

***PHP nâng cao   
với Zend framework 2***

# TỔNG QUAN ZEND FRAMEWORK 2

**Nội dung:**

Tổng quan Zend Framework 2

* Giới thiệu ZF2
* Lý do chọn ZF2
* Zend MVC
* Cài đặt và cấu hình ứng dụng ZF2

## Giới thiệu ZF

Zend Framework là một PHP Framework đuợc lập trình bằng ngôn ngữ PHP và dựa trên mô hình MVC, giúp tách các thành phần xử lý ra một cách rõ ràng cho website của bạn, giúp cho code được dễ dàng quản lý, chỉnh sửa và nâng cấp.

* Sử dụng PHP 5
* Viết huớng đối tuợng
* Cung cấp mô hình tiên tiến MVC
* Là 1 phần mềm nguồn mở có sự hỗ trợ cộng đồng
* Cung cấp 1 loạt các ứng dụng API của các nhà cung cấp hàng đầu như Google, Yahoo, Flick...

***Sử dụng ZF để…***

* Tạo ứng dụng web theo mô hình chuẩn MVC
* Url chuẩn, ngắn gọn
* Hỗ trợ phân quyền tới từng Action
* Có các thành phần thư viên hỗ trợ API của các nhà cung cấp như Google, Yahoo, Flick
* Quản lý code dễ dàng, liệt kê và lấy các truy vấn history
* Dễ dàng phát triển thêm các ứng dụng nhúng, sử dụng Plugins

***Lý do chọn ZF***

* Có rất nhiều thư viện được hỗ trợ sẵn
* Là một PHP Framework được do chính công ty phát triển ngôn ngữ PHP phát triển ([www.zend.com](http://www.zend.com/)).
* Được xây dựng trên nền tảng lập trình hướng đối tượng nên có thể thừa kế, nâng cấp các ứng dụng một cách dễ dàng.
* Những thành viên phát triển Zend Framework là những chuyên gia nổi tiếng về mã nguồn mở và PHP trên thế giới.
* Mô hình MVC hoặc không theo chuẩn MVC trong Zend Framework rất linh hoạt giúp chúng ta có thể tùy biến cấu hình ứng dụng theo ý muốn.
* Tích hợp các thư viện, các thành phần khác một cách linh hoạt và nhanh chóng.
* Có tài liệu để tham khảo
* Cộng đồng người sử dụng Zend Framework lớn.
* Những ứng dụng lớn hiện nay đã dần chuyển qua dùng Zend Framework làm nền tảng.
* Nhu cầu tuyển dụng lập trình viên biết Zend Framework lớn
* Bảo mật của các ứng dụng rất tốt nên tránh được các lỗi thường gặp trong các ứng dụng viết bằng PHP thường.

## Giới thiệu ZF2

Zend Framework 2 là một framework mã nguồn mở để phát triển các ứng dụng web và các dịch vụ sử dụng PHP 5.3 +.

Zend Framework 2 sử dụng mã hướng đối tượng 100% và sử dụng hầu hết các tính năng mới của PHP 5.3, cụ thể là namespace, late static ràng buộc, hàm lambda và bao đóng.

Các thành phần của Zend Framework 2 là duy nhất, mỗi thành phần được thiết kế với ít phụ thuộc vào các thành phần khác. ZF2 tuân theo nguyên tắc thiết kế trọng tâm hướng đối tượng. Kiến trúc này cho phép các nhà phát triển sử dụng bất cứ thành phần mà họ muốn. Chúng ta gọi đây là sử dụng theo ý muốn "use-at-will" thiết kế.

Trong khi chúng có thể được sử dụng riêng rẽ, Zend Framework 2 cung cấp thư viện chuẩn ứng dụng web mạnh mẽ và mở rộng khi kết hợp.

Ngoài ra, nó cung cấp mạnh mẽ, hiệu suất cao thực hiện MVC, một trừu tượng cơ sở dữ liệu là đơn giản để sử dụng, và các phần tử HTML5 cung cấp, xác nhận, và sàng lọc để phát triển có thể củng cố tất cả các hoạt động dễ sử dụng, đối tượng giao diện theo định hướng.

Các thành phần khác, chẳng hạn như Zend\Authentication và Zend\Permissions\Acl, cung cấp xác thực người dùng và ủy quyền đối với tất cả các ủy nhiệm lưu trữ chung.

Nhà tài trợ chính của dự án Zend Framework 2 là Zend Technologies, nhiều công ty đã đóng góp thành phần hoặc các tính năng quan trọng cho framework. Các công ty như Google, Microsoft, và có StrikeIron hợp tác với Zend để cung cấp giao diện cho các dịch vụ web và các công nghệ khác mà họ muốn thực hiện có sẵn cho nhà phát triển Zend Framework 2.

**Đặc điểm ZF2**

* Cấu trúc mới:
* Theo mô hình MVC, sử dụng Di, Events, Service, Module
* Cải tiến về chức năng
* Sử dụng PHP 5.3.3
* PSR – 2 compliant (hỗ trợ pyrus, composer)
* Hệ thống đóng gói

ZF được viết theo kiểu OOP nên nó thừa hưởng các thế mạnh của kiểu viết này. Các lớp của ZF được BA (Business Analysis) rất chuẩn và khi cần mở rộng ta có thể dùng tính chất thừa kế của OOP. Nói chung là ta không phải chỉnh sửa nội dung chính của ZF.

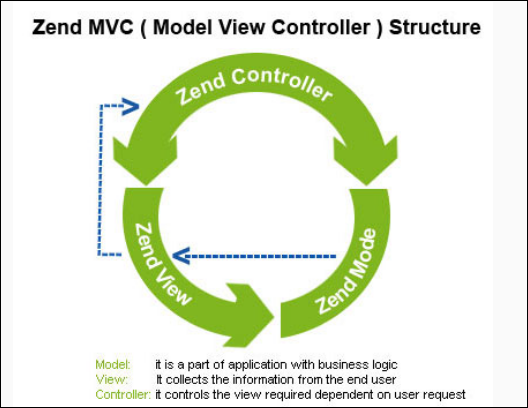
Hầu như các version mới của ZF không có nhiều thay đổi nên ta có thể dễ dàng update.

ZF tích hợp được gần như tất cả các thư viện PHP và các CMS khác để sử dụng. VD: Smarty - Pear - FCKEditor - Drupal …

Cách viết của ZF rất thân thiện và đơn giản. Tích hợp những mới nhất của lập trình web như: JSON - Search - Syndication - Web Services...

ZF được sử dụng trong các dự án lớn và có kế hoạch phát triển dài lâu

## Kiến trúc MVC của ZF2



## Zend MVC

Zend\Mvc là một nhánh thực hiện MVC mới được thiết kế cho ZF2, tập trung vào hiệu suất và tính linh hoạt.

**MVC layer được xây dựng ở phía trên của các thành phần sau:**

* Zend\ServiceManager – ZF2 cung cấp một bộ các định nghĩa dịch vụ mặc định được thiết lập ở Zend\Mvc\Service. ServiceManager tạo và cấu hình instance và workflow trong ứng dụng.
* Zend\EventManager – MVC là một event driven. Thành phần này được sử dụng mọi nơi từ khởi tạo ban đầu của ứng dụng, thông qua việc gọi trả về của response và request, để thiết lập và nhận các định tuyến và các định tuyến phù hợp (tương ứng), cũng như render các view.
* Zend\Http – đặc biệt là các đối tượng request và response, được sử dụng trong:
* Zend\Stdlib\DispatchableInterface. Tất cả các “controller” đơn giản là đối tượng dispatchable.

**Trong MVC layer, có một số thành phần con được giới thiệu:**

* Zend\Mvc\Router chứa các lớp liên quan đến định tuyến một request. Nói cách khác, nó nối các request tới các controller tương ứng của nó (hoặc dispatchable).
* Zend\Http\PhpEnvironment cung cấp một bộ decorator cho các đối tượng HTTP Request và Response nhằm đảm bảo là các request được đưa vào môi trường hiện tại (bao gồm các query parameter, POST parameter, HTTP header..)
* Zend\Mvc\Controller, một bộ các lớp “controller” trừu tượng với các trách nhiệm cơ bản như hệ thống các event, action gửi đi…
* Zend\Mvc\Service cung một bộ các yếu tố và định nghĩa ServiceManager workflow của ứng dụng mặc định.
* Zend\Mvc\View cung cấp các thông số mặc định cho renderer selection, độ phân giải view script, đăng ký helper…; ngoài ra, nó cung cấp các listener liên lạc với MVC workflow, cung cấp các tính năng như phân giải tên template tự động, tạo view model tự động...

**Gateway của MVC là đối tượng Zend\Mvc\Application (gọi tắt là “Application”).**

Nhiệm vụ chính của “Application” là khởi tạo các tài nguyên, định tuyến request, nhận và gửi controller phù hợp trong các quá trình định tuyến. Sau khi hoàn thành, nó sẽ render view, hoàn thành các request, trả về và gửi response.

## Cài đặt

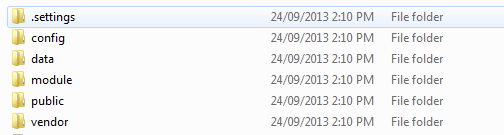
Các bước thực hiện

Download zend framework 2.x tại <http://framework.zend.com/downloads/latest>

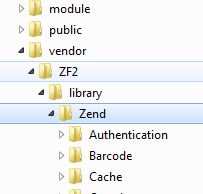
Download ứng dụng mẫu (Skeleton Application) của zend framework 2.x tại <https://github.com/zendframework/ZendSkeletonApplication>

Giải nén ứng dụng mẫu vào thư mục lưu trữ web

Cấu trúc cây thư mục ứng dụng mẫu:

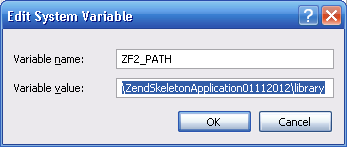


Copy thư mục library vào thư mục vendor

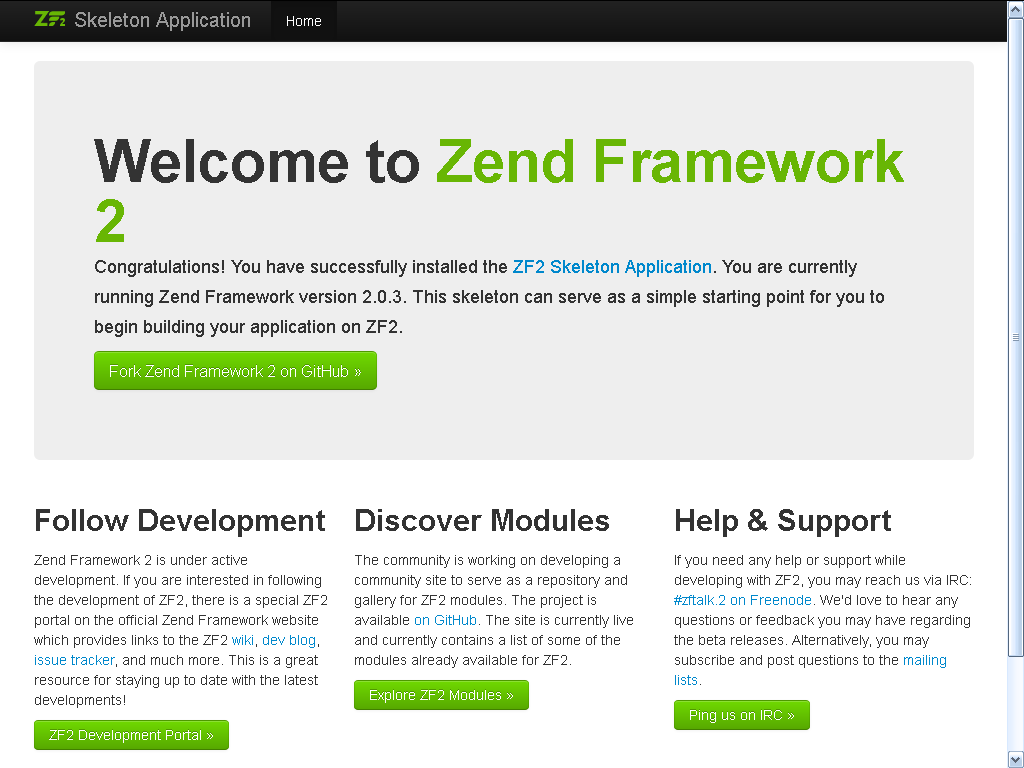


***Chú ý:*** *Nếu muốn sử dụng thư viện một Zend cho tất cả các ứng dụng thì cần phải tạo ra một biến toàn cục như sau:*

Vào Computer => Properties => Advanced => Environment Variables => New => Tạo biến mới => OK



Kết quả sau khi cài đặt và cấu hình thành công



Cấu trúc cơ bản của ứng dụng

Tập tin ***public/index.php***: gồm tất cả các yêu cầu (request) của người dùng đến website, nhận một mảng các cấu hình đặt tại tập tin config/application.config.php, khi trả về nó gọi phương thức run() của “Application”, xử lý yêu cầu và gửi lại phản hồi cho người dùng.

Thư mục ***config*** được sử dụng để quản lý cấu hình ứng dụng.

Thư mục ***vendor*** : Dùng để quản lý thư viện thông thường được sử dụng bởi các ứng dụng. thư viện Zend Framework cũng được cài đặt trong thư mục này

Thư mục ***module***: chứa một hoặc nhiều module cung cấp các chức năng của ứng dụng

Thư mục **module/Application** đây là module ứng dụng mặc định được cung cấp với ZendSkeletonApplication.

Thư mục **Public** chứa các tài liệu chung như CSS, Hình ảnh, JavaScript.

Bên trong thư mục Public, bạn cũng có thể tìm thấy tập tin .htaccess. Mục đích chính của nó là để xác định viết lại URL, nhưng bạn cũng có thể sử dụng nó để xác định quy tắc truy cập cho trang web của bạn. Ví dụ, với htaccess. Bạn có thể cấp quyền truy cập vào trang web của bạn từ địa chỉ IP của riêng mình mà thôi, hoặc sử dụng ủy quyền HTTP để yêu cầu người dùng nhập tên người dùng và mật khẩu.

# Module

**Nội dung: Zend Module**

* Zend Module
* Zend Module Manager
* Zend Module Autoloader
* Cấu trúc của module trong ZF2
* Namespaces
* MVC layer

## Zend Module

Trong Zend Framework, một module có thể được định nghĩa là một đơn vị của phần mềm đó là có thể mang theo và tái sử dụng và có thể được kết nối với nhau đến các module khác để xây dựng một ứng dụng phức tạp lớn hơn.

Module không phải là mới trong Zend Framework, nhưng với ZF2, có một đại tu hoàn chỉnh trong cách modules sử dụng trong Zend Framework. Với ZF2, module có thể được chia sẻ trên các hệ thống khác nhau, và chúng có thể được đóng gói và phân phối tương đối dễ dàng. Một trong những thay đổi lớn khác đi vào ZF2 là ngay cả những ứng dụng chính hiện nay được chuyển đổi thành một Module, đó là module application.

Một trong những lợi thế chính của modules Zend Framework 2.0 liệt kê như sau:

* Khép kín, di động, tái sử dụng
* Phụ thuộc quản lý
* Trọng lượng nhẹ và nhanh chóng
* Hỗ trợ cho Phar đóng gói và phân phối Pyrus

**Hệ thống module được tạo thành như sau:**

* Module Autoloader
* Module Manager
* ModuleManager Listeners

**Trong đó:**

Module Autoloader: Đây là thành phần đặc biệt giúp cho việc xử lý các namespace, mapping, và giữ các instances đó khi cần xử lý ...

Module Manager: Thành phần này có chức năng quản lý module, quản lý và merger các cấu hình của các module, điều khiển việc loader ...

ModuleManager Listeners – Các Event listener có thể được gắn với các sự kiện khác nhau của module manager. Những listener có thể làm mọi việc từ việc giải quyết và load các module tới việc thực hiện khởi tạo các task phức tạp.

## Cấu trúc của module có dạng như sau:

module

nameModule

config

src

view

test

Controller

Form

Model

nameModule

Các thành phần cơ bản trong module

Thư mục **config** được sử dụng để quản lý cấu hình Module

Thư mục **src** chứa tất cả các mã nguồn module bao gồm tất cả các controllers and models.

Thư mục **view** được sử dụng để lưu trữ tất cả các views được sử dụng trong module

Ví dụ module users

module

Users

config

src

view

test

Controller

Form

Model

Users

## Namespaces

Bạn sẽ học cách namespace PHP được sử dụng để tránh va chạm tên, những gì lớp autoloading là làm thế nào để xác định các thông số cấu hình ứng dụng và các bước hiện tại trong vòng đời của một ứng dụng. Bạn cũng sẽ trở nên quen thuộc với các thành phần quan trọng ZF2 như Zend\EventManager, Zend\ModuleManager và Zend\ServiceManager.

Khi bạn sử dụng các class từ các thư viện khác nhau (hoặc thậm chí các class từ các thành phần khác nhau của một thư viện duy nhất) trong chương trình của bạn, tên class có thể xung đột. Điều này có nghĩa bạn có thể gặp phải hai lớp có cùng tên, dẫn đến lỗi PHP thông dịch.

Để đạt được cùng một mục tiêu, Zend Framework 2 sử dụng các tính năng PHP 5.3 ngôn ngữ được gọi là namespace. Các namespace cho phép để giải quyết va chạm tên giữa các thành phần mã, và cung cấp cho bạn khả năng để làm cho tên dài ngắn hơn.

Một không gian tên là một container cho một nhóm các tên. Bạn có thể lồng namespace vào nhau. Nếu một lớp hoặc Hàm này không xác định một không gian tên, nó sống bên trong của không gian tên toàn cục (ví dụ, PHP lớp ngoại lệ và Ngày giờ thuộc về không gian tên toàn cục).

Một ví dụ thực tế của một định nghĩa không gian tên (lấy từ thành phần Zend\Mvc) được trình bày dưới đây:

<?php

2 **namespace Zend\Mvc;**

3

4 // ...

5

6 /\*\*

7 \* Main application class for invoking applications.

8 \*/

9 **class Application**

10 {

11 // ... class members were omitted for simplicity ...

12 }

Trong Zend Framework 2, tất cả các lớp thuộc cấp Zend namespace. Dòng 2 định nghĩa không gian tên MVC, được lồng vào namespace Zend, và tất cả các lớp của thành phần này (bao gồm lớp ứng dụng trong ví dụ này trên dòng 9-12) thuộc namespace này. Bạn tách tên namespace lồng nhau với dấu gạch chéo ('\').

Trong các phần khác của mã, bạn tham khảo các lớp ứng dụng bằng cách sử dụng tên đầy đủ của nó:

<?php

$application = **new \Zend\Mvc\Application**;

Nó cũng có thể sử dụng bí danh (tên viết tắt của các lớp) với sự trình bày của câu lệnh sử dụng PHP:

<?php

// Xác định bí danh trong đầu của tập tin.

**use Zend\Mvc\Application;**

// Sau này trong mã của bạn, sử dụng tên lớp ngắn.

$application = **new Application**;

Tất cả các file PHP ứng dụng của bạn thường định nghĩa không gian tên (trừ mục file cấu hình và tập tin index.php). Ví dụ, các module chính trang web của bạn, các module ứng dụng, định nghĩa namespace riêng, tên tương đương với tên module:

<?php

namespace Application;

// ...

class Module

{

// ... class members were omitted for simplicity ...

