INSERT INTO 'user' (username,password,level) VALUES('kenny', '12345', 2);
10 INSERT INTO 'user' (username,password,level) VALUES('jacky', '12345', 1);
11 INSERT INTO 'user' (username,password,level) VALUES('Lena', '12345', 1);

```

Tiếp tục ta kết nối với cơ sở dữ liệu bằng cách thêm vào file application.ini của chúng ta những đoạn lệnh sau:

```

1 resources.db.adapter = "Pdo_mysql"
2 resources.db.params.host = "localhost"
3 resources.db.params.username = "root"
4 resources.db.params.password = ""
5 resources.db.params.dbname = "qhonline"

```

Đoạn thông tin này khai báo cho hệ thống biết các tham số như host, user, pass, dbname.

Sau khi đã kết nối được với cơ sở dữ liệu, tiếp theo. Chúng ta sẽ tạo file User.php trong thư mục Model với nội dung như sau:

```

1 <?php
2 class Model_User extends Zend_Db_Table_Abstract{
3     protected $_name="user";
4     protected $_primary="id";
5     public function listall(){
6         return $this->fetchall()->toArray();
7     }
8 }

```

Qua đoạn lệnh trên ta hiểu phần nào về quy tắc định nghĩa một lớp model trong **zend framework**. Vì tất cả các file nằm trong thư mục Model nên áp dụng theo cơ chế **lazy loading** ta có quy tắc định nghĩa: Model\_Tênfile. Cụ thể ở đây tôi tạo ra file User.php, nên lớp của tôi định nghĩa sẽ là Model\_User.

Lưu ý là thư mục Models của chúng ta có s, nhưng khi định nghĩa thì chúng ta bỏ qua s và viết bình thường là Model.

Trong lớp Model\_User ở trên, ta kế thừa lớp **Zend\_Db\_Table\_Abstract**. Và khai báo tên bảng, tên khóa chính thông qua hai thuộc tính \$\_name và \$\_primary. Cuối cùng, ta định nghĩa phương thức listall() và dùng phương thức fetchall() để lấy toàn bộ dữ liệu vốn có trong bảng user.

Ở đây **zend framework** sử dụng cơ chế **Active Record**. Nên phương thức \$this->fetchall() nó tương đương với cú pháp lập toàn bộ dữ liệu từ câu truy vấn select \* from user vậy.

Sau cùng, ta tạo lớp UserController trong file controllers/UserController.php với nội dung sau:

```

1 <?php
2 class UserController extends Zend_Controller_Action{
3     public function indexAction(){
4         $muser=new Model_User;
5         echo "<pre>";
6         print_r($muser->listall());

```

```

7         echo "</pre>";
8     }
9 }

```

Lưu ý là ta phải tạo trong thư mục views/scripts/user/index.phtml để controller tìm kiếm thấy view khi được gọi. (xem bài tìm hiểu về view trong zend framework).

Để sử dụng được Model trong controller ta phải khởi tạo đối tượng từ lớp Model mà ta đã định nghĩa ở trên. Sau đó từ đối tượng ta lại gọi các phương thức muốn thực thi. Lệnh <pre> ở trên được dùng để trình bày dữ liệu mảng.

Chạy thử nghiệm với đường dẫn: <http://localhost/zfexam/user/>

**Kết quả sẽ thông báo lỗi:**

Fatal error: Class 'Model\_User' not found in C:\xampp\htdocs\zfexam\application\controllers\UserController.php on line 4

Hệ thống của chúng ta chưa hiểu được lớp Model\_User. Vì thế ta cần phải định nghĩa như sau trong file bootstrap.php:

```

01 <?php
02 class Bootstrap extends Zend_Application_Bootstrap_Bootstrap{
03     protected function _initAutoload(){
04         $autoloader = new Zend_Application_Module_Autoloader(array(
05             'namespace' => '',
06             'basePath' => dirname(__FILE__),
07         ));
08         return $autoloader;
09     }
10 }

```

Như các bạn thấy đấy, ta đã thêm vào phương thức autoload để định nghĩa cho hệ thống biết được đường dẫn tới các lớp mà ta tạo trong Model và form sau này.

Chạy thử nghiệm lại với đường dẫn: <http://localhost/zfexam/user/>

Kết quả sẽ hiển thị danh sách user dưới dạng mảng như ta mong đợi.

## Một số phương thức thường dùng trong lớp Zend\_Db\_Table

**1- Thao tác liệt kê và nhận dữ liệu:**

Để có thể thực hiện các thao tác liệt kê dữ liệu đầy đủ và chi tiết. Chúng ta cần sử dụng phương thức:

```
1 $this->select();
```

Và từ phương thức này, chúng ta sẽ gọi các thao tác khác liên quan như điều kiện, giới hạn, sắp xếp,...