}

## Model-View-Controller (MVC layer)

Mục tiêu cơ bản của bất kỳ MVC Framework là để cho phép dễ dàng tách biệt ba lớp của MVC, cụ thể là, model, view, and controller.

Trước khi chúng ta nhận được các chi tiết của việc tạo ra module, hãy nhanh chóng cố gắng tìm hiểu làm thế nào làm việc ba lớp trong một khuôn khổ MVC.

**Model**: Mô hình này là một đại diện của dữ liệu; mô hình này cũng là công việc logic cho các giao dịch ứng dụng khác nhau.

**View**: Chứa nội dung logic cần hiển thị trên trình duyệt cho người sử dụng.

**Controller**: Bộ điều khiển điều khiển logic ứng dụng trong bất kỳ ứng dụng MVC, tất cả hành động và sự kiện được xử lý ở lớp Controller. Các lớp Controller phục vụ như là một giao diện truyền thông giữa model và view bằng cách kiểm soát tình trạng model và thể hiện những thay đổi view. Controller cũng cung cấp điểm khởi đầu cho việc truy cập các ứng dụng.

Trong cấu trúc ZF2 MVC mới, tất cả các model, view, và controller được nhóm lại theo modules. Mỗi module sẽ có thiết lập riêng của model, view, và controller, và sẽ chia sẻ một số thành phần với các module khác.

Cấu trúc MVC trong Zend Framework 2 có dạng như sau:

module

Users

config

src

view

test

Controller

Form

Model

Users

users

users

# ZF2 module – configuration

**Nội dung:**

* Cấu hình Module.php
* URL Routing
* SimpleRouteStack
* TreeRouteStack
* Các loại Routing

## Cầu hình Modul**e**

Zend Framework 2.0 cấu hình Module được trình bày trên một loạt các tập tin có thể được tìm thấy trong bộ khung Module. Một số các tập tin cấu hình được mô tả như sau:

**Module.php**: Zend Framework 2 module quản lý sẽ tìm tập tin module.php trong thư mục gốc của module. Module quản lý này sử dụng tập tin Module.php để cấu hình Module và gọi phương thức getConfig() và getAutoloaderConfig().

<?php

**namespace** Application;

**use** Zend\Mvc\ModuleRouteListener;

**use** Zend\Mvc\MvcEvent;

**class** Module

{

**public function** getConfig()

{

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

}

**public function** getAutoloaderConfig()

{

**return array**(

'Zend\Loader\ClassMapAutoloader'=>**array**(\_\_DIR\_\_.'/autoload\_classmap.php'),

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

'namespaces'=>**array**(\_\_NAMESPACE\_\_ =>\_\_DIR\_\_.'/src/'.\_\_NAMESPACE\_\_),),);

}

}

**autoload\_classmap.php**: Phương thức **getAutoloaderConfig**() trong Module.php sẽ load tập tin autoload\_classmap.php. tập tin **autoload\_classmap.php** trả về mảng của cặp tên lớp và tên tập tin (với các tên tập tin giải quyết thông qua \_\_ DIR\_\_).

Cấu hình tự động load mặc định được cung cấp bởi các ZendSkeletonModule cần phải được vô hiệu hóa, điều này có thể được thực hiện bằng cách chỉnh sửa autoload\_classmap.php và thay thế nó với nội dung sau:

**return array**();

**config/module.config.php**: Phương thức **getConfig()** trong **Module.php** load tập tin

**config/module.config.php**. tập tin này dùng để cấu hình các tùy chọn bao gồm routes, controllers, layouts, và một số cấu hình khác.

## Routing (Định tuyến)

Định tuyến là hành động phù hợp với yêu cầu một bộ điều khiển đưa ra. Thông thường, định tuyến sẽ xem xét các yêu cầu URI, và cố gắng để phù hợp với phân khúc đường URI so với ràng buộc quy định. Nếu các ràng buộc phù hợp, một tập hợp các "matches" được trả về, một trong số đó là tên bộ điều khiển để thực hiện. Định tuyến có thể sử dụng các phần khác của URI yêu cầu. - ví dụ, các máy chủ hoặc chương trình, các tham số truy vấn, tiêu đề, phương thức yêu cầu,…

Có hai bộ định tuyến được cung cấp, là SimpleRouteStack và TreeRouteStack. Theo mặc định, Zend\MVC sử dụng TreeRouteStack như các bộ định tuyến.

### SimpleRouteStack

Router này chỉ đơn giản là cá nhân có các tuyến đường đầy đủ của họ cung cấp logic kết hợp trong một đường đi, và duyệt qua chúng theo thứ tự LIFO đến một kết hợp được tìm thấy. Như vậy, tuyến đường sẽ phù hợp hầu hết thường phải được đăng ký trước, và các tuyến đường phổ biến đầu tiên. Thêm vào đó bạn sẽ cần phải đảm bảo rằng tuyến đường có khả năng trùng nhau đã được đăng ký như vậy sao cho kết hợp cụ thể nhất sẽ phù hợp lần đầu tiên (tức là đăng ký sau).

### TreeRouteStack

Zend\MVC\ Router\Http\TreeRouteStack cung cấp khả năng đăng ký cây của các tuyến đường, và sẽ sử dụng một thuật toán btree để phù hợp với tuyến đường. Như vậy, bạn đăng ký một tuyến đường với nhiều con.

### Các loại định tuyến

#### Zend\Mvc\Router\Http\Hostname

Các tuyến đường Hostname tìm cách để phù hợp với tên máy đăng ký trong yêu cầu đối với các tiêu chí cụ thể. Thông thường, đây sẽ là một trong các hình thức sau đây:

“subdomain.domain.tld”

“:subdomain.domain.tld”

Con đường thứ hai sẽ trả về một khóa "tên miền phụ" như là một phần kết hợp của tuyến đường.

'router' => array(

'routes' => array(

'example1' => array(

'type' => 'Hostname',

'options' => array(

'route' => ':subdomain.mydomain.com',

'constraints' => array(

'subdomain' => 'mysubdomain1',

),

'defaults' => array(

'controller' => 'MyModule1\Controller\MyFirstController',

'action' => 'index',

),

),

),

'example2' => array(

'type' => 'Hostname',

'options' => array(

'route' => ':subdomain.mydomain.com',

'constraints' => array(

'subdomain' => 'mysubdomain2',

),

'defaults' => array(

'controller' => 'MyModule2\Controller\MySecondController',

'action' => 'index',

),

),

),

),

),

#### Zend\Mvc\Router\Http\Literal

Các tuyến đường Literal là để thực hiện kết hợp chính xác con đường URI. Do đó, cấu hình chỉ là con đường bạn muốn kết hợp, và "mặc định", hoặc các thông số bạn muốn trả về ở trên kết hợp.

'home' => **array**(

'type' => 'Zend\Mvc\Router\Http\Literal',

'options' => **array**(

'route' => '/',

'defaults' => **array**(

'controller' => 'Application\Controller\Index',

'action' => 'index',

),

),

),

#### Zend\Mvc\Router\Http\Method

Các tuyến đường Method được sử dụng để phù hợp với phương thức http hoặc ' verb' quy định trong yêu cầu (xem RFC 2616 Sec. 5.1.1). Nó có thể được cấu hình để phù hợp với nhiều phương thức bằng cách cung cấp một danh sách bằng dấu phẩy của phương thức tokens.

'routes' => array(

'shoppingcart' => array(

'type' => 'literal',

'options' => array(

'route' => '/shoppingcart/',

'defaults' => array(

'controller' => 'ShoppingcartController',

'action' => 'shoppingcart',

),

),

'may\_terminate' => true,

'child\_routes' => array (

'add-product' => array(

'type' => 'method',

'options' => array(

'verb' => 'get,post',

),

'child\_routes' => array(

// actual route is a child of the method

'form' => array(

'may\_terminate' => true,

'type' => 'literal',

'options' => array(

'route' => 'add/',

'defaults' => array(

'controller' => 'ShoppingcartController',

'action' => 'addProductToShoppingcart',

),

),

),

),

),

),

),

),

#### Zend\Mvc\Router\Http\Part

Một Một phần tuyến đường cho phép việc tạo một cây của tuyến có thể dựa trên các phân đoạn của con đường URI. Nó thực sự mở rộng TreeRouteStack.

$route = Part::factory(**array**(

'route' => **array**(

'type' => 'literal',

'options' => **array**(

'route' => '/',

'defaults' => **array**(

'controller' => 'Application\Controller\IndexController',

'action' => 'index'

)

),

),

'route\_plugins' => $routePlugins,

'may\_terminate' => **true**,

'child\_routes' => **array**(

'blog' => **array**(

'type' => 'literal',

'options' => **array**(

'route' => 'blog',

'defaults' => **array**(

'controller' => 'Application\Controller\BlogController',

'action' => 'index'

)

),

'may\_terminate' => **true**,

'child\_routes' => **array**(

'rss' => **array**(

'type' => 'literal',

'options' => **array**(

'route' => '/rss',

'defaults' => **array**(

'action' => 'rss'

)

),

'may\_terminate' => **true**,

'child\_routes' => **array**(

'subrss' => **array**(

'type' => 'literal',

'options' => **array**(

'route' => '/sub',

'defaults' => **array**(

'action' => 'subrss'

)

)

)

)

)

)

),

'forum' => **array**(

'type' => 'literal',

'options' => **array**(

'route' => 'forum',

'defaults' => **array**(

'controller' => 'Application\Controller\ForumController',

'action' => 'index'

)

)

)

)

));

#### Zend\Mvc\Router\Http\Regex

Một tuyến đường Regex sử dụng một biểu thức thông thường để phù hợp với con đường URI. Bất kỳ biểu thức thông thường hợp lệ được phép; đề nghị của chúng tôi là sử dụng tên có được cho bất kỳ giá trị mà bạn muốn trở về trong RouteMatch. Tuyến đường Regex thường rất phức tạp.

$route = Regex::factory(**array**(

’regex’ => ’/blog/(?<id>[a-zA-Z0-9\_-]+)(\.(?<format>(json|html|xml|rss)))?’,

’defaults’ => **array**(

’controller’ => ’Application\Controller\BlogController’,

’action’ => ’view’,

’format’ => ’html’,

),

’spec’ => ’/blog/%id%.%format%’,

));

Ở trên sẽ phù hợp với "/ blog/001-some-blog\_slug-here.html", và trả về bốn mục trong RouteMatch, một "id", các "controller", "action", và "format". Khi lắp ráp một URL từ tuyến đường này, giá trị "id" và "định dạng" sẽ được sử dụng để điền vào các đặc tả.

#### Zend\Mvc\Router\Http\Scheme

Các tuyến đường Scheme phù hợp với sơ đồ URI, và phải là một kết hợp chính xác. Như vậy, tuyến đường này, giống như các tuyến đường Literal, chỉ đơn giản là có những gì bạn muốn thích hợp và "mặc định", các thông số để trả lại cho một kết hợp.

'http' => array(

'type' => 'Zend\Mvc\Router\Http\Scheme',

'options' => array(

'scheme' => 'http',

'priority' => 2,

'defaults' => array(

'controller' => 'Application\Controller\IndexController,

'action' => 'index',

),

),

),

#### Zend\Mvc\Router\Http\Segment

Một tuyến đường Segment cho phép phù hợp với bất kỳ phân đoạn của một con đường URI. Phân đoạn được biểu hiện bằng dấu hai chấm, tiếp theo là ký tự chữ và số, nếu một phân đoạn là tùy chọn, nó phải được bao quanh bởi các dấu ngoặc. Ví dụ, "/:foo[/:bar]" sẽ kết hợp một "/" tiếp theo văn bản và gán nó vào khóa "foo"; nếu có thêm "/" các ký tự được tìm thấy, bất kỳ văn bản sau sau cùng sẽ được gán cho phím "bar".

**return array**(

'router' => **array**(

'routes' => **array**(

'users' => **array**(

'type' => 'Literal',

'options' => **array**(

'route' => '/users',

'defaults' => **array**(

'\_\_NAMESPACE\_\_' => 'Users\Controller',

'controller' => 'Users',

'action' => 'index',

),

),

'may\_terminate' => **true**,

'child\_routes' => **array**(

'default' => **array**(

'type' => 'Segment',

'options' => **array**(

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

'constraints' => **array**(

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

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

),

'defaults' => **array**(

),

),

),

),

),

),

),

'controllers' => **array**(

'invokables' => **array**(

'users\Controller\Users' => 'Users\Controller\UsersController'

),

),

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

'template\_path\_stack' => **array**(\_\_DIR\_\_ . '/../view',),

),

);

# 

# Database và Model

**Nội dung: Database và Model**

**ZendDbAdapter**

* Khai báo, Kết nối CSDL
* Tạo câu truy vấn
* Thực thị câu truy vấn
* Tạo Statement

**ZendDbSql**

* Khai báo
* Đối tượng Select
* Đối tượng Insert
* Đối tượng Update
* Đối tượng Delete

**ZendDbTableGateway**

* Giới thiệu
* Kết nối CSDL với global.php
* getServiceConfig
* Model
* ModelTable
* Controller - Model

**ZendDb RowGateway**

## ZEND\DB\ADAPTER

Đối tượng Adapter là một thành phần (sub-component) quan trọng nhất của Zend\Db. Nó chịu trách nhiệm thích ứng các code được viết trong Zend\Db tới các phần mở rộng PHP đích và cơ sở dữ liệu từ nhà cung cấp. Để thực hiện việc này, nó tạo ra một lớp trừu tượng cho các thành phần mở rộng PHP, được gọi là "Driver" của Zend\Db adapter. Nó cũng tạo ra một lớp trừu tượng, được gọi là "Platform" của adapter, cho các nguồn khác nhau mà mỗi nhà cung cấp platform cụ thể có thể có trong việc thực hiện SQL / RDBMS.

### Tạo Adapter

Tạo một adapter đơn giản bằng cách sử dụng lớp Zend\Db\Adapter\Adapter. Trường hợp phổ biến nhất là thông qua một mảng cấu hình cho Adapter.

$adapter = new Zend\Db\Adapter\Adapter($configArray);

Mảng này là một trừu tượng với các tham số yêu cầu mở rộng.

Bảng dưới đây là các giá trị nên có trong mảng cấu hình:

|  |  |  |
| --- | --- | --- |
| Từ khóa | Yêu cầu/ không yêu cầu? | Giá trị |
| driver | Yêu cầu | Mysqli, Sqlsrv, Pdo\_Sqlite, Pdo\_Mysql, Pdo=OtherPdoDriver |
| database | Thường yêu cầu | Tên của CSDL |
| username | Thường yêu cầu | Tên đăng nhập khi kết nối |
| password | Thường yêu cầu | Mật khẩu |
| hostname | Không thường yêu cầu | Địa chỉ IP hoặc hostname kết nối tới |
| port | Không thường yêu cầu | port |
| charset | Không thường yêu cầu | Bộ ký tự sử dụng |

Ví dụ: Kết nối tới MySQL sử dụng ext/mysqli

$adapter = new Zend\Db\Adapter\Adapter(array(

'driver' => 'Mysqli',

'database' => 'zend\_db\_example',

'username' => 'developer',

'password' => 'developer-password'

));

Một ví dụ khác, một kết nối SQLite thông qua PDO

$adapter = **new** Zend\Db\Adapter\Adapter(**array**(

’driver’ => ’Pdo\_Sqlite’,

’database’ => ’path/to/sqlite.db’

));

**Danh sách các Driver được hỗ trợ:**

Mysqli: ext/mysqli driver

Pgsql: ext/pgsql driver

Sqlsrv: ext/sqlsrv driver (from Microsoft)

Pdo\_Mysql: MySQL thông qua mở rộng PDO

Pdo\_Sqlite: SQLite thông qua mở rộng PDO

Pdo\_Pgsql: PostgreSQL thông qua mở rộng PDO

**Chú ý:**

Sử dụng cách này để tạo adapter, lớp Adapter sẽ cố gắng để tạo ra bất kỳ phụ thuộc nào dù không được nêu rõ ràng.

Một đối tượng Driver sẽ được tạo ra từ mảng cấu hình được cung cấp trong constructor. Một đối tượng Platform sẽ được tạo ra dựa vào kiểu của lớp instantiated.

Một đối tượng mặc định ResultSet được tạo ra và sử dụng. Các đối tượng khác cũng có thể được thêm vào.

### Chuẩn bị câu lệnh truy vấn thông qua Zend\Db\Adapter\Adapter::query()

Mặc định, query() là thứ mà ta chuẩn bị sẵn và xem như một phương tiện để xử lý các câu lệnh SQL. Ta sẽ cung cấp một câu lệnh SQL với các giá trị được thay thế bằng các ký hiệu đại diện tham số.

Ví dụ về quy trình làm việc với Zend\Db\Adapter\Adapter này là:

$adapter->query('SELECT \* FROM `artist` WHERE `id` = ?', array(5));

Ví dụ này được thực hiện thông qua các bước sau:

* Tạo ra một đối tượng Statement mới
* Chuẩn bị một mảng để truyền vào ParameterContainer nếu cần
* Đưa ParameterContainer vào đối tượng Statement
* Thực thi đối tượng Statement, tạo ra một đối tượng kết quả Result

Kiểm tra đối tượng Result để kiểm tra xem sql được cung cấp có phải là một “query” (lấy dữ liệu), hay là một bộ các statement được tạo ra (thêm, cập nhật hoăc xóa dữ liệu).

Nếu kết quả là một bộ câu lệnh truy vấn được tạo ra thì sao chép ResultSet prototype, đưa Result vào như nguồn dữ liệu, trả về (trường hợp thêm, cập nhật hoặc xóa dữ liệu)

Ngược lại, trả về Result (trường hợp lấy dữ liệu để hiển thị)

### Thực thi câu lệnh truy vấn thông qua Zend\Db\Adapter\Adapter::query()

Trong một vài trường hợp ta cần phải thực thi ngay các statement. Mục đích chính của việc thực thi ngay câu lệnh truy vấn thay vì phải chuẩn bị rồi mới thực thi là vì có thể ta đang cố gắng thực thi một DDL statement (trong hầu hết các extension và các vendor platform, mà không cần chuẩn bị.

$result=$adapter->query("delete from user where id=?", **array(27), $adapter::*QUERY\_MODE\_EXECUTE);***

### Tạo Statement

Trong khi query() rất hữu ích khi thực hiện truy vấn nhanh một lần tới CSDL thông qua Adapter, nó thường có ý nghĩa hơn khi tạo ra một statement và tương tác trực tiếp để ta có thể kiểm soát tốt hơn với các công việc cần chuẩn bị và thực hiện theo luồng.

Để thực hiện công việc này, Adapter cung cấp một phương thức gọi là **createStatement()** cho phép ta tạo ra một Driver Statement cụ thể để sử dụng giúp ta có thể quản lý công việc cần chuẩn bị và thực hiện theo luồng (prepare-then-execute workflow)

$statement = $adapter->createStatement($sql, $optionalParameters);

$result = $statement->execute();

## ZEND\DB\SQL

Zend\Db\Sql là một lớp SQL trừu tượng dùng để xây dựng các truy vấn SQL nền tảng, cụ thể thông qua object-oriented API.

Kết quả cuối cùng của đối tượng Zend\Db\Sql sẽ vừa tạo ra một Statement và Parameter container đại diện cho truy vấn đích, hoặc là một chuỗi đầy đủ có thể thực thi trực tiếp trên nền tảng của CSDL.

Để thực hiện việc này, các đối tượng Zend\Db\Sql cung cấp một đối tượng Zend\Db\Adapter\Adapter *(đã giới thiệu)* để tạo ra những kết quả mong muốn.

### Zend\Db\Sql\Sql

Có một số công việc chính cần phải tương tác với CSDL (từ DML - Data Manipulation Language): truy vấn lọc dữ liệu, thêm dữ liệu, cập nhật và xóa dữ liệu.

Như vậy có bốn đối tượng chính mà developer cần tương tác hoặc xây dựng các câu lệnh truy vấn: Zend\Db\Sql\Select, Insert, Update và Delete.

Tuy nhiên bốn công việc này có liên quan khá mật thiết với nhau, và thường kết hợp cùng nhau trong cùng một ứng dụng, các đối tượng Zend\Db\Sql\Sql giúp ta tạo ra chúng và tạo ra kết quả mà ta mong muốn có được.

use Zend\Db\Sql\Sql;

$sql = new Sql($adapter);

$select = $sql->select(); // @return Zend\Db\Sql\Select

$insert = $sql->insert(); // @return Zend\Db\Sql\Insert

$update = $sql->update(); // @return Zend\Db\Sql\Update

$delete = $sql->delete(); // @return Zend\Db\Sql\Delete

Bây giờ developer có thể tương tác với những đối tượng này.

### Sử dụng đối tượng Select

use Zend\Db\Sql\Sql;

$sql = new Sql($adapter);

$select = $sql->select(); // $select = $sql->select(‘table’);

**Các phương thức:**

#### Phương thức Columns()

Chỉ định số field cần liệt kê

Cú pháp:

$select->columns(array('field1,field2,field3,...'));

Ví dụ:

$select->columns(array('customers\_name', 'address', 'phone\_number', 'delivery\_address', 'email'));

#### Phươngthức table()

Chỉ định bảng cần thực hiện

Cú pháp:

$select->from(‘table’));

**Lưu ý:**

*Nếu ở phương thức select chúng ta chỉ định table rồi, thì phương thức này không thể chỉ định lại table*

#### Phương thức limit()

Chỉ định số mẩu tin cần liệt kê

Cú pháp:

$select->limit($limit);

Ví dụ:

$select->limit(3);

#### Phương thức offset()

Chỉ định vị trí mẩu tin bắt đầu lấy

Cú pháp:

$select->offset($offset);

Ví dụ:

$select->offset(3);

#### Phương thức where()

Thiết lập biểu thức điều kiện

Cú pháp:

$select->where(array());

Ví dụ:

$select->where(array(‘id’=>2));

#### Phương thức order()

Sắp xếp dữ liệu

Cú pháp:

$select->order(string|array);

Ví dụ:

$select->order(‘id desc, age DESC');

$select->order(array('name ASC', 'age DESC'));

#### Phương thức join()

Kết hợp đến bảng cần tham chiếu field

Cú pháp:

$select->join('table2', 'table.field=table2.field', array('field1,field2,...'),

$select:: *JOIN\_INNER*);// *JOIN\_LEFT/ JOIN\_RIGHT*

Ví dụ:

$select->join('users‘, ‘users.id=baiviet.ID‘, array('fullname'),$select::*JOIN\_INNER);*

#### Phương thức group()

Kết nhóm dữ liệu

Cú pháp:

$select->group(array('field,...'))

Ví dụ:

$select ->group(array('bill\_id'));

**Phương thức having()**

Biểu thức điều kiện field thống kê dữ liệu

Cú pháp:

$select->having(array('biểu\_thức\_điều\_kiện'));

Ví dụ:

$select->columns(array('bill\_id','tssp'=>new Expression('count(\'product\_id\')')));

$select ->group(array('bill\_id'));

$select ->having(array('tssp>2'));

#### Thực thi câu truy vấn

**Có 2 cách**

**Cách 1:**

$selectString = $sql->getSqlStringForSqlObject($select);

$results=$adapter->query($selectString, $adapter::QUERY\_MODE\_EXECUTE);

**Cách 2:**

$state=$sql->prepareStatementForSqlObject($select);

$results=$state->execute();

#### Sử dụng đối tượng Insert

$adapter =new Adapter($this->connect);

$sql = new Sql($adapter);

$insert=$sql->insert();//$insert=$sql->insert("table");

$insert->into($table)

$insert->values($values);

Ví dụ:

$adapter=new Adapter($this->adapter);

$sql=new Sql($adapter);

$insert=$sql->insert('loai\_mon\_an');

$insert->columns(array('ten\_loai','mo\_ta','hinh'));

$insert->values( array($ten\_loai,$mo\_ta,$hinh),$insert::VALUES\_MERGE);

$state=$sql->prepareStatementForSqlObject($insert);

$result=$state->execute();

return $result->getAffectedRows();

#### Sử dụng đối tượng Update

$adapter = new Adapter($this->connect);

$sql=new Sql($adapter);

$update=$sql->update();

$update->table($table)

$update->set($values);

$update->where(string|array);

Ví dụ:

$adapter=new Adapter($this->connect);

$sql=new Sql($adapter);

$update=$sql->update('loai\_mon\_an');

$data=array(

'ten\_loai'=>$ten\_loai,

'mo\_ta'=>$mo\_ta,

'hinh'=>$hinh

);

$update->set($data);

$update->where(array('ma\_loai'=>$ma\_loai));

$state=$sql->prepareStatementForSqlObject($update);

$result=$state->execute();

return $result->getAffectedRows()**;**

#### Sử dụng đối tượng Delete

$adapter = **new Adapter($this->connect);**

$sql=**new Sql($adapter);**

$delete=$sql->delete();

$delete->from($table);

$delete->where(string|**array**)

**Ví dụ:**

$adapter=new Adapter($this->connect);

$sql=new Sql($adapter);

$delete=$sql->delete('loai\_mon\_an');

$delete->where(array('ma\_loai'=>$ma\_loai));

$state=$sql->prepareStatementForSqlObject($delete);

$result=$state->execute();

return $result->getAffectedRows();

## ZEND\DB\TABLEGATEWAY

Đối tượng TableGateway được thiết kế để cung cấp một đối tượng đại diện cho một bảng trong một cơ sở dữ liệu, và các phương thức của đối tượng này phản ánh các hoạt động phổ biến nhất trên một bảng cơ sở dữ liệu.

Vi dụ:

Chúng ta có table Album

album (

id int(11) **NOT NULL** auto\_increment,

artist varchar(100) **NOT NULL**,

title varchar(100) **NOT NULL**,

**PRIMARY KEY** (id)

);

Để thực hiện các chức năng như select, insert, update, delete trên table này ta thực hiện tuần tự như sau:

**Tạo Model files**

Zend Framework không cung cấp một thành phần Zend\Model bởi vì mô hình này là công việc logic của bạn và nó thuộc vào bạn để quyết định cách bạn muốn nó làm việc. Có rất nhiều thành phần mà bạn có thể sử dụng cho điều này tùy thuộc vào nhu cầu của bạn. Một cách tiếp cận là phải có class model đại diện cho mỗi thực thể trong ứng dụng của bạn và sau đó sử dụng các đối tượng mapper tải và lưu các đơn vị cơ sở dữ liệu.

Đối với hướng dẫn này, chúng ta sẽ tạo ra một model rất đơn giản bằng cách tạo ra một lớp AlbumTable có sử dụng lớp Zend\DB\TableGateway\TableGateway, trong đó mỗi đối tượng album là một đối tượng Album (được gọi là một tổ chức). Đây là một thực hiện của mẫu thiết kế bảng dữ liệu Gateway để cho phép giao tiếp với dữ liệu trong một bảng cơ sở dữ liệu.

Hãy bắt đầu bằng cách tạo ra một tập tin gọi là Album.php dưới Module/Album/src/Album/Model:

**namespace** Album\Model;

**class Album**

{

**public** $id;

**public** $artist;

**public** $title;

**public function** exchangeArray($data)

{

$this->id = (!**empty**($data[’id’])) ? $data[’id’] : **null**;

$this->artist = (!**empty**($data[’artist’])) ? $data[’artist’] : **null**;

$this->title = (!**empty**($data[’title’])) ? $data[’title’] : **null**;

}

}

Đối tượng thực thể Album của chúng ta là một lớp PHP đơn giản. Để làm việc với Zend\Db lớp TableGateway, chúng ta cần phải thực hiện phương thức exchangeArray(). Phương pháp này chỉ đơn giản là sao chép dữ liệu từ truyền qua trong mảng đến các thuộc tính thực thể.

Tiếp theo, chúng ta tạo ra tập tin AlbumTable.php của chúng ta trong thư mục Module/Album/src/Album/Model như thế này:

**namespace** Album\Model;

**use** Zend\Db\TableGateway\TableGateway;

**class AlbumTable**

{

**protected** $tableGateway;

**public function** \_\_construct(TableGateway $tableGateway)

{

$this->tableGateway = $tableGateway;

}

**public function** fetchAll()

{

$resultSet = $this->tableGateway->select();

**return** $resultSet;

}

**public function** getAlbum($id)

{

$id = (int) $id;

$rowset = $this->tableGateway->select(**array**(’id’ => $id));

$row = $rowset->current();

**if** (!$row) {

**throw new** \Exception("Could not find row $id");

}

**return** $row;

}

**public function** saveAlbum(Album $album)

{

$data = **array**(

’artist’ => $album->artist,

’title’ => $album->title,

);

$id = (int)$album->id;

**if** ($id == 0) {

$this->tableGateway->insert($data);

} **else** {

**if** ($this->getAlbum($id)) {

$this->tableGateway->update($data, **array**(’id’ => $id));

} **else** {

**throw new** \Exception(’Album id does not exist’);

}

}

}

**public function** deleteAlbum($id)

{

$this->tableGateway->delete(**array**(’id’ => $id));

}

}

### Cấu hình Adapter

Chúng ta cần phải cấu hình ServiceManager để nó biết làm thế nào để có được một Zend\Db\Adapter\Adapter. Việc này được làm bằng cách sử dụng một factory được gọi là Zend\Db\Adapter\AdapterServiceFactory mà chúng ta có thể cấu hình trong hệ thống cấu hình kết hợp. ModuleManager của Zend Framework 2 kết hợp tất cả các cấu hình từ mỗi tập tin module.config.php của module và sau đó kết hợp các tập tin trong thư mục config/autoload (\*.global.php and then \*.local.php files).

Chúng ta thêm thông tin cấu hình cơ sở dữ liệu của chúng ta trong tập tin global.php.

**return array**(

’db’ => **array**(

’driver’ => ’Pdo’,

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

‘driver\_options’ => **array**(PDO::MYSQL\_ATTR\_INIT\_COMMAND => ’SET NAMES \’UTF8\’’),),

‘service\_manager’ => **array**(

’factories’ => **array**(

’Zend\Db\Adapter\Adapter’=> ’Zend\Db\Adapter\AdapterServiceFactory’,),

),

);

Bạn nên đặt ủy nhiệm cơ sở dữ liệu của bạn trong config/autoload/local.php:

**return array**(

’db’ => **array**(

’username’ => ’YOUR USERNAME HERE’,

’password’ => ’YOUR PASSWORD HERE’,

),

);

### Sử dụng ServiceManager để cấu hình table gateway

Để luôn luôn sử dụng cùng một ví dụ của AlbumTable của chúng ta. Điều này rất dễ thực hiện nhất trong lớp module mà chúng ta tạo ra một phương thức gọi là getServiceConfig() được gọi tự động bởi ModuleManager và áp dụng cho các ServiceManager. Sau đó chúng tôi sẽ có thể lấy nó trong bộ điều khiển của chúng tôi khi chúng tôi cần.

Để cấu hình ServiceManager, chúng ta có thể cung cấp tên của lớp được khởi tạo hoặc factory khởi tạo các đối tượng khi ServiceManager cần nó. Chúng tôi bắt đầu bằng cách thực hiện getServiceConfig() để cung cấp một factory tạo ra một AlbumTable. Thêm phương thức này dưới cùng của tập tin Module.php trong module/Album.

**namespace** Album;

// Add these import statements:

**use** Album\Model\Album;

**use** Album\Model\AlbumTable;

**use** Zend\Db\ResultSet\ResultSet;

**use** Zend\Db\TableGateway\TableGateway;

**class Module**

{

// getAutoloaderConfig() and getConfig() methods here

// Add this method:

**public function** getServiceConfig()

{

**return array**(

’factories’ => **array**(

’Album\Model\AlbumTable’ => **function**($sm) {

$tableGateway = $sm->get(’AlbumTableGateway’);

$table = **new** AlbumTable($tableGateway);

**return** $table;

},

’AlbumTableGateway’ => **function** ($sm) {

$dbAdapter = $sm->get(’Zend\Db\Adapter\Adapter’);

$resultSetPrototype = **new** ResultSet();

$resultSetPrototype->setArrayObjectPrototype(**new** Album());

**return new** TableGateway(’album’, $dbAdapter, **null**, $resultSetPrototype);

},

),

);

}

}

### Thực hiện Controller

Bây giờ ServiceManager có thể tạo ra một thể hiện AlbumTable cho chúng ta, chúng ta có thể thêm một phương thức để điều khiển để lấy nó. Thêm getAlbumTable() vào lớp AlbumController:

**protected** $albumTable;

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

**public function** getAlbumTable()

{

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

$sm = $this->getServiceLocator();

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

}

**return** $this->albumTable;

}

Bây giờ chúng ta có thể gọi getAlbumTable () từ bên trong controller của chúng ta bất cứ khi nào chúng ta cần phải tương tác với model của chúng ta.

Ví dụ:

$album= $this->getAlbumTable()->fetchAll();

Một số phương thức trong TableGateway

|  |  |
| --- | --- |
| getTable() |  |
| getAdapter() |  |
| getColumns() |  |
| getFeatureSet() |  |
| getResultSetPrototype() |  |
| getSql() |  |
| select($where = null) |  |
| selectWith(Select $select) |  |
| insert($set) |  |
| insertWith(Insert $insert) |  |
| update($set, $where = null) |  |
| updateWith(Update $update) |  |
| delete($where) |  |
| deleteWith(Delete $delete) |  |
| getLastInsertValue() |  |

## ZEND\DB\ROWGATEWAY

Zend\DB\RowGateway là một phần của Zend\Db, thực hiện mô hình RowGateway từ PoEAA. Đối tượng Row gateway chủ yếu thực hiện một dòng trong cơ sở dữ liệu, cung cấp hai phương thức save() và delete() để thực hiện ngược lại một dòng dữ liệu được lấy lên.

**interface** RowGatewayInterface

{

**public function** save();

**public function** delete();

}

Khi hầu hết thời gian, RowGateway sẽ được dùng kết hợp với Zend\DB\ResultSet tạo các đối tượng, nó có thể sử dụng độc lập. Để sử dụng độc lập, bạn chỉ cần một Adapter và một tập hợp các dữ liệu để làm việc. Các trường hợp sử dụng dưới đây trình bày Zend\DB\RowGateway\RowGateway sử dụng dạng đơn giản nhất:

**use** Zend\Db\RowGateway\RowGateway;

// query the database

$resultSet = $adapter->query(’SELECT \* FROM ‘user‘ WHERE ‘id‘ = ?’, **array**(2));

// get array of data

$rowData = $resultSet->current()->getArrayCopy();

// row gateway

$rowGateway = **new** RowGateway(’id’, ’my\_table’, $adapter);

$rowGateway->populate($rowData);

$rowGateway->first\_name = ’New Name’;

$rowGateway->save();

// or delete this row:

$rowGateway->delete();

Công việc mô tả ở trên là rất đơn giản khi RowGateway được sử dụng kết hợp với các tính năng TableGateway. những gì đạt được là đối tượng Table gateway khi lựa chọn từ một bảng, sẽ tạo ra ResultSet đó là khi có khả năng tạo các đối tượng Row Gateway có hiệu lực.

Sử dụng của nó trông như thế này:

**use** Zend\Db\TableGateway\Feature\RowGatewayFeature;

**use** Zend\Db\TableGateway\TableGateway;

$table = **new** TableGateway(’artist’, $adapter, **new** RowGatewayFeature(’id’));

$results = $table->select(**array**(’id’ => 2));

$artistRow = $results->current();

$artistRow->name = ’New Name’;

$artistRow->save();

# FORM và ACTION

**Nội dung:**

* Giới thiệu Zend/Form
* Các thành phần chính của Zend/Form
* Sử dụng Zend/Form

## Giới thiệu Zend/Form

Zend/Form được thiết kế chủ yếu như một cầu nối giữa các model và View Layer. Nó tạo ra một layer chứa các đối tượng đại diện cho các element, Input Filter, và một số phương thức ràng buộc dữ liệu trên form và các đối tượng kèm theo.

Zend\Form component bao gồm các đối tượng:

* Elements, bao gồm tên và các thuộc tính
* Fieldsets, mở rộng từ Elements, nhưng cho phép kết hợp các fieldset và element khác.
* Forms, mở rộng từ các Fieldsets (và cả Elements). Chúng cung cấp dữ liệu và object binding, và kết hợp với InputFilters. Data binding được thực hiện thông qua *ZendStdlibHydrator*.

Để tạo điều kiện thuận lợi cho việc sử dụng view layer, Zend\Form cũng tập hợp các view helper cụ thể dành cho form.