```
1 $query=$this->select();
```

**+ Liệt kê dữ liệu theo cột:**

```
1 $query->from('tên_bảng',array('cột 1','cột 2'));
```

**+ Liệt kê dữ liệu với một điều kiện:**

```
1 $query->where('cột =?', 'giá trị');
```

Cú pháp ở trên cho ta liệt kê dữ liệu với điều kiện cột bằng giá trị nào đó. Quy tắc trong zend framework đối với mệnh đề where là tên cột, rồi đến phép so sánh, rồi đến ký hiệu "?". Và sau cùng là mới là giá trị.

**+ Sắp xếp dữ liệu theo cột thuộc tính**

```
1 $query->order('tên_cột ASC hoặc DESC');
```

**+Giới hạn dữ liệu hiển thị**

```
1 $query->limit(vị trí bắt đầu, số record muốn hiển thị);
```

**+ Hiển thị tất cả thông tin:**

```
1 $this->fetchall();
```

Trường hợp hiển thị với các điều kiện ở phía trên thì ta truyền \$query vào fetchall()

```
1 $this->fetchall($query);
```

#### + Hiển thị 1 dòng dữ liệu

```
1 $this->fetchRow();
```

Ví dụ đầy đủ về hiển thị dữ liệu:

```
1 public function listuser() {
2     $data=$this->select();
3     $data->from('user',array('username','id'));
4     $data->where('id > ?',1);
5     $data->order('username DESC');
6     $data->limit(3);
7     $data=$this->fetchAll($data);
8     return $data;
9 }
```

## 2- Thao tác thêm, xóa, sửa dữ liệu:

#### + Thêm dữ liệu:

```
1 $this->insert($data);
```

Ví dụ:

```
1 public function insert_user($data) {
2
3     $this->insert($data);
4 }
```

#### + Sửa dữ liệu:

```
1 $this->update($data,$where);
```

Ví dụ:

```
1 public function update_user($data,$where) {
2     $where="id='1'";
3     $this->update($data,$where);
4 }
```

Với \$data của chúng ta là một mảng dữ liệu:

```
1 $data=array(
2     "username" => "kenny",
3     "password" => "12345",
4     "level"   => "2"
5 );
```

#### + Xóa dữ liệu:

```
1 $this->delete($where);
```

Ví dụ:

```
1 public function delinfo($id) {
2     $where="id=".$id;
3     $this->delete($where);
4 }
```

Kết thúc bài này, ta đã hiểu được cơ bản về cách thức sử dụng **Model** trong **Zend Framework**, đồng thời cũng nắm được phần nào về các phương thức phổ biến thường sử dụng trong ứng dụng. Trong bài kế tiếp, tôi sẽ hướng dẫn các bạn tìm hiểu về phương pháp tương tác với cơ sở dữ liệu trên lớp Zend\_Db. Qua đó có sự tùy biến tốt hơn trong việc viết ứng dụng trên **zend framework**.

## Zend Framework: Tương tác cơ sở dữ liệu với Zend\_Db

Trong bài trước, chúng ta đã tìm hiểu về quy trình **thao tác với cơ sở dữ liệu của lớp Zend\_Db\_Table**. Tuy nhiên, với những câu truy vấn phức tạp, đòi hỏi phải kết nhiều bảng thì rõ ràng Zend\_Db\_Table rất khó khăn trong việc thực hiện. Vì thế trong bài này, chúng ta sẽ cùng tìm hiểu thêm phương pháp tương tác cơ sở dữ liệu với các câu truy vấn tùy ý từ thư viện Zend\_Db trong **Zend Framework**.

Trước hết, ta cần tìm hiểu về lớp **Zend\_Registry**. Đây là thư viện tạo ra một giá trị toàn cục trong ứng dụng của chúng ta, nó có thể chứa giá trị, mảng và một đối tượng. Để thiết lập ta sử dụng cú pháp:

```
1 Zend_Registry::set('tên', 'giá trị');
```

Sau khi thiết lập, ta có thể lấy giá trị ở bất kỳ đâu bởi cú pháp:

```
1 Zend_Registry::get('tên');
```

Trong bài này, ta sử dụng chúng để lưu trữ hành động kết nối với cơ sở dữ liệu của chúng ta.

Tạo cơ sở dữ liệu và thêm dữ liệu vào bảng theo cú pháp sau:

```
01 CREATE TABLE user (  
02     id int(10) unsigned NOT NULL AUTO_INCREMENT,  
03     username varchar(50) NOT NULL,  
04     password char(32) NOT NULL,  
05     level int(1) NOT NULL DEFAULT '1',  
06     PRIMARY KEY (id)  
07 );  
08 INSERT INTO 'user' (username,password,level) VALUES('admin', '12345', 2);  
09 INSERT INTO 'user' (username,password,level) VALUES('kenny', '12345', 2);  
10 INSERT INTO 'user' (username,password,level) VALUES('jacky', '12345', 1);  
11 INSERT INTO 'user' (username,password,level) VALUES('Lena', '12345', 1);
```

Khai báo kết nối cơ sở dữ liệu tại file application/configs/application.ini như sau:

```
1 resources.db.adapter = "Pdo_mysql"  
2 resources.db.params.host = "localhost"  
3 resources.db.params.username = "root"  
4 resources.db.params.password = ""  
5 resources.db.params.dbname = "qhonline"
```

Tiếp tục, mở file bootstrap.php thêm đoạn code sau:

```
1 <?php  
2 class Bootstrap extends Zend_Application_Bootstrap_Bootstrap{  
3     protected function _initDatabase(){  
4         $db = $this->getPluginResource('db')->getDbAdapter();  
5         Zend_Registry::set('db', $db);  
6     }  
7 }
```

Theo đoạn code trên, ta thêm vào phương thức initDatabase(). Đây là phương thức sẽ được triệu gọi khi chạy ứng dụng. Tại đây, ta lấy thông tin kết nối từ nội dung đã cấu hình ở file application.ini.

Kế tới, ta thực hiện việc tạo registry để lưu trữ đối tượng kết nối này. Tại thư mục Models, ta tạo file tên User.php. Với nội dung lớp này như sau:

```
01 <?php  
02 class Model_User{  
03     protected $db;
```

```

04     public function __construct() {
05         $this->db=Zend_Registry::get('db');
06     }
07     public function listall() {
08         $sql=$this->db->query("select * from user order by id DESC");
09         return $sql->fetchAll();
10     }
11 }

```

Như bạn thấy, tại lớp `Model_User` này trước hết ta tạo một `construct()` lấy giá trị từ registry để đưa vào thuộc tính tên `$db`. Để mỗi khi ta khởi tạo đối tượng thì đối tượng kết nối đã được tạo sẵn.

Kế đến ta tạo phương thức `listall()` thực hiện công việc liệt kê toàn bộ người dùng có trong bảng `user`. Lúc này, để thực hiện được câu truy vấn thì ta dùng phương thức `query()` chạy. Và lấy kết quả bởi phương thức `fetchAll()`.

Tiếp tục, tại controller `User` ta gọi model như sau:

```

01 <?php
02 class UserController extends Zend_Controller_Action{
03     public function indexAction() {
04         $muser=new Model_User;
05         $data=$muser->listall();
06         echo "<pre>";
07         print_r($data);
08         echo "</pre>";
09     }
10 }

```

Đoạn code trên có tác dụng gọi model `user` và gọi tiếp phương thức `listall()` của model để hiển thị thông tin của người dùng bởi các thẻ thử nghiệm bên dưới.

Với phương pháp tương tác này, rõ ràng chúng ta có thể chạy bất cứ câu lệnh nào mà ta muốn trong ứng dụng một cách dễ dàng. Đồng thời, có thể tạo ra những tùy biến kết hợp giữa **Zend\_Db\_Table** và **Zend\_Db**. Chẳng hạn, với câu truy vấn đơn giản, ta có thể sử dụng **Zend\_Db\_Table** để lấy giá trị. Nhưng với các câu truy vấn phức tạp đòi hỏi phải kết bảng, xử lý nhiều, thì việc dùng cách tương tác **Zend\_Db** lại là sự lựa chọn hiệu quả nhất.

```

01 <?php
02 class Model_User extends Zend_Db_Table_Abstract{
03     protected $_name="user";
04     protected $_primary="id";
05     protected $db;
06     public function __construct() {
07         $this->db=Zend_Registry::get('db');
08     }
09     public function listall() {
10         $sql=$this->db->query("select * from user where level='1' order
    by id DESC");
11         return $sql->fetchAll();
12     }
13     public function listall2() {

```