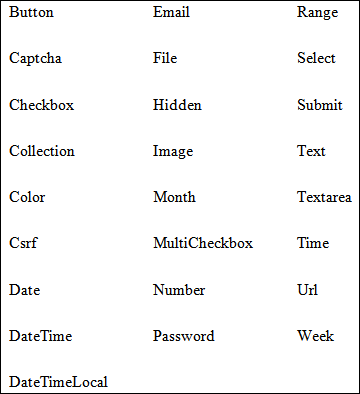
Nó chấp nhận các element, fieldset, form và sử dụng các thuộc tính nó kết hợp để đánh dấu (render markup)

Giải pháp của ZF2 dành cho Form là kết hợp nhiều thành phần làm việc với nhau:

* Zend\Form
* Zend\Input Filter
* View helpers
* Zend\Stdlib\Hydrator

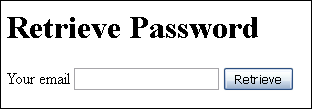
Zend/Form là một cầu nối giữa model entity và view, gồm có:

* Element và fieldset
* Input filter
* Data transfer methods (các phương pháp chuyển dữ liệu)
* Elements



* Zend/ InputFilter
* Cung cấp
* Lọc dữ liệu nhập vào trên form
* Xác nhận dữ liệu nhập vào trên form
* Thực hiện
* Thêm các đối tượng Input vào Input Filter
* Hoặc Factory cho một mảng cụ thể
* View helper
* Thực hiện (render) form sử dụng một loạt các view helper sau:
* Form(): để mở và đóng thẻ form
* formXxx() cho mỗi element. Ví dụ formHidden()
* formLable cho mỗi nhãn của element
* Ngoài ra, formRow() để thực hiện (render) lable, element, và thông điệp báo lỗi cho một đối tượng
* Developer phải tự thiết kế & trang trí form
* Data transfer (Hydrator)
* Ràng buộc một đối tượng với form, sau đó sẽ:
* Ánh xạ các giá trị đã được xác nhận vào các đối tượng sau khi post
* Trích xuất giá trị từ đối tượng tới form chứa nó
* Các chiến lược khác:
* ArraySerializable
* ObjectProperty
* ClassMethods

## Xem xét form sau:



<h1> Retrieve Password</h1>

<form action="/" method="post">

<label for="email">Your email</label>

<input type="text" name="email" id="email" value="" />

<input name="submit" type="submit" value="Retrieve" />

</form>

* Để sử dụng Zend/Form
* Tạo một form
* Khai báo form trong Controller
* Tạo View Script
* Tự động hóa View Helper
* Tự động hóa View Script
* Validation: InputFilter
* Quản lý lỗi
* Quản lý thông tin nhập
* Data Transfer: Binding (Ràng buộc dữ liệu nhập)

#### Tạo một form

* Trước tiên cần phải tạo một form

Ví dụ: Khai báo form “Retrieve Password”

// đặt tại: module\Application\src\Application\Form\RetrievePassword.php

namespace Application\Form;

use Zend\Form\Form;

class RetrievePassword extends Form

{

public function \_\_construct($name = 'null')

{

// we want to ignore the name passed

parent::\_\_construct(’retrievepassword’);

$this->setAttribute(’method’, ’post’);

**$this->add(array**(

’name’ => ’email’,

‘type’ => ‘Zend\Form\Element\Text’,

‘option’ => array(‘label’ => ‘Your email’)

’attributes’ => array(‘id’ => ‘email’,),));

**$this->add(array**(

’name’ => ’submit’,

‘type’ => ‘Zend\Form\Element\Submit’,

’attributes’ => array(‘value’ => ’Retrieve’,

‘class’ => ‘btn-success’),));

}

#### Khai báo form trong Controller

* Khi đã tạo form, cần khai báo form trong Controller

// đặt tại: module\Application\src\Application\Controller\IndexCotroller.php

namespace Application\Controller;

use Zend\Mvc\Controller\AbstractActionController;

use Application\Form\RetrievePassword as RetrievePasswordForm;

class IndexController extends AbstractActionController

{

public function retrievePasswordAction()

{

$form = new RetrievePasswordForm();

return array(‘form’ => $form);

}

}

#### Tạo View Script

* Sau đó tạo một View Script để hiển thị form này

// đặt tại module\Application\view\application\index\retrieve\_password.phtml

<?php

$form = $this->form;

$form->setAttribute('action', $this->url('retrieve\_password));

$form->prepare();

echo $this->form()->openTag($form). PHP\_EOL;

echo $this->formLabel($form->get('email')) . PHP\_EOL;

echo $this->formInput($form->get(‘email’)) . PHP\_EOL;

echo $this->FormElementErrors($form->get(‘email’)) . PHP\_EOL;

echo $this->formSubmit($form->get('submit')) . PHP\_EOL;

echo $this->form()->closeTag($form). PHP\_EOL;

?>

* Cũng có thể sử dụng formRow view helper giúp đơn giản hóa việc thực hiện các element.

Ví dụ:

echo $this->formLabel($form->get('email')) . PHP\_EOL;

echo $this->formInput($form->get(‘email’)) . PHP\_EOL;

echo $this->FormElementErrors($form->get(‘email’)) . PHP\_EOL;

Chuyển thành:

echo $this->formRow($form->get('email')) . PHP\_EOL;

## Validation: InputFilter

* Đưa một InputFilter vào form

namespace Application\Form

use Zend\InputFilter\ InputFilter;

class RetrievePasswordFilter extends InputFilter

{

public function \_\_construct()

{

// input object define here (\*)

}

}

Ví dụ: Đưa một InputFilter có tên là email vào form

namespace Application\Form

use Zend\InputFilter\ InputFilter;

class RetrievePasswordFilter extends InputFilter

{

public function \_\_construct(){

$this->add(array(

'name' => 'email',

'required' => true,

'filters' => array(

array('name' => 'StringTrim'),),

‘validator’ => array(

array(‘name’ => ‘EmailAddress’),

)

));

}

}

* Đưa InputFilter vào và thực hiện yêu cầu POST (trong Controller)

public function retrievePasswordAction(){

$form = new RetrievePasswordForm();

$form->setInputFilter(new RetrievePasswordFilter());

$request = $this->getRequest();

if($request->isPost()){

$form->setData($request->getPost());

if($form->isValid())

{

// do something here

}

}

return array(‘form’ => $form);

}

## Quản lý lỗi

Thông thường khi có lỗi nhập liệu trên form sẽ có các thông điệp thông báo nhưng là thông báo có sẵn, chúng ta cũng có thể thay đổi những câu thông báo này (error message)

// thêm vào (\*) trong function \_\_construct() của class RetrievePasswordFilter đã xây dựng ở trên

$this->add(array(

'name' => 'email',

'required' => true,

'filters' => array(

array('name' => 'Strip Tags'),

array('name' => 'StringTrim'),

),

‘validators’ => array(

array(

‘name’ => ‘EmailAddress’,

‘options’ => array(

‘message’ => array(

‘EmailAddressInvalidFormat’ => ‘Please enter a valid email address)

),

),

),

));

### Quản lý thông tin nhập

Thông thường khi người dùng không nhập dữ liệu vào một số field bắt buộc nhập liệu trên form, form sẽ có các thông điệp thông báo nhưng là thông báo có sẵn, chúng ta cũng có thể thay đổi những câu thông báo này (‘required’ message)

// thêm vào (\*) trong function \_\_construct() của class RetrievePasswordFilter

$this->add(array(…

),

‘validators’ => array(

array( ‘name’ => ‘NotEmpty’,

‘break\_chain\_on\_failure’ => true,

‘options’ => array(

‘message’ => array(

‘isEmpty’ => ‘Email address is require’),)),

array(

‘name’ => ‘EmailAddress’,

‘options’ => array(

‘message’ => array(

‘EmailAddressInvalidFormat’ => ‘Please enter a valid email address)

)

),

),

);

# File Transfer và Paginator

**Nội dung:**

File Transfer và Paginator

* File Transfer
* Paginator

## File Transfer

### Đặt vấn đề:

Thông thường các trang web đều yêu cầu các sản phẩm được giới thiệu phải có hình ảnh kèm theo để khách hàng biết được sản phẩm.

### Để làm được việc này thì:

Ta phải có thư mục lưu trữ hình ảnh chung

Mở rộng ứng dụng bằng cách trong form dành cho người dùng nhập liệu, người dùng có thể trực tiếp chọn file hình ảnh.

Ứng dụng sẽ upload hình ảnh này vào thư mục hình ảnh, còn tên của hình ảnh này thì được lưu vào CSDL cùng với các thông tin khác của sản phẩm.

Ở trang hiển thị sản phẩm sẽ hiển thị hình ảnh đã được lưu trữ tên trong CSDL và hình trong thư mục lưu trữ hình ảnh chung

### Thực hiện:

Tạo thư mục lưu trữ hình ảnh chung

Ví dụ: Tạo thư mục **images** trong thư mục **public** của ứng dụng

Tạo element kiểu file để người dùng có thể chọn hình ảnh

Ví dụ: Trong ProductForm.php, bổ sung một element có kiểu là file với tên là **image** với mục đích để người dùng chọn hình ảnh trên máy tính.

<?php

// module/Product/src/Product/Form/ProductForm.php:

namespace Product\Form;

use Zend\Form\Form;

use Zend\Form\Element;

class ProductForm extends Form

{

public function \_\_construct($name = null)

{

…

$this->add(array(

'name' => 'image',

'attributes' => array(

'type' => 'file',

),

'options' => array(

'label' => 'Chọn hình ',

),

));

…

Thêm Input Filter cho element file **image:** Sau khi đã có element file image, ta cần phải tạo một Input Filter cho element mới này.

Ví dụ:

public function getInputFilter()

{

…

$inputFilter->add($factory->createInput(array(

'name' => 'image',

'required' => false,

'validators' => array(

array('name' => 'fileupload'),

array('name' => 'filesize', 'options' => array(

'min' => 4000, 'max' => 5000,

)),

),

)

));

…

}

Chuyển file người dùng chọn vào thư mục lưu trữ hình ảnh chung: Cần sử dụng lớp Transfer của Zend Framework để thực hiện công việc chuyển file.

Ví dụ: Cần phải thêm vào phương thức thêm sản phẩm addAction() công việc chuyển file này

class ProductController extends AbstractActionController

{…

public function addAction()

{ ...

$form->setData($request->getPost());

$upload = new Transfer();

$destination = './public/images/';

$upload->setDestination($destination);

try { $upload->receive();

$filename = substr($upload->getFileName(),strlen($destination));

} catch (Zend\_File\_Transfer\_Exception $e) {

echo $e->message();

}

if ($form->isValid()) {

$data = $form->getData();

$data['image'] = $filename;

…

}

}

return array('form' => $form);

}

…

}

## Paginator

### Giới thiệu Zend\Paginator

Zend\Paginator là một component rất linh hoạt giúp cho việc trình bày dữ liệu theo kiểu phân trang. Giúp người dùng dễ dàng xem các nội dung trên một trang.

Mục tiêu chính khi thiết kế Zend/Paginator là:

Phân trang dữ liệu tùy ý, không chỉ là cơ sở dữ liệu quan hệ.

Chỉ lấy phần kết quả cần hiển thị

Không buộc người dùng phải tuân theo một cách hiển thị dữ liệu duy nhất

Độc lập khi sử dụng

Phân trang

### Phân trang bộ dữ liệu

Để chia dữ liệu vào các trang, Zend \ Paginator phải có một cách chung để truy cập dữ liệu. vì lý do đó, tất cả truy cập dữ liệu diễn ra thông qua các adapter (bộ điều hợp) nguồn dữ liệu.

Dưới đây là một vài Adapter mặt định của Zend Framework:

|  |  |
| --- | --- |
| **Adapter** | **Mô tả** |
| Array | Chấp nhận một mảng PHP |
| DbSelect | Chấp nhật một Zend\Db\Sql\Select cộng với một Zend\Db\Adapter\Adapter hoặc Zend\Db\Sql\Sql để phân trang các dòng lấy từ CSDL |
| Iterator | Chập nhận một Iterator instance |
| Null | Không sử dụng Zend\Paginator để quản lý dữ liệu phân trang. Ta vẫn có thể tận dụng lợi thế của tính năng kiểm soát việc phân trang. |

Để tạo một instance của Zend\Paginator, cần phải cung cấp một adapter

$paginator = new \Zend\Paginator\Paginator(

new \Zend\Paginator\Adapter\Array($array));

// $array: là mảng các mục lấy được từ CSDL

Thiết lập trang hiện hành

$page = (int) $this->params()->fromRoute('id', 0);

$paginator->setCurrentPageNumber($page);

Trang hiện hành này được lấy từ URL xuống – dựa trên định tuyến đã được khai báo trong file module.config.php của module

return array(

'routes' => array(

'paginator' => array(

'type' => 'segment',

'options' => array(

'route' => '/paginator[/:action][/:id][/title/:title][/]',

'defaults' => array(

'id' => 1,

),),),),

);

Ta cũng có thể viết như sau

$paginator->setCurrentPageNumber(

$this->params()->fromRoute('id',0)

);

Sau đó, ta đặt số mục muốn hiển thị trên một trang

$paginator->setItemCountPerPage(4);

Rồi gọi hiển thị view là xong

return new ViewModel(array(

'paginator' => $paginator,

));

**Gọi sử dụng phân trang ở trang view**

<?php

echo $this->paginationControl(

$this->paginator,'Sliding','my\_pagination\_control',

array('route' => 'application/paginator')

);

?>

Phương thức paginationControl có 4 tham số truyền vào: paginator instance, kiểu scrolling, tên của file view script, và các tham số bổ sung khác (tham số thứ tư là tham số tùy chọn).

Tham số thứ hai và thứ ba rất quan trọng. Trong đó tham số thứ ba là tên file của view script, file này chứa nội dung sẽ được hiển thị ở phần điều khiển phân trang.

Còn tham số thứ hai là loại scrolling, quy định kiểu hiển thị mà “Sliding” là một trong 4 loại scrolling hiển thị kiểu phân trang mà Zend cung cấp:

|  |  |
| --- | --- |
| **Kiểu Scrolling** | **Mô tả** |
| All | Trả về danh sách tất cả các trang. Tiện lợi khi sử dụng menu dropdown điểu khiển phân trang với số lượng trang không nhiều. Trong trường hợp này, ta muốn hiển thị tất cả các trang cho người dùng thấy một lúc. |
| Elastic | Kiểu scrolling giống như Google-like có thể mở rộng ra danh sách các trang khi người dùng cuộn qua các trang. |
| Jumping | Khi người dùng cuộn qua, số trang tăng lên đến cuối của một dãy, sau đó bắt đầu lại ở đầu của một dãy mới. |
| Sliding | Kiểu scrolling giống Yahoo!-like mà vị trí của trang hiện tại nằm ở trung tâm của dãy các trang, hoặc càng gần càng tốt. Đây là kiểu mặt định. |

### Sử dụng Paginator để phân trang

Bước 1: Tạo file pager.phtml để hiển thị dòng phân trang theo kiểu ‘Sliding’ và lưu vào thư mục view của module đang làm việc.

Ví dụ: zend\_app\module\Product\view\pager.phtml

Bước 2: Gọi sử dụng lớp Paginator cùa Zend trong phương thức action sẽ thực hiện phân trang trong file Xcontroller.php

Ví dụ: Gọi sử dụng lớp Paginator của Zend trong phương thức adminAction() trong file zend\_app\module\Product\src\Product\Controller\ProductController.php

public function adminAction()

{

$page = (int) $this->params()->fromRoute('id', 0);

$dataItems = $this->getProductTable()->fetchAllForPagination();

$iteratorAdapter = new \Zend\Paginator\Adapter\Iterator($dataItems);

$paginator = new \Zend\Paginator\Paginator($iteratorAdapter);

$paginator->setCurrentPageNumber($page);

$paginator->setItemCountPerPage(4);

return new ViewModel(array(

'products' => $paginator,

));

}

Bước 3: Xây dựng thêm phương thức fetchAllForPagination() trong tập tin XTable.php của thư mục zend\_app\module\Product\src\Product\Model: để thực hiện lấy các mẩu tin dành cho việc phân trang:

Ví dụ: zend\_app\module\Product\src\Product\Model\ProductTable.php

public function fetchAllForPagination()

{

$resultSet = $this->tableGateway->select();

$resultSet->buffer();

$resultSet->next();

return $resultSet;

}

Bước 4: Ở trang hiển thị danh mục theo phân trang, ta gọi sử dụng phân trang ở cuối trang này.

Ví dụ: Ở cuối trang quản lý danh mục: admin.phtml

…

<?php

echo ( count($products) > 0 ) ? $this->paginationControl($products, 'Sliding', 'pager.phtml') : "";

?>

Bước 5: Kiểm tra xem có cần phải điều chỉnh route thì điều chỉnh tại file module.config.php của module đang làm việc

Ví dụ:

…

'router' => array(

'routes' => array(

'product' => array(

'type' => 'segment',

'options' => array(

'route' =>

'/product[/:action][/:id][/title/:title][/]',

'constraints' => array(

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

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

),

'defaults' => array(

'controller' => 'Product\Controller\Product',

'action' => 'index',

),

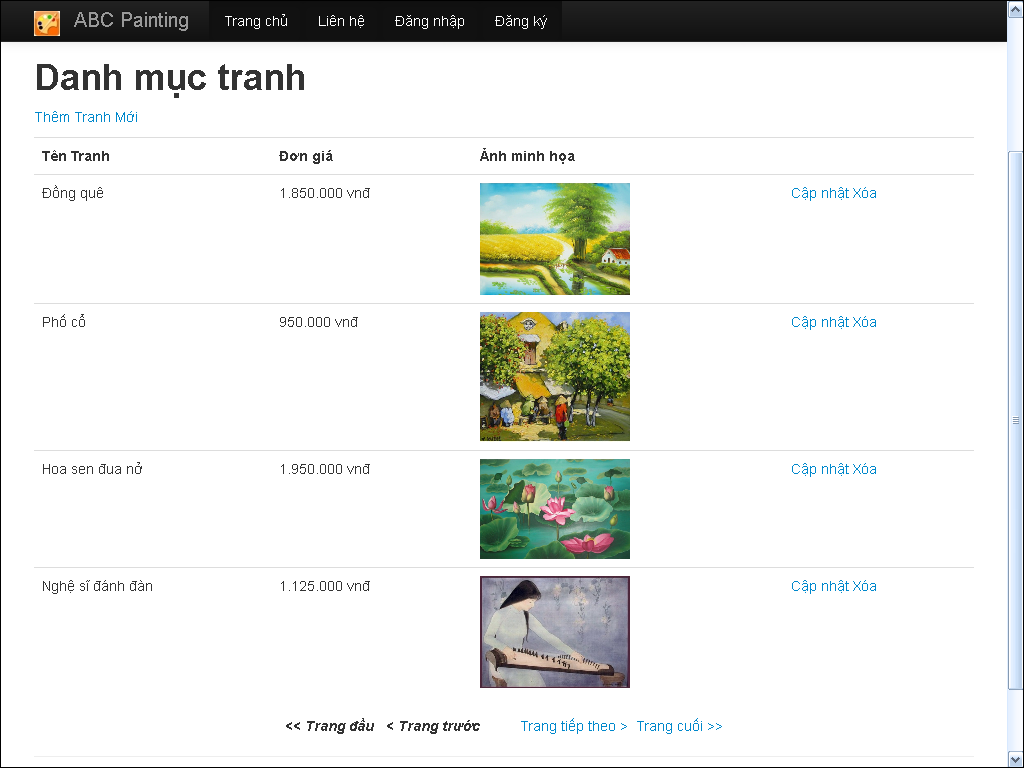
),

),

),

),

…



# Captcha

**Nội dung:**

* **Giới thiệu**
* **Captcha Operation**
* **Captcha Adapter**
* **Thực hiện Captcha**

## Giới thiệu:

* CAPTCHA (“Completely Automated Public Turing test to tell Computers and Humans Apart”) được sử dụng trên trang để đảm bảo là các thông tin cá nhân là được một người nhập vào chứ không phải là quá trình nhập tự động. CAPTCHA thường được sử dụng trong form khi người dùng điền và xác nhận thông tin.
* CAPTCHA có thể hiển thị dưới nhiều hình thức khác nhau.
  + Ví dụ: câu hỏi logic, trình bày font chữ trên hình ảnh, hiển thị nhiều hình ảnh kèm câu hỏi…

Zend\Captcha component cung cấp các backends của CAPTCHA có thể sử dụng độc lập cũng có thể kết hợp với các thành phần của Zend/Form.

## Captcha Operation:

Tất cả các *CAPTCHA* adapter đều được implement từ Zend\Captcha\AdapterInterface, nội dung như sau.

namespace Zend\Captcha;

use Zend\Validator\ValidatorInterface;

interface AdapterInterface extends ValidatorInterface

{

public function generate();

public function setName($name);

public function getName();

// Get helper name used for rendering this captcha type

public function getHelperName();

}

* + Trong đó, phương thức generate() được sử dụng để tạo ra mã CAPTCHA – gọi là token. Thông thường, quá trình này sẽ lưu trữ token trong session để ta có thể so sánh nó trong các yêu cầu tiếp theo.
  + Phương thức render() tạo ra thông tin hiển thị cho captcha như hình ảnh, figlet hoặc các CAPTCHA khác.
  + *Chú ý:* Thường ta hay sử dụng các chức năng của Zend\Captcha với các thành phần của Zend\Form.

Ví dụ: Một trường hợp có sử dụng CAPTCHA đơn giản như sau

// Originating request:

$captcha = new Zend\Captcha\Figlet(array(

'name' => 'foo',

'wordLen' => 6,

'timeout' => 300,

));

$id = $captcha->generate();

//this will output a Figlet string

echo $captcha->getFiglet()->render($captcha->getWord());

// On a subsequent request:

// Assume a captcha setup as before, with corresponding form fields, the value of $\_POST['foo']

// would be key/value array: id => captcha ID, input => captcha value

if ($captcha->isValid($\_POST['foo'], $\_POST)) {

// Validated!

}

## Captcha Adapter

### Các adapter dưới đây là các adapter mặc định của Zend Framework:

* + Zend\Captcha\AbstractWord
  + Zend\Captcha\Dumb
  + Zend\Captcha\Figlet
  + Zend\Captcha\Image
  + Zend\Captcha\ReCaptcha

### Zend\Captcha\AbstractWord

* + Zend\Captcha\AbstractWord là một adapter trừu tượng phục vụ như lớp cơ sở cho hầu hết các *CAPTCHA* adapter khác. Nó cung cấp các mutators để xác định word length, session *TTL* và session container object để sử dụng. Zend\Captcha\AbstractWord cũng đóng gói các validation logic.
  + Mặc định, chiều dài của từ (word length) là 8 ký tự, session timeout là 5 phút, và Zend\Session\Container được sử dụng cho persistence (sử dụng namespace “Zend\Form\Captcha\<captcha ID>”).
  + Ngoài ra còn có các phương thức được yêu cầu bởi Zend\Captcha\AdapterInterface interface.

### Zend\Captcha\Dump

* + Zend\Captcha\Dumb adapter chủ yếu là tự mô tả (self-descriptive). Nó cung cấp một chuỗi ngẫu nhiên là phải được gõ ngược để xác nhận.
  + Thực tế, nó không phải là một giải pháp CAPTCHA tốt và chỉ nên dùng để thử nghiệm.
  + Zend\Captcha\Dumb kế thừa Zend\Captcha\AbstractWord.

### Zend\Captcha\Figlet

* + Zend\Captcha\Figlet adapter sử dụng Zend\Text\Figlet để trình bày một figlet cho người sử dụng.
  + Các tùy chọn được bỏ qua ở phương thức khỏi tạo cũng được bỏ qua với đối tượng Zend*\Text\Figlet* object. Xem thêm tài liệu về *Zend\Text\Figlet* để biết các thông tin chi tiết về cấu hình các tùy chọn có sẵn.

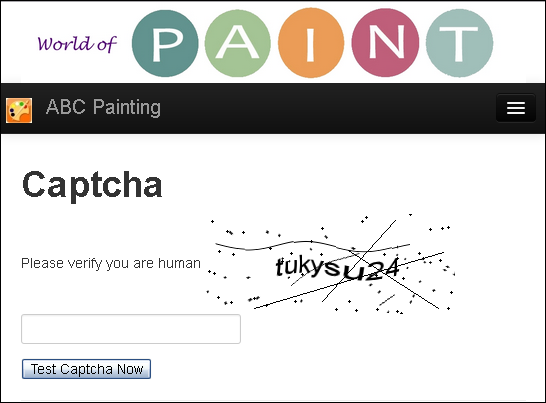
### Zend\Captcha\Image

* + Zend\Captcha\Image adapter lấy từ được tạo ra và làm cho nó trở thành một hình ảnh, thực hiện các hoán vị skewing khác nhau làm cho nó trở nên khó khăn khi tự động giải mã. Nó yêu cầu phải có GD extension biên dịch với hỗ trợ TrueType hoặc Freetype. Hiện tại, Zend\Captcha\Image adapter chỉ có thể tạo ra hình ảnh loại PNG.
  + Zend\Captcha\Image kế thừa từ Zend\Captcha\AbstractWord.

### Zend\Captcha\Recaptcha

* + Zend\Captcha\ReCaptcha adapter sử dụng Zend*\Service\ReCaptcha\ReCaptcha* để tạo ra và xác định các *CAPTCHA*.

## Thực hiện Captcha



### Chuẩn bị:

* + Tạo một thư mục dùng để chứa hình ảnh captcha, và thiết lập quyền được ghi cho thư mục này
    1. Ví dụ: Tạo thư mục captcha trong thư mục public/images
  + Tạo thư mục fonts dùng để chứa font chữ sẽ hiển thị trong hình ảnh captcha và thiết lập quyền chỉ đọc cho thư mục này
    1. Ví dụ: Tạo thư mục fonts trong thư mục public/images
  + Donwload và chép tập tin arial.ttf vào thư mục fonts vừa tạo

Ví dụ: chép tập tin arial.ttf vào public/images/fonts.

* Bước 1: Tạo module “Captcha” nằm trong thư mục module của ứng dụng có cấu trúc tương tự như những module khác:
  + Captcha
    1. module.php: cấu hình load tự động cho module Captcha
    2. Config
       1. module.config.php: đăng ký và định tuyến cho các thành phần trong module Captcha
    3. src
       1. Captcha
          - Controller

CaptchaController.php: xây dựng các action xử lý cho Captcha

* + - * + Form

TestCaptchaForm.php: xây dựng các thành phần tạo ra form Captcha

* + 1. view
       1. captcha
          - captcha

Form.phtml: view hiển thị của form Captcha

* Bước 2: Đăng ký module Captcha trong tập tin application.config.php đặt tại thư mục config của ứng dụng:
  + Đăng ký:

return array(

'modules' => array(

'Application',

'Product',

'Auth',

'Contact',

**‘Captcha'**

),

* Bước 3: Đăng ký và định tuyến cho các thành phần trong module Captcha
  + Đăng ký:

'controllers' => array(

'invokables' => array(

'Captcha\Controller\Captcha' => 'Captcha\Controller\CaptchaController',

),

),

Định tuyến:

'router' => array(

'routes' => array(

‘captcha' => array(

'type' => 'segment',

'options' => array(

'route'=> ‘/captcha[/:action][/:id]',

'constraints' => array(

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

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

),

'defaults' => array(

'controller' => ‘Captcha\Controller\Captcha',

'action' => 'form',

),

),

…

* Bước 4: Xây dựng các thành phần tạo ra form Captcha (CaptchaForm)
  + Gồm có: một lable, một image captcha, một textfield để người dùng nhập nội dung captcha, và một button để submit



namespace Captcha\Form;

use Zend\Form\Form,

Zend\Form\Element\Captcha,

Zend\Captcha\Image as CaptchaImage;

class TestCaptchaForm extends Form

{ public function \_\_construct($urlcaptcha = null)

{ parent::\_\_construct('Test Form Captcha');

$this->setAttribute('method', 'post');

// thư mục chứa thư mục captcha và thư mục fonts

$dirdata = './public/images';

// thiết lập các thông số cho captchaImage

$captchaImage = new CaptchaImage( array(

'font' => $dirdata . '/fonts/arial.ttf',

'width' => 250,

'height' => 100,

'dotNoiseLevel' => 40,

'lineNoiseLevel' => 3)

);

// thiết lập đường dẫn thư mục chứa captchaImage

$captchaImage->setImgDir($dirdata.'/captcha');

$captchaImage->setImgUrl($urlcaptcha);

// tạo các thành phần trong form Captcha

$captcha = new Captcha('captcha');

$captcha->setCaptcha($captchaImage);

$captcha->setLabel('Please verify you are human ');

$this->add($captcha);

$this->add(array(

'name' => 'submit',

'attributes' => array(

'type' => 'submit',

'value' => 'Test Captcha Now'

),

));

}

}

* Bước 5: Xây dựng các action xử lý cho Captcha
  + generateAction(): dùng để tạo ra captchaImage và đưa vào thư mục chứa captchaImage đã chuẩn bị trước
  + formAction(): gọi sử dụng captcha form đã được xậy dựng ở bước trên và kiểm tra thông tin người dùng nhập dựa trên CaptchaImage

namespace Captcha\Controller;

use Zend\Mvc\Controller\AbstractActionController,

Captcha\Form\TestCaptchaForm;

class CaptchaController extends AbstractActionController

{ public function generateAction()

{ $response = $this->getResponse();

$response->getHeaders()->addHeaderLine('Content-Type', "image/png");

$id = $this->params('id', false);

if ($id) {

$image = './public/images/captcha' . $id;

if (file\_exists($image) !== false) {

$fp = fopen($image,"r");

$imageread = fpassthru($fp);

$response->setStatusCode(200);

$response->setContent($imageread);

if (file\_exists($image) == true) { unlink($image); }

}

}

return $response;

}

public function formAction()

{

$form = new TestCaptchaForm($this->getRequest()->getBaseUrl().'/images/captcha/');

$request = $this->getRequest();

if ($request->isPost()) {

//set data post

$form->setData($request->getPost());

if ($form->isValid()) {

echo "captcha is valid ";

// và thực hiện các công việc khác

}

}

return array('form' => $form);

}

}

* Bước 6: Tạo view hiển thị của form Captcha

$form = $this->form;

$form->setAttribute('action',

$this->url(‘captcha/captcha\_form',

array('controller'=>’captcha', 'action' => 'form')));

$form->prepare();

echo $this->form()->openTag($form);

echo $this->formRow($form->get('captcha'));

echo $this->formSubmit($form->get('submit'));

echo $this->form()->closeTag();

# ZEND\CONFIG\READER và WRITER

**Nội dung:**

**Zend\Config\Reader**

* Zend\Config\Reader\Ini
* Zend\Config\Reader\Xml
* Zend\Config\Reader\Json
* Zend\Config\Reader\Yaml

**Zend\Config\Writer**

* Zend\Config\Writer\Ini
* Zend\Config\Writer\Xml
* Zend\Config\Writer\Json
* Zend\Config\Writer\Yaml

## Zend\Config\Reader

Zend\Config\Reader cho bạn khả năng để đọc một tập tin cấu hình. Nó hoạt động với việc triển khai cụ thể cho các định dạng tập tin khác nhau. Zend\Config\Reader chỉ một giao diện, xác định hai phương thức fromFile() và fromString(). Việc triển khai cụ thể của giao diện này là:

* Zend\Config\Reader\Ini
* Zend\Config\Reader\Xml
* Zend\Config\Reader\Json
* Zend\Config\Reader\Yaml

Phương thức fromFile() và fromString() trả về một mảng PHP chứa dữ liệu của tập tin cấu hình.

### Zend\Config\Reader\Ini

**Zend\Config\Reader\Ini** cho phép các nhà phát triển lưu trữ dữ liệu cấu hình trong một định dạng INI quen thuộc và đọc chúng trong ứng dụng bằng cách sử dụng một cú pháp mảng.

Zend\Config\Reader\Ini sử dụng hàm parse\_ini\_file() PHP.

Chú ý: Key phân cách

Theo mặc định, ký tự phân cách là ("."). Điều này có thể được thay đổi, bằng cách sử dụng phương thức setNestSeparator(). Ví dụ:

$reader = **new** Zend\Config\Reader\Ini();

$reader->setNestSeparator(‘-’);

Ví dụ:

webhost = ‘www.example.com’

database.adapter = ‘pdo\_mysql’

database.params.host = ‘db.example.com’

database.params.username = ‘dbuser’

database.params.password = ‘secret’

database.params.dbname = ‘dbproduction’

Chúng ta có thể sử dụng Zend\Config\Reader\Ini để đọc file INI này:

$reader = **new** \Zend\Config\Reader\Ini();

$data = $reader->fromFile('./public/config/config.ini');

**echo** $data['webhost']; // prints "www.example.com"

**echo** $data['database']['params']['dbname'];

// prints "dbproduction"

### Zend\Config\Reader\Xml

Zend \ Config \ đọc \ Xml cho phép các nhà phát triển đọc dữ liệu cấu hình trong một định dạng XML quen thuộc và đọc chúng trong ứng dụng bằng cách sử dụng một cú pháp mảng. Phần tử gốc của file XML hoặc chuỗi là không liên quan và có thể đặt tên tùy ý.

Ví dụ sau đây minh họa việc sử dụng cơ bản của Zend \ Config \ đọc \ Xml cho cấu hình tải dữ liệu từ một tập tin XML. Giả sử chúng ta có các tập tin cấu hình XML sau đây:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<danh\_sach>

<hoc\_vien ho\_ten=*"Phạm Công Anh"* ngay\_sinh=*"27/10/1988"* dia\_chi=*"197 Trần Hưng Đạo"* dien\_thoai=*"98877668"* />

<hoc\_vien ho\_ten=*"Lê Việt Nhân"* ngay\_sinh=*"02/04/1987"* dia\_chi=*"179 Chánh hưng"* dien\_thoai=*"8504122"* />

<hoc\_vien ho\_ten=*"Phạm Phan Trung"* ngay\_sinh=*"03/06/1988"* dia\_chi=*"124 Bắc Hải P.6 Q.TB"* dien\_thoai=*"0918121188"* />

</danh\_sach>

Chúng ta có thể sử dụng Zend\Config\Reader\Xml để đọc file XML này:

$reader = **new** \Zend\Config\Reader\Xml();

$data = $reader->fromFile('./public/config/config.xml');

**foreach** ($data['hoc\_vien'] **as** $hv)

{

**echo** $hv['ho\_ten']; **echo** ' | ';

**echo** $hv['dia\_chi']; **echo** ' | ';

**echo** $hv['dien\_thoai']; **echo** ' <br> ';

}

### Zend\Config\Reader\Json

Zend\Config\Reader\Json cho phép các nhà phát triển đọc dữ liệu cấu hình trong một định dạng JSON và đọc chúng trong ứng dụng bằng cách sử dụng một cú pháp mảng.

Ví dụ:

{

"webhost" : "www.example.com",

"database" : {

"adapter" : "pdo\_mysql",

"params" : {

"host" : "db.example.com",

"username" : "dbuser",

"password" : "secret",

"dbname" : "dbproduction"

}

}

}

Chúng ta có thể sử dụng Zend\Config\Reader\Json để đọc file JSON này:

$reader = **new** \Zend\Config\Reader\Json();

$data = $reader->fromFile('./public/config/config.json');

**echo** $data['webhost']; // prints "www.example.com"

**echo** $data['database']['params']['dbname']; // prints "dbproduction"

### Zend\Config\Reader\Yaml

Zend\Config\Reader\Yaml cho phép các nhà phát triển để đọc dữ liệu cấu hình ở định dạng YAML và đọc chúng trong ứng dụng bằng cách sử dụng một cú pháp mảng**.**

Ví dụ sau đây minh họa việc sử dụng cơ bản của Zend\Config\Reader\Yaml sử dụng phần mở rộng YAML PECL. Giả sử chúng ta có các tập tin cấu hình YAML sau:

webhost: www.example.com

database:

adapter: pdo\_mysql

params:

host: db.example.com

username: dbuser

password: secret

dbname: dbproduction

Chúng ta có thể sử dụng Zend\Config\Reader\Yaml để đọc file Yaml này:

**require\_once** ('./public/config/spyc.php');

$reader = **new** \Zend\Config\Reader\Yaml(**array**('Spyc','YAMLLoadString'));

$data = $reader->fromFile('./public/config/config.yaml');

**echo** $data['webhost']; // prints "www.example.com"

**echo** $data['database']['params']['dbname']; // prints "dbproduction"

## ZEND\CONFIG\WRITER

Zend\Config\Writer cung cấp cho bạn khả năng ghi tập tin cấu hình trong mảng, Zend\Config\Config và bất kỳ đối tượng. Zend\Config\Writer là một giao diện định nghĩa hai phương thức toFile() và toString(). Chúng tôi có năm ghi cụ thể thực hiện giao diện này:

Zend\Config\Writer\Ini

Zend\Config\Writer\Xml

Zend\Config\Writer\PhpArray

Zend\Config\Writer\Json

Zend\Config\Writer\Yaml

### Zend\Config\Writer\Ini

Ghi INI có hai chế độ để biểu diễn liên quan đến các phần. Theo mặc định, cấu hình cấp cao nhất là luôn luôn ghi vào tên phần. Bằng cách gọi $writer->setRenderWithoutSectionsFlags(true), tất cả các tùy chọn được viết vào không gian tên toàn cầu của các tập tin INI và không có phần được áp dụng.

Như một sự bổ sung Zend\Config\Writer\Ini thêm tham số tùy chọn nestSeparator, trong đó xác định với ký tự mà các nút đơn được tách ra (mặc định là chấm “.”).

Khi sửa đổi hoặc tạo ra một đối tượng Zend\Config\Config, có một số điều cần biết. Để tạo ra hoặc sửa đổi một giá trị, bạn chỉ đơn giản nói rằng thiết lập các tham số của các đối tượng cấu hình thông qua tham số accessor (->).Để tạo ra một phần trong thư mục gốc hoặc để tạo ra một chi nhánh, bạn chỉ cần tạo một mảng mới (“$config->branch = array();”).

Ví dụ:

$config = **new** \Zend\Config\Config(**array**(), **true**);

$config->production = **array**();

$config->production->webhost = 'www.example.com';

$config->production->database = **array**();

$config->production->database->params = **array**();

$config->production->database->params->host = 'localhost';

$config->production->database->params->username = 'production';

$config->production->database->params->password = 'secret';

$config->production->database->params->dbname = 'dbproduction';

$writer = **new** \Zend\Config\Writer\Ini();

**echo** $writer->toFile("./public/config/tam.ini", $config);

Kết quả

[production]

webhost = "www.example.com"

database.params.host = "localhost"

database.params.username = "production"

database.params.password = "secret"

database.params.dbname = "dbproduction"

### Zend\Config\Writer\Xml

Zend\Config\Writer\Xml có thể được sử dụng để tạo ra một chuỗi XML hoặc tập tin bắt đầu từ một đối tượng Zend\Config\Config.

Ví dụ:

$config = **new** \Zend\Config\Config(**array**(), **true**);

$config->production = **array**();

$config->production->webhost = 'www.example.com';

$config->production->database = **array**();

$config->production->database->params = **array**();

$config->production->database->params->host = 'localhost';

$config->production->database->params->username = 'production';

$config->production->database->params->password = 'secret';

$config->production->database->params->dbname = 'dbproduction';

$writer = **new** \Zend\Config\Writer\Xml();

**echo** $writer->toFile("./public/config/tam.xml", $config);

kết quả

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<zend-config>

<production>

<webhost>www.example.com</webhost>

<database>

<params>

<host>localhost</host>

<username>production</username>

<password>secret</password>

<dbname>dbproduction</dbname>

</params>

</database>

</production>

</zend-config>

Ví dụ 2:

$config = **new** \Zend\Config\Config(**array**(), **true**);

$config->sv1 = **array**();

$config->sv1->hoten = 'www.example.com';

$config->sv1->ngaysinh = '1/1/1996';

$config->sv1->noisinh = 'asdasd';

$config->sv2 = **array**();

$config->sv2->hoten = 'www.example.com';

$config->sv2->ngaysinh = '1/1/1996';

$config->sv2->noisinh = 'asdasd';

$writer = **new** \Zend\Config\Writer\Xml();

**echo** $writer->toFile("./public/config/tam.xml", $config);

Kết quả

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<zend-config>

<sv1>

<hoten>www.example.com</hoten>

<ngaysinh>1/1/1996</ngaysinh>

<noisinh>asdasd</noisinh>

</sv1>

<sv2>

<hoten>www.example.com</hoten>

<ngaysinh>1/1/1996</ngaysinh>

<noisinh>asdasd</noisinh>

</sv2>

</zend-config>

### Using Zend\Config\Writer\Json

Zend\Config\Writer\JSON có thể được sử dụng để tạo ra một mã PHP trả về các đại diện JSON của một đối tượng Zend \ Config \ Config.