```

14         $query=$this->select();
15         $query->where('level =?', '2');
16         return $this->fetchAll($query);
17     }
18 }

```

Ở trên là một ví dụ về sự kết hợp cả hai, với `listall()` là phương thức sử dụng câu truy vấn thuần. Trong khi `listall2()` ta sử dụng mô hình `active record` để lấy dữ liệu.

Kết thúc bài này, ít nhiều chúng ta đã có thể định hình và hiểu rõ hơn về cơ chế tương tác cơ sở dữ liệu trong **zend framework**. Qua đó có thể vận dụng linh hoạt chúng trong thực tế để giải quyết các vấn đề về cơ sở dữ liệu. Trong bài tiếp theo tôi sẽ hướng dẫn các bạn tìm hiểu và sử dụng lớp **Zend\_Paginator** để phân trang dữ liệu cho ứng dụng.

## Zend Framework: Hướng dẫn sử dụng layout trong ứng dụng

Trong bài trước, chúng ta đã nói về **quy trình hoạt động của zend view**. Và qua đó, ta hoàn toàn kiểm soát được việc dữ liệu của view hiển thị thế nào với các thẻ như `doctype`, `meta`, `title`, `link css`, `link js`. Tiếp tục ở phần này, tôi sẽ trình bày phương pháp sử dụng `layout` (hay còn gọi là `master page`) trong ứng dụng như thế nào.

Trước hết, ta cần hiểu về quy trình sử dụng `layout` trong **zend framework**.

Khi sử dụng **layout**, thì mọi dữ liệu của view đều sẽ hiển thị trong vùng chỉ định của `layout` đó mà chúng ta không cần phải khai báo chúng trong từng view một. Như vậy, nói cách khác chúng ta sẽ không cần quan tâm đến bộ cục của **layout** khi làm việc với view. Mà chỉ quan tâm đến dữ liệu, vai trò mà view đó thể hiện như thế nào mà thôi.

Để làm được điều đó, trong file *application.ini* ta thêm vào 2 dòng sau:

```

1 resources.layout.layout="layout"
2 resources.layout.layoutPath=APPLICATION_PATH "/layouts/scripts"

```

Ý nghĩa của 2 dòng này như thế nào?  
 + Dòng đầu tiên cho ta biết file chứa **layout** của chúng ta tên là `layout`  
 + Dòng thứ hai cho ta biết đường dẫn tới file **layout** này như thế nào.

Với 2 dòng ở trên ta hoàn toàn có thể cấu hình thay đổi tên và đường dẫn theo ý thích của riêng ta một cách dễ dàng.

Kế tới, ta tạo thư mục `layouts/scripts`. Và tạo file `layout.phtml` trong thư mục này với nội dung như sau:

```

01 <?php echo $this->doctype() ?>
02 <html>
03 <head>
04     <?php echo $this->headTitle() ?>
05     <?php echo $this->headMeta() ?>
06     <?php echo $this->headLink() ?>
07     <?php echo $this->headScript() ?>
08 </head>
09 <body>
10     <?php
11         echo $this->layout()->content;
12     ?>
13 </body>
14 </html>

```

Bạn thấy đấy, cấu trúc của nó giống với phần view ở bài **tìm hiểu cơ bản về zend view** phải không nào. Chỉ có sự khác biệt là phần nội dung `body`. Chúng ta dùng cú pháp:

```

1 $this->layout()->content;

```

Cú pháp này sẽ gọi các view của controller vào để hiển thị trên bố cục này.

Vậy trong `IndexController.php` ta viết đoạn lệnh như sau:

```
1 <?php
2 class IndexController extends Zend_Controller_Action{
3     public function init(){
4         $this->view->headTitle("QHOnline - Zend Layout")
5     }
6     public function indexAction(){
7         echo "<h1>Welcome to Zend Framework - QHOnline.Info";
8     }
9 }
```

Ở đoạn code phía trên, tôi sử dụng phương thức `init()`. Đây là phương thức mặc định sẽ được gọi đầu tiên khi chúng ta tiến hành chạy controller này. Và dĩ nhiên, khi bạn chạy bất kể một action nào thì `init()` cũng sẽ được gọi trước hết. Và trong ví dụ này, công việc của `init()` là hiển thị thông tin tiêu đề của trình duyệt là **QHOnline-Zend Layout**.

Khi bạn chạy ứng dụng, sẽ thấy view của chúng ta được nằm trong một bố cục mà bạn đã bố trí một cách hợp lý.

Ngoài cách cấu hình trực tiếp trong file **application.ini** ra, chúng ta cũng có thể cấu hình và gọi trực tiếp chúng bởi việc gọi nạp trong các controller như sau:

```
01 <?php
02 class IndexController extends Zend_Controller_Action{
03     public function init(){
04         $option=array(
05             "layout" => "layout",
06             "layoutPath" => APPLICATION_PATH."/layouts/scripts/"
07         );
08         Zend_Layout::startMvc($option);
09         $this->view->headTitle("QHOnline - Zend Layout")
10     }
11     public function indexAction(){
12         echo "<h1>Welcome to Zend Framework - QHOnline.Info";
13     }
14 }
```

Như bạn thấy đấy, ở trên tôi tạo ra một mảng chứa 2 tham số là `layout` để chứa tên file và đường dẫn của layout nhằm gọi tới thư mục chứa layout. Sau đó tôi gọi lớp **Zend\_Layout** đồng thời gọi phương thức **startMvc()** để truyền tham số mà chúng ta vừa thiết lập trong mảng.

Khi chạy ứng dụng thì kết quả hiển thị so với phương pháp cấu hình trong `application` là như nhau.

Vậy sự khác biệt giữa hai phương pháp cấu hình này là gì ?.

+ Phương pháp cấu hình bằng `application.ini` cho phép ta chỉ thiết lập một lần, sau đó các controller không cần phải thiết lập lại. Nhưng chúng chỉ cố định duy nhất một layout cho tất cả ứng dụng.

+ Phương pháp cấu hình trực tiếp tuy phải cấu hình trong từng controller nhưng có thể cho ta tùy chọn thay đổi theo chiều hướng multi layout cho từng module hoặc controller mà ta muốn một cách dễ dàng.

Sử dụng 2 phương pháp này thế nào và ra sao là còn tùy vào sự linh động và sáng tạo trong lập trình của các bạn. Kết thúc bài này, tôi đã hướng dẫn bạn sử dụng layout trong **zend framework** để tạo master page đồng thời hướng tới khái niệm multi layout cho ứng dụng với những tùy chọn mà bạn mong muốn. Trong bài tới,

chúng ta sẽ cùng tìm hiểu về tương tác cơ sở dữ liệu trong **zend framework** ra sao, qua đó vận dụng model trong **zend framework** như thế nào.

## Zend Framework: Hướng dẫn cấu hình ứng dụng theo mô hình module

Ở bài trước, chúng ta đã tìm hiểu về nguyên tắc xây dựng và mô tả cơ bản về kiến trúc **zend framework**. Tiếp tục trong bài này, tôi sẽ trình bày kỹ thuật cấu hình ứng dụng theo mô hình đa module (multi module). Việc cấu hình này giúp ứng dụng trở nên rõ ràng và dễ phát triển hơn rất nhiều so với cách trình bày mặc định của **zend framework**.

Nếu bạn vẫn chưa thể thực hiện được cách cấu hình mặc định của **zend framework**. Vui lòng xem lại 2 bài [Hướng dẫn cài đặt và cấu hình ứng dụng đầu tiên](#), tìm [hiểu quy trình làm việc trong zend framework](#).

Trước tiên, ta xét lại cấu trúc mặc định của hệ thống qua tấm hình bên dưới:

Như chúng ta thấy, với cấu hình ở trên thì trong application có 3 thư mục chủ đạo là controllers, models, views. Vậy để tạo ứng dụng theo mô hình module ta tạo 1 thư mục modules. Trong thư mục này ta tạo tiếp 2 module là default và admin theo cấu trúc:

```
zf2/application/modules/default
```

```
zf2/application/modules/admin
```

Tại mỗi module default và admin ta lại tạo tiếp 3 thư mục con là controllers, models, views. Theo cấu trúc.

### Module default:

```
zf2/application/modules/default/controllers
```

```
zf2/application/modules/default/models
```

```
zf2/application/modules/default/views
```

### Module admin:

```
zf2/application/modules/admin/controllers
```

```
zf2/application/modules/admin/models
```

```
zf2/application/modules/admin/views
```

Tại các thư mục controllers này lần lượt tạo các file tên IndexController.php với nội dung như sau:

File IndexController.php của module default

```
1 <?php
2 class IndexController extends Zend_Controller_Action{
3     public function indexAction() {
4     }
5
6 }
```

File IndexController.php của module admin

```
1 <?php
2 class Admin_IndexController extends Zend_Controller_Action{
3     public function indexAction() {
4     }
5
6 }
```

Nếu để ý kỹ, chúng ta có thể thấy rằng trước tên mỗi class ta lại thêm tên module của chúng ở trước. Đây là kỹ thuật lazy loading trong OOP, được sử dụng để triệu nạp file controller trong từng module của **zend framework**.



Riêng đối với module default là module mặc định nên trong class controller ta không cần khai báo thêm tên module giống với module admin.

Tiếp tục, ta di chuyển 2 file index.php và .htaccess ra khỏi thư mục public theo cấu trúc như sau:

www/zf2/index.php

www/zf2/.htaccess

Vì chúng ta đã di chuyển file index.php ra khỏi thư mục public và nó ngang cấp với thư mục application nên lúc này đường dẫn triệu nạp trong file cũng sẽ thay đổi. File index.php cũ của chúng ta:

```
01 <?php
02 define('APPLICATION_PATH',
03         realpath(dirname(__FILE__) . '/../application'));
04 define('APPLICATION_ENV', 'production');
05 set_include_path(dirname(dirname(__FILE__)) . '/library');
06 require_once 'Zend/Application.php' ;
07 $application = new Zend_Application(
08     APPLICATION_ENV,
09     APPLICATION_PATH . '/configs/application.ini'
10 );
11 $application->bootstrap()->run();
```

Vì giờ file index.php đã đưa ra ngoài public. Nên chắc chắn đường dẫn vào thư mục application sẽ thay đổi. Khi đó sẽ là: realpath(dirname(\_\_FILE\_\_) . '/application');

Vậy hằng APPLICATION\_PATH cũng chúng ta đã thay đổi và trở đường dẫn tới thư mục application.

Tiếp tục ta cấu hình cho ứng dụng tìm tới được thư mục library.

set\_include\_path(APPLICATION\_PATH . '/../library');

Bạn hiểu thế nào về đoạn cấu hình này ? Vì hằng APPLICATION\_PATH ở trên đã có thể tìm thấy được thư mục application. Cho nên từ thư mục ấy ta back trở ra để tìm tới thư mục library. Vì thế trước library ta có dùng "../library" là vì vậy.

Vậy file index.php cũng chúng ta sau khi chỉnh sửa sẽ như sau:

```
01 <?php
02 define('APPLICATION_PATH',
03         realpath(dirname(__FILE__) . '/application'));
04 define('APPLICATION_ENV', 'production');
05 set_include_path(APPLICATION_PATH . '/../library');
06 require_once 'Zend/Application.php' ;
07 $application = new Zend_Application(
08     APPLICATION_ENV,
09     APPLICATION_PATH . '/configs/application.ini'
10 );
11 $application->bootstrap()->run();
```

Tiếp tục ta tạo file index.phtml trong từng thư mục views của từng module.

www/zf2/application/modules/default/views/scripts/index/index.phtml

```
1 <h1>Hello Zend Framework - Default Module</h1>
```

www/zf2/application/modules/admin/views/scripts/index/index.phtml

```
1 <h1>Hello Zend Framework - Admin Module</h1>
```

Chạy thử ứng dụng xem nào:

```
http://localhost/zf2/default
http://localhost/zf2/admin/index
```

Bị lỗi rồi phải không nào ?. Ứng dụng bị lỗi là vì chúng ta vẫn chưa cấu hình để tìm thấy được thư mục module.

Vì thế, để giải quyết. Tả mở file application.ini trong thư mục configs ra và thêm vào 2 dòng sau:

```
1 resources.frontController.moduleDirectory=APPLICATION_PATH "/modules"
2 resources.modules=""
```

Dòng ở trên làm gì vậy ?. Câu trả lời nó chỉ ra đường dẫn tới thư mục modules của chúng ta. Và dòng thứ 2 chỉ ra ta đang gọi cơ chế module autoload. Khi đó các class, model, form,... sẽ được tự động nạp vào từng module của chúng ta.

Chạy xem lại xem nào.

Kết quả sẽ như thế này đây

Hình ảnh cấu trúc multi Module trong Zend Framework (file .htaccess không hiển thị trong mô hình cây này).

Download mã nguồn của bài học [tại đây](#)

### Zend Framework: Tìm hiểu cơ bản về Zend\_Form

Ở bài trước, tôi đã hướng dẫn các bạn [tìm hiểu về Zend\\_Paginator](#). Qua đó sử dụng chúng để thực hiện phân trang cho ứng dụng. Tiếp theo bài này, chúng ta sẽ cùng tìm hiểu về **Zend\_Form**. Một trong những lớp khá tiện dụng trong **Zend Framework** nhằm thực hiện tạo ra các thành phần tương tác trong một form dữ liệu.

Để thao tác được với **zend form**. Trước hết, ta tạo một thư mục forms trong application và trong thư mục này ta tạo file User.php với nội dung như sau:

```
01 <?php
02 class Form_User extends Zend_Form{
03     public function init(){
04         $this->setAction('')->setMethod('post');
05         $name=$this->createElement("text","name",array(
06             "label" => "Full Name",
07             "size" => "30",
08         ));
09         $email=$this->createElement("text","email",array(
10             "label" => "Email",
11             "size" => "30",
12         ));
13         $gender=$this->createElement("radio","gender",array(
14             "label" => "Gender",
15             "multioptions"=> array(
16                 "1" => "Male",
17                 "2" => "Female",
18             )
19         ));
20         $country=$this->createElement("select","country",array(
21             "label" => "Country",
```

```

22             "multioptions"=> array(
23                 "1" => "VietNam",
24                 "2" => "Cambodia",
25                 "3" => "Thai Lan",
26             )
27         ));
28         $note=$this->createElement("textarea","note",array(
29             "label" => "Note",
30             "cols"  => "30",
31             "rows"  => "5",
32         ));
33         $submit=$this->createElement("submit","submit");
34         $this->addElements(
35             array($name,$email,$gender,$country,$note,$submit)
36         );
37     }
38 }

```

Ở đoạn code trên ta tạo ra lớp tên `Form_User` tức là chỉ ra đường dẫn từ thư mục forms tới file user.php. Tại lớp này ta lại có phương thức `init()`. Đây là phương thức sẽ được gọi trước tiên khi chúng ta khởi tạo lớp `Form_User`.

Vì lớp `Form_User` của chúng ta kế thừa lớp **Zend\_Form** nên trong lớp này chúng ta có thể sử dụng từ khóa `$this` để gọi các phương thức của lớp **Zend\_Form**.

Tại đây ta khai báo action và method bằng phương thức `setAction()` và `setMethod()`. Kế tới ta tạo các thành phần trong form như text, radio, select, textarea, submit. Bởi phương thức:

```
1 $this->createElement("Thành Phần", "Tên", "Thuộc tính thêm")
```

+ Thành phần là: text, radio, select, textarea, checkbox, submit.  
+ Tên là tên của form mà bạn muốn gán vào.  
+ Thuộc tính thêm là những phần tử ta muốn thêm vào như label, size, cols, rows,...  
Riêng với thành phần radio, select ta dùng thêm `multioptions` để biểu diễn các giá trị của chúng bên trong form.  
Sau cùng ta dùng phương thức

```
1 $this->addElements($Mảng các phần tử)
```

Để thêm chúng vào trong ứng dụng.

Giống với phần **làm việc với zend\_db\_table** vậy, để **zend framework** có thể hiểu được lớp `Form_User`. Chúng ta phải cấu hình trong file `bootstrap.php` của thư mục application thông tin như sau:

```

01 <?php
02 class Bootstrap extends Zend_Application_Bootstrap_Bootstrap{
03     protected function _initAutoload() {
04         $autoloader = new Zend_Application_Module_Autoloader(array(
05             'namespace' => '',
06             'basePath' => dirname(__FILE__),
07         ));
08         return $autoloader;
09     }
10 }

```

Sau khi đã hoàn tất file cấu hình này. Kế tới ta mở file `UserController.php` trong thư mục `application/controllers` và thêm vào nội dung gọi form như sau:

```
1 <?php
```

```

2 class UserController extends Zend_Controller_Action{
3     public function indexAction()
4     {
5         $form=new Form_User;
6         $this->view->form=$form;
7     }
8 }

```

Vì ta đã truyền các thành phần của form ở trên vào view form. Do vậy, ta cần xuất các thành phần này trong view để hiển thị cho người dùng thấy. Mở file index.phtml theo đường dẫn views/scripts/user/ và đưa vào đoạn lệnh bên dưới:

```

1 <?php
2 echo $this->form;
3 ?>

```

Chạy ứng dụng: <http://localhost/zform/user/>  
Và kết quả sẽ hiển thị:

Bạn thấy đấy, zend form cho ta kết quả như mong đợi. Tuy nhiên, **Zend Form** cũng tích hợp sẵn một số đặc điểm khi thể hiện form. Như khái niệm về các thẻ <dt>, <dd> xuất hiện trong **HTML 5** cũng được đưa vào sử dụng. Vậy làm thế nào để có thể tùy biến các thẻ trong zend form để dạng nào ?.

Muốn tùy biến trong **zend form**, ta sử dụng thêm một khái niệm nữa là decorator trong **zend form**. Phương thức này cho phép ta thêm thắt và thay đổi các định dạng của thành phần trong form theo cách của riêng ta.

Vậy trước khi ta thực hiện việc addElements ta cần khai báo như sau:

```

1 $this->setDecorators(array(
2     array('viewScript',
3         array('viewScript'=>'Form_Register.phtml'),
4     ));

```

Vì ở trên ta có khai báo cho các phần tử đều tồn tại một label. Và chúng được bọc trong cặp thẻ <dt>. Để xóa bỏ nó, ta cần bỏ các dòng khai báo label ở trên của các phần tử.

Tiếp tục ta thêm vào đoạn code sau để xóa bỏ các thẻ <dd> và <dt> bọc lấy các phần tử form.