$config = **new** \Zend\Config\Config(**array**(), **true**);

$config->production = **array**();

$config->production->webhost ='www.example.com';

$config->production->database = **array**();

$config->production->database->params = **array**();

$config->production->database->params->host ='localhost';

$config->production->database->params->username ='production';

$config->production->database->params->password ='secret';

$config->production->database->params->dbname ='dbproduction';

$writer = **new** \Zend\Config\Writer\Json();

**echo** $writer->toFile("./public/config/tam.json", $config);

Kết quả:

{

"production":

{

"webhost":"www.example.com",

"database":

"params":

{

"host":

"localhost",

"username": "production",

"password": "secret",

"dbname":"dbproduction"

}

}

}

### Zend\Config\Writer\Yaml

Ví dụ này minh họa cơ bản của việc sử dụng Zend\Config\Writer\YAML để tạo ra một tập tin cấu hình mới sử dụng phần mở rộng YAML PECL:

$config = **new** \Zend\Config\Config(**array**(), **true**);

$config->production = **array**();

$config->production->webhost ='www.example.com';

$config->production->database = **array**();

$config->production->database->params = **array**();

$config->production->database->params->host ='localhost';

$config->production->database->params->username ='production';

$config->production->database->params->password ='secret';

$config->production->database->params->dbname ='dbproduction';

**require\_once** ('./public/config/spyc.php');

$writer = **new** \Zend\Config\Writer\Yaml(**array**('Spyc','YAMLDump'));

**echo** $writer->toFile("./public/config/tam.yaml", $config);

Kết quả

---

production:

webhost: www.example.com

database:

params:

host: localhost

username: production

password: secret

dbname: dbproduction

# Mail

**Nội dung:**

* Zend\Mail\Message
* Zend\Mail\Transport
* Zend\Mail\Transport\SmtpOptions
* Zend\Mail\Transport\FileOptions

## Giới thiệu

* Zend\Mail cung cấp chức năng tổng quát để soạn và gửi mail dưới dạng văn bản và tin nhắn có nội dung tổng hợp MIME.
* Mail có thể được gửi thông qua Mail\Transport\SendMail, Mail\Transport\Smtp hoặc Mail\Transport\File.
* Tuy nhiên, ta cũng có thể thực hiện việc gửi mail theo cách riêng của chúng ta bằng cách implement Mail\Transport\TransportInterface.
* Ví dụ: Một email đơn giản sẽ có: người nhận, chủ đề, nội dung và người dùng. Để gửi mail này ta sử dụng Zend\Mail\Transport\Sendmail:

use Zend\Mail;

$mail = new Mail\Message();

$mail->setBody('This is the text of the email.');

$mail->setFrom('Freeaqingme@example.org', 'Sender\'s name');

$mail->addTo('Matthew@example.com', 'Name o. recipient');

$mail->setSubject('TestSubject');

$transport = new Mail\Transport\Sendmail();

$transport->send($mail);

* Ghi chú: Để gửi mail bằng Zend/Mail, ta phải chỉ định ít nhất một người nhận cũng như nội dung. Và đối với mỗi Transport có thể yêu cầu các tham số khác cần phải thiết lập.
* Với hầu hết các thuộc tính của mail, có thể dùng phương thức “get” để đọc thông tin được lưu trữ trong đối tượng message.
* Ta cũng có thể sử dụng hầu hết các phương thức của Mail\Message cùng lúc để thiết lập các thông số khi gửi mail.

use Zend\Mail;

$mail = new Mail\Message();

$mail->setBody('This is the text of the mail.')

->setFrom('somebody@example.com', 'Some Sender')

->addTo('somebody\_else@example.com', 'Some Recipient')

->setSubject('TestSubject');

* Nếu muốn thì một tin nhắn email có nhiều phần cũng có thể được tạo ra bằng cách sử dụng component Zend/Mime để gắn các phần này vào nội dung của tin nhắn.
* Lớp Message chỉ là một đối tượng giá trị đơn giản, tự bản thân nó không có khả năng gửi và lưu trữ chính nó, vì vậy nếu cần thì ta cần phải sử *dụng* Storage adapter *or* Transport adapter.
* Có thể sử dụng rất nhiều phương thức trong lớp Message, ví dụ như:
* isValid(): để kiểm tra message (tin nhắn) có hợp lệ không
* setEncoding(string $encoding): đặt bộ hiển thị nội dung

Ví dụ: $message->setEncoding("UTF-8");

* getEncoding(): lấy bộ hiển thị nội dung
* addFrom(…): danh sách người gửi
* from(): truy cập danh sách người gửi, trả về Zend\Mail\AddressList instance.
* Và rất nhiều phương thức khác…
* Ví dụ:

use Zend\Mail\Message;

$message = new Message();

$message->addFrom("matthew@zend.com", "Matthew Weier O'Phinney")

->addTo("foobar@example.com")

->setSubject("Sending an email from Zend\Mail!");

$message->setBody("This is the message body.");

* RFC822 cũng cho phép có nhiều người gửi cùng một mail.

Ví dụ:

$message->addFrom("ralph.schindler@zend.com", "Ralph Schindler")

->addFrom("enrico.z@zend.com", "Enrico Zimuel")

->setSender("matthew@zend.com", "Matthew Weier O'Phinney");

* Nếu là tin nhắn có kèm theo nội dung HTML hoặc nội dung có nhiều phần thì cần sử dụng đối tượng MIME message:

use Zend\Mail\Message;

use Zend\Mime\Message as MimeMessage;

use Zend\Mime\Part as MimePart;

$text = new MimePart($textContent);

$text->type = "text/plain";

$html = new MimePart($htmlMarkup);

$html->type = "text/html";

$image = new MimePart(fopen($pathToImage, 'r'));

$image->type = "image/jpeg";

$body = new MimeMessage();

$body->setParts(array($text, $html, $image));

$message = new Message();

$message->setBody($body);

* Các Transport làm công việc phân phối thực tế của mail.
* Thông thường, ta chỉ cần quan tâm về hai khả năng: sử dụng phương thức mail() gốc của PHP, trong đó sử dụng tài nguyên hệ thống để gửi mail, hoặc sử dụng giao thức SMTP để chuyển mail thông qua máy chủ (remote server).
* ZF cũng có “File” transport, giúp tạo ra tập tin mail dành cho các tin nhắn được gửi; sau này có thể được xem như các bản ghi (log) cho mục đích gửi thông qua một cơ chế gửi thay thế sau này.
* Zend\Mail\Transport interface xác định chính xác một phương thức send(). Phương thức này chấp nhận một Zend\Mail\Message instance, mà sau đó đồng bộ để gửi.

## Sử dụng mail transport

### Sendmail Transport

use Zend\Mail\Message;

use Zend\Mail\Transport\Sendmail as SendmailTransport;

$message = new Message();

$message->addTo('matthew@zend.com')

->addFrom('ralph.schindler@zend.com')

->setSubject('Greetings and Salutations!')

->setBody("Sorry, I'm going to be late today!");

$transport = new SendmailTransport();

$transport->send($message);

### SMTP Transport

use Zend\Mail\Message;

use Zend\Mail\Transport\Smtp as SmtpTransport;

use Zend\Mail\Transport\SmtpOptions;

$message = new Message();

$message->addTo('matthew@zend.com')

->addFrom('ralph.schindler@zend.com')

->setSubject('Greetings and Salutations!')

->setBody("Sorry, I'm going to be late today!");

// Setup SMTP transport using LOGIN authentication

$transport = new SmtpTransport();

$options = new SmtpOptions(array(

'name' => 'localhost.localdomain',

'host' => '127.0.0.1',

'connection\_class' => 'login',

'connection\_config' => array(

'username' => 'user',

'password' => 'pass',

),

));

$transport->setOptions($options);

$transport->send($message);

### File Transport

use Zend\Mail\Message;

use Zend\Mail\Transport\File as FileTransport;

use Zend\Mail\Transport\FileOptions;

$message = new Message();

$message->addTo('matthew@zend.com')

->addFrom('ralph.schindler@zend.com')

->setSubject('Greetings and Salutations!')

->setBody("Sorry, I'm going to be late today!");

 // Setup SMTP transport using LOGIN authentication

$transport = new FileTransport();

$options = new FileOptions(array(

'path' => 'data/mail/',

'callback' => function (FileTransport $transport) {

return 'Message\_' . microtime(true) . '\_' . mt\_rand() . '.txt';

},

));

$transport->setOptions($options);

$transport->send($message);

# SESSION

**Nội dung: Session**

* Session Config
* Session Container
* Session Manager
* Session Save Handlers
* Session Storage

## Session Config

Zend Framework đi kèm với một tập hợp của các lớp config chuẩn bị sẵn để bạn sử dụng. Config xử lý các cài đặt khác nhau cấu hình như cookie lives, lifetime, bao gồm một số các bit để cấu hình ext/session khi sử dụng Zend\Session\Config\SessionConfig

### Standard Config

Zend\Session\Config\StandardConfig cung cấp cho bạn một giao diện cơ bản để thực hiện session khi không tận dụng ext/session. Điều này được sử dụng nhiều hơn cho các trường hợp đặc biệt như khi bạn có thể quản lý session được thực hiện bởi một hệ thống khác

#### Tùy chọn cấu hình cơ bản

|  |  |  |
| --- | --- | --- |
| **Option Data** | **Type** | **Description** |
| cache\_expire | integer | Quy định cụ thể thời gian để sống cho các trang session lưu trữ trong vài phút |
| cookie\_domain | string | Quy định cụ thể tên miền để thiết lập trong các session cookie |
| cookie\_httponly | boolean | Đánh dấu các tập tin cookie như truy cập chỉ thông qua giao thức HTTP |
| cookie\_lifetime | integer | Xác định thời gian tồn tại của cookie trong giây được gửi đến trình duyệt. |
| cookie\_path | string | Xác định đường dẫn để thiết lập các session cookie. |
| cookie\_secure | boolean | Chỉ định các tập tin cookie chỉ nên được gửi qua kết nối an toàn. |
| entropy\_length | integer | Quy định cụ thể số lượng các byte sẽ được đọc từ tập tin quy định tại entropy\_file. |
| entropy\_file | string | Định nghĩa đường dẫn đến tài nguyên bên ngoài (tập tin) sẽ được sử dụng như một dữ liệu ngẫu nhiên bổ sung. |
| gc\_maxlifetime | integer | Chỉ định số giây mà sau đó dữ liệu sẽ được xem như 'rác'. |
| gc\_divisor | integer | Xác định xác suất mà quá trình gc được bắt đầu trên tất cả các phiên khởi tạo. |
| gc\_probability | integer | Xác định khả năng mà quá trình gc được bắt đầu trên tất cả các phiên khởi tạo. |
| hash\_bits\_per\_character | integer | Xác định có bao nhiêu bit được lưu trữ trong mỗi ký tự khi chuyển đổi dữ liệu băm nhị phân. |
| name | string | Chỉ định tên của session được sử dụng như tên của cookie. |
| remember\_  me\_seconds | integer | Xác định bao lâu để nhớ session trước khi dữ liệu bị xóa. |
| save\_path | string | Định nghĩa tham số được thông qua để xử lý lưu lại dữ kiện |
| use\_cookies | boolean | Chỉ định các mô-đun sẽ sử dụng cookie để lưu trữ các session id. |

#### Cách sử dụng cơ bản

**use** Zend\Session\Config\StandardConfig;

**use** Zend\Session\SessionManager;

$config = **new** StandardConfig();

$config->setOptions(**array**(

’remember\_me\_seconds’ => 1800,

’name’ => ’zf2’,

));

$manager = **new** SessionManager($config);

## Session Config

Zend\Session\Config\SessionConfig cung cấp cho bạn một giao diện cơ bản để thực hiện session khi mà tận dụng của PHP ext/session. Tùy chọn cấu hình hầu hết các cấu hình hoặc Zend\Session\Storage hoặc cấu hình ext/session trực tiếp.

### Tùy chọn cấu hình cơ bản

Các tùy chọn cấu hình sau đây được xác định bởi **Zend\Session\Config\SessionConfig**, lưu ý rằng nó được thừa hưởng tất cả các cấu hình từ **Zend\Session\Config\StandardConfig**

|  |  |  |
| --- | --- | --- |
| Option | Data  Type | Description |
| cache\_limiter | string | Chỉ định phương pháp điều khiển bộ nhớ cache được sử dụng các trang session. |
| hash\_function | string | Cho phép bạn chỉ định các thuật toán băm được sử dụng để tạo ra các session IDs. |
| php\_save\_handler | string | Định nghĩa tên của một save\_handler PHP nhúng vào PHP. |
| serialize\_  handler | string | Định nghĩa tên của trình xử lý được sử dụng để serialize / deserialize dữ liệu. |
| url\_rewriter\_tags | string | Quy định cụ thể trong đó thẻ HTML được viết lại để bao gồm session id nếu transparent sid kích hoạt. |
| use\_trans\_sid | boolean | Xem sid transparent được kích hoạt hay không. |

### Cách sử dụng cơ bản

**use** Zend\Session\Config\SessionConfig;

**use** Zend\Session\SessionManager;

$config = **new** SessionConfig();

$config->setOptions(**array**(

’phpSaveHandler’ => ’redis’,

’savePath’ => ’tcp://127.0.0.1:6379?weight=1&timeout=1’,

));

$manager = **new** SessionManager($config);

## SESSION CONTAINER

* **Zend\Session\container** cung cấp các API chính cho các thao tác dữ liệu trong session Zend Framework. Container được sử dụng để tách tất cả các dữ liệu session làm việc, mặc dù một không gian tên mặc định tồn tại cho những người chỉ muốn một không gian tên cho tất cả các dữ liệu session của họ.
* Mỗi trường hợp của **Zend\Session\container** tương ứng với một mục nhập của **Zend\Session\Storage**, nơi mà không gian tên được sử dụng như là chìa khóa. **Zend\Session\container** chính nó là một thể hiện của một ArrayObject

### Cách sử dụng cơ bản

**use** Zend\Session\Container;

$container = **new** Container(’namespace’);

$container->item = ’foo’;

### Thiết lập mặc định Session Manager

Trong trường hợp bạn đang sử dụng nhiều session managers hoặc muốn được rõ ràng, session manager mặc định được sử dụng có thể được thiết lập một cách rõ ràng.

**use** Zend\Session\Container;

**use** Zend\Session\SessionManager;

$manager = **new** SessionManager();

Container::setDefaultManager($manager);

## SESSION MANAGER

* Quản lý session, **Zend\Session\SessionManager**, là một lớp chịu trách nhiệm cho tất cả các khía cạnh của quản lý session. Nó khởi tạo và cấu hình configuration, lưu trữ và xử lý lưu. Thêm quản lý session có thể được đưa vào trong session container để cung cấp một gói hoặc tên miền không gian xung quanh session dữ liệu của bạn.
* Trình quản lý session chịu trách nhiệm cho session start, session exists, session write, tái tạo id, thời gian để sống và thời gian tự phá hủy. Trình quản lý session có thể session hợp lệ từ một chuỗi xác nhận để đảm bảo rằng dữ liệu session thực sự là chính xác.

### Khởi tạo Session Manager

Nói chung, bạn sẽ luôn luôn muốn khởi tạo session manager và đảm bảo rằng bạn đã khởi tạo nó cuối cùng, điều này đặt ra một giải pháp đơn giản để ngăn chặn so với phiên cố định. Thông thường bạn sẽ có khả năng thiết lập cấu hình và sau đó bên trong mô-đun ứng dụng bạn tái khởi động session manager.

Ngoài ra bạn có thể sẽ muốn cung cấp xác nhận để ngăn chặn việc cướp quyền.

* Sau đây minh họa cách bạn có thể cấu hình session manager bằng cách thiết lập tùy chọn trong cấu hình **local** or **global** của bạn:

**return array**(

’session’ => **array**(

’config’ => **array**(

’class’ => ’Zend\Session\Config\SessionConfig’,

’options’ => **array**(

’name’ => ’myapp’,

),

),

’storage’ => ’Zend\Session\Storage\SessionArrayStorage’,

’validators’ => **array**(

’Zend\Session\Validator\RemoteAddr’,

’Zend\Session\Validator\HttpUserAgent’,

),

),

);

Minh họa cách bạn có thể sử dụng các cấu hình ở trên để tạo ra các session manager :

**use** Zend\Session\SessionManager;

**use** Zend\Session\Container;

**class** **Module**

{

**public** **function** onBootstrap($e)

{

$eventManager = $e->getApplication()->getEventManager();

$serviceManager = $e->getApplication()->getServiceManager();

$moduleRouteListener = **new** ModuleRouteListener();

$moduleRouteListener->attach($eventManager);

$this->bootstrapSession($e);

}

**public** **function** bootstrapSession($e)

{

$session = $e->getApplication()

->getServiceManager()

->get('Zend\Session\SessionManager');

$session->start();

$container = **new** Container('initialized');

**if** (!isset($container->init)) {

$session->regenerateId(**true**);

$container->init = 1;

}

}

**public** **function** getServiceConfig()

{

**return** **array**(

'factories' => **array**(

'Zend\Session\SessionManager' => **function** ($sm) {

$config = $sm->get('config');

**if** (isset($config['session'])) {

$session = $config['session'];

$sessionConfig = **null**;

**if** (isset($session['config'])) {

$class = isset($session['config']['class']) ? $session['config']['class'] : 'Zend\Session\Config\SessionConfig';

$options = isset($session['config']['options']) ? $session['config']['options'] : **array**();

$sessionConfig = **new** $class();

$sessionConfig->setOptions($options);

}

$sessionStorage = **null**;

**if** (isset($session['storage'])) {

$class = $session['storage'];

$sessionStorage = **new** $class();

}

$sessionSaveHandler = **null**;

**if** (isset($session['save\_handler'])) {

*// class should be fetched from service manager since it will require constructor arguments*

$sessionSaveHandler = $sm->get($session['save\_handler']);

}

$sessionManager = **new** SessionManager($sessionConfig, $sessionStorage, $sessionSaveHandler);

**if** (isset($session['validator'])) {

$chain = $sessionManager->getValidatorChain();

**foreach** ($session['validator'] **as** $validator) {

$validator = **new** $validator();

$chain->attach('session.validate', **array**($validator, 'isValid'));

}

}

} **else** {

$sessionManager = **new** SessionManager();

}

Container::setDefaultManager($sessionManager);

**return** $sessionManager;

},

),

)

}

}

* Khi bạn tạo mới một Zend\Session\container (xem trang container session) trong một bộ điều khiển ví dụ, nó sẽ sử dụng phiên cấu hình ở trên.