```

01 $name->removeDecorator('HtmlTag')
02     ->removeDecorator('Label');
03 $email->removeDecorator('HtmlTag')
04     ->removeDecorator('Label');
05 $gender->removeDecorator('HtmlTag')
06     ->removeDecorator('Label');
07 $country->removeDecorator('HtmlTag')
08     ->removeDecorator('Label');
09 $note->removeDecorator('HtmlTag')
10     ->removeDecorator('Label');
11 $submit->removeDecorator('DtDdWrapper');

```

Phương thức **removeDecorator('HtmlTag')** cho phép ta xóa bỏ cặp thẻ <dd> trong các phần tử khi chúng phát sinh.

Phương thức **removeDecorator('Label')** cho phép ta xóa bỏ thẻ <dt>. Vì mặc dù ta đã xóa bỏ label ở phía trên nhưng khi hiển thị thì cặp thẻ <dt> vẫn chưa mất hoàn toàn. Do vậy cần có thêm phương thức này để xóa bỏ hoàn toàn thẻ <dt>.

Riêng ở nút submit thì do không có thẻ <dt> nên để định dạng nó ta phải sử dụng phương thức: **removeDecorator('DtDdWrapper');**

Sau khi đã khai báo và cấu hình xong. Kế tới ta tạo file Form\_Register.phtml tại thư mục views/scripts với nội dung:

```
01 <form action='<?php echo $this->element->getAction(); ?>'
02 method= '<?php echo $this->element->getMethod(); ?>'>
03 <p>Your Name:<br />
04 <?php echo $this->element->name; ?>
05 </p>
06 <p>Your Email:<br />
07 <?php echo $this->element->email; ?>
08 </p>
09 <p>Your Gender:<br />
10 <?php echo $this->element->gender; ?>
11 </p>
12 <p>Your Country:<br />
13 <?php echo $this->element->country; ?>
14 </p>
15 <p>Your Note:<br />
16 <?php echo $this->element->note; ?>
17 </p>
18 <p><?php echo $this->element->submit; ?>
19 </p>
20 </form>
```

Đây là trang đưa các thành phần trong form ra bên ngoài. Tại đây để lấy thông tin action hoặc method, ta dùng các phương thức như `getAction()`, `getMethod()`,... Và để lấy các thành phần khác trong form ta dùng `$this->element->Tên` mà ta đã khai báo trong lớp tạo form.

Cuối cùng chạy lại ứng dụng: <http://localhost/zfform/user>

Kết quả như ta mong đợi phải không nào.

Để lấy dữ liệu hoặc kiểm tra thông tin từ form tại `UserController.php` ta sửa `indexAction()` như sau:

```
01 <?php
02 class UserController extends Zend_Controller_Action{
03     public function indexAction() {
04         $form=new Form_User;
05         if($this->_request->isPost()) {
06             $name=$this->_request->getPost('name');
07             //Lấy các tham số còn lại
08         }
09         $this->view->form=$form;
10     }
11 }
```

Để kiểm tra xem người sử dụng có truyền dữ liệu từ form hay không ?. Ta dùng phương thức:

```
1 $this->_request->isPost()
```

Và để lấy giá trị từ form ta dùng phương thức:

```
1 $this->_request->getPost('name')
```

Tài toàn bộ mã nguồn của bài học tại đây.

Như vậy, ở phần này. Tôi đã hướng dẫn các bạn tìm hiểu quy trình làm việc trên **zend form**. Qua đó áp dụng nó

để xây dựng mẫu biểu tương tác người dùng cho ứng dụng. Thực tế thì chúng ta không nhất thiết phải sử dụng zend form. Có khi chỉ cần dùng HTML thuần trong view thì cũng đã tạo được form rồi. Nhưng lợi thế của việc sử dụng **zend\_form** được thể hiện rõ ở phần validation, filter, captcha...Nghĩa là chúng ta sử dụng **zend\_form** kết hợp với vấn đề kiểm tra tính hợp lệ trên dữ liệu dựa vào các lớp được xây dựng sẵn của **zend framework**.

Tiếp tục ở bài kế tới, chúng ta sẽ tìm hiểu cách sử dụng **zend\_validation** trong zend form như thế nào. Qua đó ta hoàn toàn có thể xây dựng những mẫu biểu tương tác với người dùng một cách dễ dàng.

(Bùi Quốc Huy)