## Session Save Handlers

* Zend Framework đi kèm với một bộ tiêu chuẩn của các lớp save handlers đã sẵn sàng để bạn sử dụng. Save Handlers bản thân được tách riêng từ hàm save handlers của PHP và chỉ thực hiện như một save handler PHP khi sử dụng kết hợp với Zend\Session\SessionManager.

### Bộ nhớ đệm

Zend\Session\SaveHandler\cache cho phép bạn cung cấp một instance của Zend\Cache để được sử dụng như là một phiên save handler. Nói chung nếu bạn đang sử dụng Cache save handler, bạn có khả năng sử dụng các sản phẩm như memcached.

### Cách sử dụng cơ bản

|  |
| --- |
| **use** Zend\Cache\StorageFactory;  **use** Zend\Session\SaveHandler\Cache;  **use** Zend\Session\SessionManager;  $cache = StorageFactory::factory(**array**(  'name' => 'memcached',  'options' => **array**(  'server' => '127.0.0.1',  ),  ));  $saveHandler = **new** Cache($cache);  $manager = **new** SessionManager();  $manager->setSaveHandler($saveHandler); |

## Session Storage

Zend Framework đi kèm với một bộ tiêu chuẩn của các lớp lưu trữ đã sẵn sàng để bạn sử dụng. Xử lý lưu trữ là trung gian giữa khi bắt đầu session và khi session ghi và đóng. Lưu trữ session mặc định là Zend\Session\Storage\SessionArrayStorage.

### Array Storage

Zend\Session\Storage\ArrayStorage cung cấp một cơ sở để lưu trữ tất cả thông tin trong một ArrayObject. Phương pháp lưu trữ này có thể không tương thích với thư viện của bên thứ 3 và tất cả tài sản sẽ không thể truy cập thông qua các $ \_SESSION properties. Ngoài ra ArrayStorage sẽ không tự động phục hồi lại storage container trong các trường hợp của mỗi yêu cầu mới và sẽ phải tự re-populated.

**Cách sử dụng cơ bản**

**use** Zend\Session\Storage\ArrayStorage;

**use** Zend\Session\SessionManager;

$populateStorage = **array**('foo' => 'bar');

$storage = **new** ArrayStorage($populateStorage);

$manager = **new** SessionManager();

$manager->setStorage($storage);

### Session Storage

Zend\Session\Storage\SessionStorage thay thế $ \_SESSION cung cấp một cơ sở để lưu trữ tất cả thông tin trong một ArrayObject. Điều này có nghĩa rằng nó có thể không tương thích với thư viện của bên thứ 3. Mặc dù thông tin được lưu trữ trong $ \_SESSION superglobal nên có sẵn trong phạm vi khác.

**Cách sử dụng cơ bản**

**use** Zend\Session\Storage\SessionStorage;

**use** Zend\Session\SessionManager;

$manager = **new** SessionManager();

$manager->setStorage(**new** SessionStorage());

### Session Array Storage

Zend\Session\Storage\SessionArrayStorage cung cấp một cơ sở để lưu trữ tất cả các thông tin trực tiếp trong $ \_SESSION superglobal.

Lớp lưu trữ này cung cấp khả năng tương thích hầu hết với các thư viện của bên thứ 3 và cho phép lưu trữ thông tin trực tiếp vào $ \_SESSION.

**Cách sử dụng cơ bản**

**use** Zend\Session\Storage\SessionArrayStorage;

**use** Zend\Session\SessionManager;

$manager = **new** SessionManager();

$manager->setStorage(**new** SessionArrayStorage());

# Cart

**Nội dung: Cart**

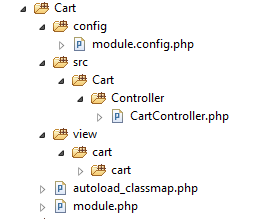
* **Module Cart**
* **CartController**

Một trong những điều đầu tiên mà phải được thiết kế trong khi thiết lập một cửa hàng trực tuyến là giỏ mua hàng. Các giỏ mua hàng lý tưởng nên cho phép người dùng cuối để chọn và thêm nhiều sản phẩm vào giỏ hàng và có thể kiểm tra từ các trang web.

## Quy trình kiểm tra được trình bày như sau:

* Khách hàng truy cập vào trang danh sách sản phẩm.
* Khách hàng lựa chọn một sản phẩm; anh/cô ấy được đưa đến trang chi tiết sản phẩm.
* Khách hàng sau đó chọn mua sản phẩm; khách hàng dự kiến ​​sẽ thêm số lượng mong muốn vào giỏ hàng.
* Khách hàng được chuyển hướng đến các trang giỏ mua hàng; vào đây khách hàng có thể thực hiện bất kỳ thay đổi thứ tự nếu cần thiết.
* Khách hàng lựa chọn phương thức thanh toán và nhập thông tin thanh toán.
* Nếu thành công, khách hàng được trình bày với một tùy chọn để cập nhật các thông tin chi tiết vận chuyển.
* Khách hàng sau đó xác nhận đơn đặt hàng.

## Xây dựng Module Cart



## Xây dựng Controller

### Phương thức thêm mặt hàng vào giỏ hàng

**public function** insertAction()

{

$id=$this->params()->fromRoute('id');

$sl=$this->params()->fromRoute('sl');

$dg=$this->params()->fromRoute('dg');

$config = **new** StandardConfig();

$config->setRememberMeSeconds(1800);

$config->setName('cartzf2');

$manager = **new** SessionManager($config);

Container::*setDefaultManager*($manager);

$cart=**new** Container('cart');

**if**($cart->offsetExists("sanpham"))

{

**if**(**isset**($cart->sanpham[$id]))

{

$giohang->tongtien-=$giohang->sanpham[$id]\*$dg;

$giohang->soluong-=$giohang->sanpham[$id];

}

$giohang->sanpham[$id]=$sl;

$giohang->tongtien+=$sl\*$dg;

$giohang->soluong+=$sl;

}

**else**

{

$cart->sanpham=**array**($id=>$sl);

$cart->tongtien=$sl\*$dg;

$cart->soluong=$sl;

}

}

### Phương thức cập nhật Giỏ hàng

**public function** update($id,$so\_luong,$don\_gia)

{

$cart=**new** Container('cart');

**if**(**isset**($cart->sanpham[$id]))

{

**if**($cart->sanpham[$id]!=$so\_luong)

{

$cart->tongtien-=$giohang->product[$id]\*$don\_gia;

$cart->soluong-=$giohang->sanpham[$id];

$cart->sanpham[$id]=$so\_luong;

$cart->tongtien+=$so\_luong\*$don\_gia;

$cart->soluong+=$sl;

}

}

}

### Phương thức xóa một mặt hàng trong Giỏ hàng

**public function** deletesanpham($id,$don\_gia)

{

$cart=**new** Container('cart');

$cart->tongtien-=$cart->sanpham[$id]\*$don\_gia;

$cart->soluong-=$cart->sanpham[$id];

$cart\_new=**array**();

**foreach**($cart->sanpham **as** $key=>$value)

{

**if**($key!=$id)

{

$cart\_new[$key]=$value;

}

}

**if**($cart\_new)

$cart->sanpham=$cart\_new;

**else**

$cart->getManager()->getStorage()->clear();

**return false**;

}

### Phương thức xóa Giỏ hàng

**public function** clearAction()

{

$cart=**new** Container('cart');

**if**($cart->offsetExists("sanpham"))

{

$cart->getManager()->getStorage()->clear();

}

}

# Authentication

**Nội dung:** Authentication

* Giới thiệu
* Zend\Authentication\Adapter
* Zend\Authentication\AuthenticationService
* Zend\Authentication\Adapter\DbTable

## Authentication

Zend\Authentication thành phần cung cấp một API để xác thực và bao gồm bộ điều hợp xác thực cụ thể cho các kịch bản trường hợp sử dụng thông thường.

Theo mặc định, **Zend\Authentication** cung cấp lưu trữ liên tục về danh tính từ một nỗ lực xác thực thành công bằng cách sử dụng PHP session.

Khi một nỗ lực xác thực thành công, **Zend\Authentication\AuthenticationService::authenticate()** lưu trữ các danh tính từ kết quả xác thực vào lưu trữ liên tục. Trừ khi có quy định khác,

Zend\Authentication adapter trả về một thể hiện của Zend\Authentication\Result với authenticate() để đại diện cho các kết quả của một nỗ lực xác thực. Adapters chuyển đến đối tượng Zend\Authentication\Result khi xây dựng, sau đây là bốn phương thức cung cấp một thiết lập cơ bản của hoạt động người dùng thực hiện được phổ biến đến các kết quả của Zend\Authentication adapters:

**isValid**() - trả về TRUE khi và chỉ khi kết quả đại diện cho một nỗ lực xác thực thành công

**getCode**() - trả về một Zend\Authentication\Result nhận dạng liên tục để xác định loại xác thực thất bại hay thành công đã xảy ra.

**getIdentity**() - trả về danh tính của các nỗ lực xác thực

**getMessages**() - trả về một mảng các thông điệp liên quan đến một nỗ lực xác thực không thành công

## Zend\Authentication\Adapter\DbTable

**Zend\Authentication\Adapter\DbTable** cung cấp khả năng xác thực đối với thông tin được lưu trữ trong một bảng cơ sở dữ liệu. Vì **Zend\Authentication\Adapter\DbTable** yêu cầu một thể hiện của **Zend\Db\Adapter\Adapter** thông qua để xây dựng nó, mỗi trường hợp ràng buộc với một cơ sở dữ liệu riêng kết nối. Tùy chọn cấu hình khác có thể được thiết lập thông qua constructor và thông qua phương thức, cho mỗi tuỳ chọn.

### Các tùy chọn cấu hình có sẵn bao gồm:

**tableName**: Đây là tên của bảng cơ sở dữ liệu có chứa các thông tin xác thực, và dựa vào đó các truy vấn cơ sở dữ liệu xác thực được thực hiện.

**identityColumn**: Đây là tên cột của bảng cơ sở dữ liệu được sử dụng để đại diện cho danh tính. Cột Identity phải chứa các giá trị duy nhất, chẳng hạn như tên người dùng hoặc địa chỉ e-mail.

**credentialColumn**: Đây là tên cột của bảng cơ sở dữ liệu được sử dụng để đại diện cho credential.

### Cách thực hiện

use Zend\Authentication\Adapter\DbTable;

$dbTable=new DbTable($dbAdapter);

//or $dbTable=new DbTable($dbAdapter,'user','username','password');

### Một số phương thức

$dbTable->setTableName('user'); //Bảng chứa thông tin cần xác thực

$dbTable->setIdentityColumn('username'); //Cột xác thực

$dbTable->setCredentialColumn('password'); //Cột xác thực mật khẩu

$dbTable->setIdentity('admin122'); //Giá trị đem xác thực

$dbTable->setCredential('admin'); //Giá trị mật khẩu xác thực

* Ngoài sự sẵn có của các phương pháp getIdentity() khi kết quả xác thực đối tượng, Zend\Authentication\Adapter\DbTable cũng hỗ trợ lấy các dòng của bảng khi xác thực thành công.

**echo** $authAdapter ->getIdentity() . "**\n\n**";

**print\_r**($authAdapter->getResultRowObject());

# MVCEVENT

Lớp MVC của Zend Framework 2 kết hợp và sử dụng tùy chỉnh Zend\EventManager\Event thực hiện - Zend\Mvc\MvcEvent. Sự kiện này được tạo ra trong Zend\MVC\Application :: bootstrap() và được truyền trực tiếp đến tất cả các sự kiện phương thức triggers().

## MvcEvent thêm bộ truy xuất và sự biến đổi sau đây.

* Đối tượng Application.
* Đối tượng Request.
* Đối tượng Response.
* Đối tượng Router.
* Đối tượng RouteMatch.
* Result - Thông thường là Kết quả của việc gửi đi một bộ điều khiển.
* Đối tượng ViewModel, điển hình việc đại diện cho cách trình bày view model.

## Các phương thức định nghĩa nó là:

setApplication($application)

getApplication()

setRequest($request)

getRequest()

setResponse($response)

getResponse()

setRouter($router)

getRouter()

setRouteMatch($routeMatch)

getRouteMatch()

setResult($result)

getResult()

setViewModel($viewModel)

getViewModel()

isError()

setError()

getError()

getController()

setController($name)

getControllerClass()

setControllerClass($class)

Các Application, Request, Response, Router, và ViewModel có tất cả chèn vào trong sự kiện bootstrap. Sau sự kiện route, nó sẽ được chèn vào với các đối tượng RouteMatch đóng gói kết quả.

Kể từ khi đối tượng này được thông qua xung quanh trên toàn MVC, nó là một vị trí chung để lấy kết quả của định tuyến, các bộ định tuyến, và các yêu cầu và đối tượng phản ứng. Ngoài ra, chúng tôi khuyến khích thiết lập các kết quả thực hiện trong sự kiện này, để cho phép lắng nghe sự kiện để xem xét và sử dụng chúng bên trong của họ. thực hiện Ví dụ, kết quả có thể được thông qua view renderer.

## Thứ tự của các sự kiện

Các sự kiện sau đây được kích hoạt, theo thứ tự sau:

|  |  |  |
| --- | --- | --- |
| Tên | Hằng | Diễn giải |
| bootstrap | MvcEvent::EVENT\_BOOTSTRAP | Bootstrap ứng dụng bằng cách tạo ra các ViewManager. |
| route | MvcEvent::EVENT\_ROUTE | Thực hiện tất cả các công việc tuyến đường (matching. ..). |
| dispatch | MvcEvent::EVENT\_DISPATCH | Gửi các tuyến đường phù hợp với một bộ controller/action. |
| dispatch.error | MvcEvent::EVENT\_DISPATCH\_ERROR | Sự kiện kích hoạt trong trường hợp của một vấn đề thực hiện không thành công (điều khiển không biết ...). |
| render | MvcEvent::EVENT\_RENDER | Chuẩn bị dữ liệu và sử dụng kết quả lớp view. |
| render.error | MvcEvent::EVENT\_RENDER\_ERROR | Sự kiện kích hoạt trong trường hợp của một vấn đề trong quá trình render (không renderer tìm thấy ...). |
| finish | MvcEvent::EVENT\_FINISH | Thực hiện bất kỳ nhiệm vụ một lần tất cả mọi thứ được thực hiện. |

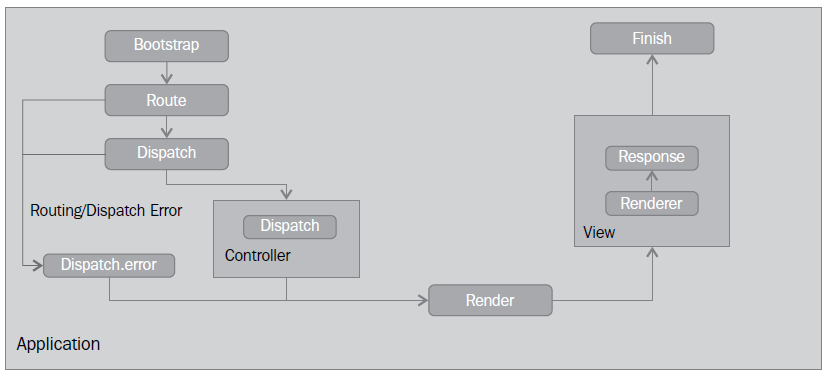
Zend Framework 2 là một framework hướng sự kiện; quản lý sự kiện cho phép bạn đính kèm sự kiện hầu như bất kỳ tính năng. Có ba điều khoản chính được sử dụng trong Zend Framework quản lý sự kiện, đó là như sau:

Event Manager: Các đối tượng EventManager là đối tượng chứa một tập hợp các lắng nghe và các sự kiện tương đối của chúng

Listener: Lắng nghe là gọi lại phản ứng với các sự kiện được kích hoạt

Event: Sự kiện này là hành động được kích hoạt bởi quản lý sự kiện

Quản lý sự kiện cung cấp attach() và trigger() để tạo ra và kích hoạt sự kiện tương ứng. Chủ yếu là chúng ta sẽ tùy thuộc vào các sự kiện MVC cho các hoạt động khác nhau, và trình tự thực hiện các sự kiện ứng dụng MVC được mô tả trong sơ đồ sau:



Ví dụ:

**public function** onBootstrap(MvcEvent $e)

{

$eventManager = $e->getApplication()->getEventManager();

$moduleRouteListener = **new** ModuleRouteListener();

$moduleRouteListener->attach($eventManager);

$e -> getApplication() -> getEventManager() -> attach('route', **array**($this, 'checkRoute'));

}

**public function** checkRoute(MvcEvent $e)

{

$route = $e -> getRouteMatch()-> getMatchedRouteName();

$routeMatch = $e->getRouteMatch();

$action = $routeMatch->getParam('action');

//echo $action; exit();

$kt=0;

**if**($route=='customers' && array\_search($action,**array**('view','edit','delete','bill','index'))!==**false**)

$kt=1;

**else**

**if**($route=='san-pham' && array\_search($action,**array**('add','edit','delete','view','test'))!==**false**)

$kt=1;

**else**

**if**($route=='article' && array\_search($action,**array**('add','delete','edit','view'))!==**false**)

$kt=1;

**else**

**if**($route=='user' && array\_search($action,**array**('add','delete','edit','view'))!==**false**)

$kt=1;

…

}

# ZEND\PERMISSIONS\ACL

**Nội dung:** ACL

* **Giới thiệu**
* **Resource**
* **Role**
* **Privilege**
* **Cấp quyền**
* **Kiểm tra quyền**

## Giới thiệu

* Zend\Permissions\Acl (Access control list ) là thành phần cung cấp danh sách quyền truy cập tài nguyên của ứng dụng, trong đó:
* resource: là nguồn tài nguyên của ứng dụng
* role: là nhóm quyền truy cập tài nguyên của ứng dụng
* privilege: quyền truy cập (chẳng hạn như đọc, chỉnh sửa dữ liệu nào đó)

**use** Zend\Permissions\Acl\Acl ;

$acl = **new** Acl () ;

## Resource

* Zend\Permissions\Acl\Resource\ResourceInterface resource là interface hỗ trợ việc tạo nguồn tài nguyên mà người dùng có thể truy cập thông qua quyền được cấp.
* Zend\Permissions\Acl\Resource\GenericResource cung cấp nền tảng cơ bản trong việc định nghĩa nguồn tài nguyên của ứng dụng

**use** Zend\Permissions\Acl\Acl ;

**use** Zend\Permissions\Acl\Resource\GenericResource **as** Resource ;   
$acl = **new** Acl ();

*// module product*

$acl -> addResource ( **new** Resource (‘product ' ));

*// module user*

$acl -> addResource ( **new** Resource ( ‘user ' ));

## Role

* Zend\Permissions\Acl\Role\RoleInterface là interface hỗ trợ việc tạo ra nhóm quyền truy cập tài nguyên của ứng dụng.
* Zend\Permissions\Acl\Role\GenericRole cung cấp nền tảng cơ bản trong việc định nghĩa các nhóm quyền truy cập nguồn tài nguyên của ứng dụng

**use** Zend\Permissions\Acl\Acl ;

**use** Zend\Permissions\Acl\Role\GenericRole **as** Role ;

$acl = **new** Acl ();

*// role guest*

$acl -> addRole (**new** Role ( 'guest' ));

*// role member kế thừa role guest*

$acl -> addRole ( **new** Role ( ‘member' ), 'guest' );

Chú ý: Khi cấp quyền, thì do role member kế thừa từ role guest nên mặc định role member sẽ có tất cả các quyền được cấp cho role guest

## Privilege

* Quyền hạn cụ thể khi truy cập tài nguyên của ứng dụng, chẳng hạn:

$guestPrivileges = array('view');

$memberPrivileges = array('add', 'edit', ‘view');

$adminPrivileges = array('add', 'edit’, ’delete’, ‘view');

**Cấp quyền**

* Cho phép người dùng truy cập tài nguyên

Cú pháp:

$acl->allow(role, resource, privilege=null)

* Không cho phép người dùng truy cập tài nguyên

Cú pháp:

$acl->deny(role, resource, privilege=null)

* Chú ý: Mặc định các nguồn tài nguyên ở trạng thái không cho phép truy cập

**use** Zend\Permissions\Acl\Acl ;

**use** Zend\Permissions\Acl\Role\GenericRole **as** Role ;

**use** Zend\Permissions\Acl\Resource\GenericResource **as** Resource ;

$acl = **new** Acl ();

*// module product*

$acl -> addResource ( **new** Resource (‘product ' ));

*// module user*

$acl -> addResource ( **new** Resource ( ’user' ));

*// role guest*

$acl -> addRole (**new** Role ( 'guest' ));

*// role member kế thừa role guest*

$acl -> addRole ( **new** Role ( ‘member' ), 'guest' );

*// Cấp quyến: cho phép role guest được sử dụng module user*

$acl -> allow('guest' ,’user’);

*// Cấp quyến: không cho phép role member được sử dụng action delete trong controller product*

$acl -> deny(‘member' ,’product’,’product:delete’);

## Kiểm tra quyền

* Để kiểm tra người dùng có được phép truy cập 1 tài nguyên trong ứng dụng hay không

Cú pháp:

$acl->isAllowed(role, resource, privilege)

Nếu phương thức trả về true: được phép truy cập

Nếu phương thức trả về false: không được phép truy cập

**use** Zend\Permissions\Acl\Acl ;

**use** Zend\Permissions\Acl\Role\GenericRole **as** Role ;

**use** Zend\Permissions\Acl\Resource\GenericResource **as** Resource ;

$acl = **new** Acl ();

*// Khai báo như ví dụ cấp quyền*

…

*// Kiểm tra quyền*

echo $acl -> isAllowed(‘guest' ,’user’); *-> true*

echo $acl -> isAllowed(‘member' ,’product’,’product:delete’); *-> false*

# GIỚI THIỆU VỀ Zend\JSON

**Nội dung: Zend\JSON**

* Sử dụng cơ bản
* Sử dụng nâng cao zend\json
* Đối tượng Encoding PHP

Zend \ JSON cung cấp phương pháp thuận tiện cho tuần tự hóa gốc PHP để JSON và giải mã JSON đến gốc PHP. Để biết thêm thông tin về JSON, hãy truy cập trang web của dự án JSON.

JSON, JavaScript Object Notation, có thể được sử dụng cho dữ liệu trao đổi giữa JavaScript và các ngôn ngữ khác. Kể từ khi JSON có thể được đánh giá trực tiếp bằng JavaScript, nó là một định dạng hiệu quả hơn và nhẹ hơn so với XML để trao đổi dữ liệu với khách hàng JavaScript.

Ngoài ra, Zend\JSON cung cấp một cách hữu ích để chuyển đổi bất kỳ định dạng chuỗi XML tùy ý vào một chuỗi JSON định dạng. Xây dựng tính năng này sẽ cho phép các nhà phát triển PHP để chuyển đổi dữ liệu doanh nghiệp được mã hóa ở định dạng XML sang định dạng JSON trước khi gửi nó cho các ứng dụng máy khách Ajax dựa trên trình duyệt. Nó cung cấp một cách dễ dàng để làm chuyển đổi dữ liệu động trên mã phía máy chủ do đó tránh không cần thiết phân tích cú pháp XML trong các ứng dụng bên trình duyệt. Nó cung cấp chức năng tiện ích tốt đẹp mà kết quả dễ dàng ứng dụng cụ thể xử lý dữ liệu kỹ thuật.

## SỬ DỤNG CƠ BẢN

Sử dụng Zend \ JSON liên quan đến việc sử dụng hai phương thức tĩnh chung có sẵn: Zend\JSON\JSON::encode()và Zend\JSON\JSON :: decode().

Ví dụ:

$data = **array**('hoten'=>'Tran Thi Thao','ngaysinh'=>'1997-1-1','noisinh'=>'Ho Chi Minh');

$endata = \Zend\Json\Json::*encode*($data,**TRUE**);

**echo** $endata;

//{"hoten":"Tran Thi Thao","ngaysinh":"1997-1-1","noisinh":"Ho Chi Minh"}

$dedata = \Zend\Json\Json::*decode*($endata,**TRUE**,**array**('encoding'=>'utf8'));

print\_r($dedata);

//Array ([hoten] => Tran Thi Thao [ngaysinh] => 1997-1-1 [noisinh] => Ho Chi Minh )

Đối số tùy chọn thứ hai của Zend \ JSON \ JSON :: prettyPrint () là một mảng tùy chọn. Tùy chọn thụt lề cho phép thiết lập chuỗi thụt đầu dòng - theo mặc định đó là một ký tự tab duy nhất.

## SỬ DỤNG NÂNG CAO Zend\JSON

Đối tượng Json

Khi mã hóa các đối tượng PHP như JSON, tất cả các thuộc tính chung của đối tượng đó sẽ được mã hóa trong một đối tượng JSON.

JSON không cho phép tham chiếu đối tượng, vì vậy cần phải cẩn thận không để mã hóa các đối tượng với tham chiếu đệ quy. Nếu bạn có vấn đề với đệ quy, Zend\JSON\JSON::encode ) và Zend JSON\Encoder::encode() cho phép tùy chọn tham số thứ hai để kiểm tra đệ quy; nếu một đối tượng được tuần tự hóa hai lần, một ngoại lệ sẽ được ném ra.

Đối tượng Decoding JSON đặt ra một khó khăn thêm, Tuy nhiên, các đối tượng JavaScript tương ứng chặt chẽ nhất để mảng kết hợp của PHP. Một số người cho rằng một định danh lớp nên được thông qua, và một trường hợp đối tượng của lớp đó phải được tạo ra và đặt các cặp key/value của đối tượng JSON; những người khác cảm thấy điều này có thể gây ra một nguy cơ bảo mật đáng kể. Theo mặc định, Zend\JSON\JSON sẽ giải mã các đối tượng JSON như mảng kết hợp. Tuy nhiên, nếu bạn mong muốn một đối tượng quay trở lại, bạn có thể xác định điều này:

// Decode JSON objects as PHP objects

$phpNative = Zend\Json\Json::decode($encodedValue, Zend\Json\Json::TYPE\_OBJECT);

Bất kỳ đối tượng do đó giải mã được trả lại như stdClass các đối tượng với các thuộc tính tương ứng với các cặp key/value trong các ký hiệu JSON.

Đề nghị của Zend Framework là nhà phát triển cá nhân nên quyết định làm thế nào để giải mã các đối tượng JSON. Nếu một đối tượng của một loại quy định phải được tạo ra, nó có thể được tạo ra trong mã phát triển và cư trú với các giá trị giải mã sử dụng Zend\JSON.

## Đối tượng Encoding PHP

Nếu bạn được mã hóa đối tượng PHP theo mặc định cơ chế mã hóa chỉ có thể truy cập thuộc tính public của các đối tượng. Khi phương thức toJson() được thực hiện trên một đối tượng để mã hóa, Zend\JSON\JSON gọi phương thức này và hy vọng đối tượng trả về một đại diện JSON của trạng thái bên trong.

Zend \ JSON \ JSON có thể mã hóa các đối tượng PHP đệ quy nhưng không làm như vậy theo mặc định. Điều này có thể được kích hoạt bằng cách thực sự như một tham số thứ hai tới Zend\JSON\JSON::encode().

// Encode PHP object recursively

$jsonObject = Zend\Json\Json::encode($data, **true**);

Khi thực hiện mã hóa đệ quy của các đối tượng, Khi thực hiện mã hóa đệ quy của các đối tượng, như JSON không hỗ trợ chu kỳ, một Zend\Json\Exception\RecursionException sẽ được ném ra. Nếu bạn muốn, bạn có thể tắt các trường hợp ngoại lệ bằng cách lựa chọn silenceCyclicalExceptions:

$jsonObject = Zend\Json\Json::encode($data,**true**,

**array**(’silenceCyclicalExceptions’ => **true**));

## Internal Encoder/Decoder

Zend \ JSON có hai chế độ khác nhau tùy thuộc nếu ext / json được kích hoạt trong cài đặt PHP của bạn hay không. Nếu ext/json cài đặt mặc định chức năng json\_encode() và json\_decode() được sử dụng để mã hóa và giải mã JSON. Nếu ext/JSON không cài đặt Zend Framework thực hiện đầy đủ trong mã PHP được sử dụng cho en-/decoding. Điều này là chậm hơn đáng kể so với sử dụng phần mở rộng PHP, nhưng cư xử chính xác như nhau.

Vẫn đôi khi bạn có thể muốn sử dụng encoder/decoder nội bộ ngay cả khi bạn có ext/json cài đặt. Bạn có thể đạt được điều này bằng cách gọi:

Zend\Json\Json::$useBuiltinEncoderDecoder = **true**;

### JSON Expressions

JavaScript làm cho sử dụng nặng nề của chức năng callbacks ẩn danh, có thể được lưu trong biến đối tượng JSON. Họ vẫn chỉ làm việc nếu không trả lại bên trong dấu ngoặc kép, mà Zend\JSON tự nhiên không. Với sự hỗ trợ biểu thức của Zend\JSON hỗ trợ bạn có thể mã hóa các đối tượng JSON với JavaScript callback hợp lệ. Công việc này cho cả hai json\_encode() hoặc bộ mã hóa nội bộ.

Một cuộc gọi lại JavaScript được đại diện bằng cách sử dụng đối tượng Zend\JSON\expr. Nó thực hiện các mô hình đối tượng giá trị là không thay đổi. Bạn có thể đặt các biểu hiện JavaScript như là đối số nhà xây dựng đầu tiên. Theo mặc định Zend\JSON\JSON::encode không mã hóa JavaScript gọi lại, bạn phải thông qua các tùy chọn enableJsonExprFinder và đặt nó là TRUE vào chức năng mã hóa. Nếu được kích hoạt các công việc hỗ trợ biểu thức cho tất cả các biểu thức lồng nhau trong cấu trúc đối tượng lớn.

Ví dụ:

$data = **array**(

'onClick' => **new** \Zend\Json\Expr('function(){window.alert(

"I am a valid JavaScript callback created by Zend\Json"

);

}'

),

'orther' => 'no expression',

);

$jsonObjectWithExpression = \Zend\Json\Json::*encode*($data,**false**,**array**('enableJsonExprFinder' => **true**));

### XML TO JSON CONVERSION

Zend \ JSON cung cấp một phương pháp thuận tiện cho việc chuyển đổi dữ liệu XML định dạng sang định dạng JSON. Tính năng này được lấy cảm hứng từ một bài báo của IBM developerWorks.

Zend\JSON bao gồm một chức năng tĩnh gọi là Zend\JSON\JSON::fromXml().

Chức năng này sẽ tạo ra JSON từ một đầu vào XML. Chức năng này sẽ lấy bất kỳ chuỗi XML tùy ý như là một tham số đầu vào. Nó cũng lấy ra tùy chọn boolean tham số đầu vào để hướng dẫn chuyển đổi logic bỏ qua hay không bỏ qua thuộc tính XML trong quá trình xử lý. Nếu tham số đầu vào tùy chọn này không được đưa ra, thì hành vi mặc định là bỏ qua các thuộc tính XML.

Gọi chức năng này được thực hiện như hình dưới đây:

$jsonContents = Zend\Json\Json::fromXml($xmlStringContents, **true**);

Chức năng Zend\JSON\JSON::fromXml() thực hiện chuyển đổi các tham số đầu vào chuỗi định dạng XML và trả về đầu ra chuỗi JSON định dạng tương đương. Trong trường hợp có đầu vào lỗi định dạng XML hoặc logic chuyển đổi lỗi, chức năng này sẽ ném một ngoại lệ. Logic chuyển đổi cũng sử dụng kỹ thuật đệ quy để đi qua cây XML. Nó hỗ trợ đệ quy tối đa 25 cấp độ sâu. Ngoài chiều sâu, nó sẽ ném một Zend\Json\Exception. Có một số tập tin XML với mức độ phức tạp khác nhau được cung cấp trong thư mục kiểm tra của Zend Framework. Chúng có thể được sử dụng để kiểm tra các chức năng của tính năng xml2json.

Sau đây là một ví dụ đơn giản cho thấy cả hai chuỗi đầu vào XML thông qua và các chuỗi đầu ra JSON trở lại kết quả là từ chức năng Zend\JSON\JSON::fromXml().Ví dụ này sử dụng các tham số chức năng tùy chọn như không bỏ qua các thuộc tính XML trong khi chuyển đổi. Do đó, bạn có thể nhận thấy rằng các chuỗi kết quả JSON bao gồm một đại diện của các thuộc tính XML hiện trong chuỗi đầu vào XML.

Chuỗi đầu vào XML thông qua chức năng Zend\JSON\JSON::fromXml():

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<books>

<book id=*"1"*>

<title>Code Generation in Action</title>

<author>

<first>Jack</first>

<last>Herrington</last>

</author>

<publisher>Manning</publisher>

</book>

<book id=*"2"*>

<title>PHP Hacks</title>

<author>

<first>Jack</first>

<last>Herrington</last>

</author>

<publisher>O’Reilly</publisher>

</book>

<book id=*"3"*>

<title>Podcasting Hacks</title>

<author>

<first>Jack</first>

<last>Herrington</last>

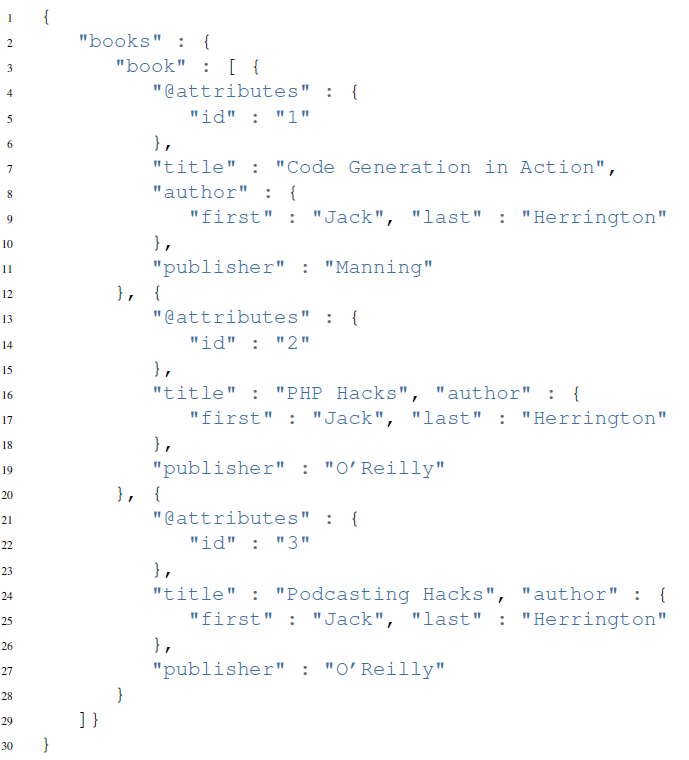
</author>

<publisher>O’Reilly</publisher>

</book>

</books>

Json xuất ra chuỗi kết quả từ chức Zend\Json\Json::fromXml()



# UNIT TESTING A ZF2

**Nội dung: Zend\JSON**

* Thiết lập các thư mục kiểm tra
* Khởi tạo các bài kiểm tra của bạn
* Controller thử nghiệm đầu tiên của bạn
* Test Model
* Test Model Tabel
* Test Controller

Một bộ kiểm tra đơn vị vững chắc là điều cần thiết cho sự phát triển liên tục trong các dự án lớn, đặc biệt là với nhiều người tham gia. Sẽ trở lại và tự kiểm tra tất cả các thành phần riêng lẻ của một ứng dụng sau mỗi lần thay đổi là không thực tế. Unit tests của bạn sẽ giúp làm giảm bớt là bằng cách tự động kiểm tra các thành phần ứng dụng của bạn và cảnh báo cho bạn khi một cái gì đó không hoạt động khi bạn đã viết các bài kiểm tra của bạn.

Hướng dẫn này được viết với hy vọng cho thấy cách để kiểm tra các phần khác nhau của một ứng dụng MVC Zend Framework 2. Như vậy, hướng dẫn này sẽ sử dụng các ứng dụng viết bằng sự bắt đầu. hướng dẫn người dùng. Nó là không có cách hướng dẫn để kiểm tra đơn vị nói chung, nhưng ở đây chỉ để giúp vượt qua những trở ngại ban đầu bằng văn bản Unit tests để ZF2 ứng dụng.

Nó được khuyến khích là có ít nhất một sự hiểu biết cơ bản của Unit tests, khẳng định và giả lập.

Như vậy sẽ hướng dẫn Zend Framework 2 API sử dụng PHPUnit. Hướng dẫn này giả định rằng bạn đã có PHPUnit cài đặt. Các phiên bản của PHPUnit sử dụng nên được 3.7. \*

## Thiết lập các thư mục kiểm tra

Như Zend Framework 2 ứng dụng được xây dựng từ các module cần được khối độc lập của một ứng dụng, chúng tôi không kiểm tra các ứng dụng trong toàn bộ của nó, nhưng module bằng module. Chúng tôi sẽ hiển thị như thế nào để thiết lập các yêu cầu tối thiểu để thử nghiệm một module các module Album chúng tôi đã viết trong hướng dẫn sử dụng, và sau đó có thể được sử dụng như một cơ sở để thử nghiệm bất kỳ thành phần khác.

zf2-tutorial/

/module

/Album

/test

/AlbumTest

/Controller

Cấu trúc của thư mục kiểm tra phù hợp chính xác với các tập tin nguồn của module và nó sẽ cho phép bạn để giữ cho các bài kiểm tra của bạn được tổ chức và dễ dàng để tìm thấy.

## Khởi tạo các bài kiểm tra của bạn

Tiếp theo, tạo ra một tập tin gọi là phpunit.xml dưới zf2-tutorial/module/Album/test:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<phpunit bootstrap=*"Bootstrap.php"* colors=*"true"*>

<testsuites>

<testsuite name=*"fashion203"*>

<directory>./UserTest</directory>

</testsuite>

</testsuites>

</phpunit>

Và một tập tin gọi là Bootstrap.php, cũng theo zf2-tutorial/module/Album/test:

<?php

**namespace** UserTest;

**use** Zend\Loader\AutoloaderFactory;

**use** Zend\Mvc\Service\ServiceManagerConfig;

**use** Zend\ServiceManager\ServiceManager;

**use** RuntimeException;

error\_reporting ( E\_ALL | E\_STRICT );

chdir ( \_\_DIR\_\_ );

/\*\*

Test bootstrap, for setting up autoloading

\*/

**class** Bootstrap {

**protected static** *$serviceManager*;

**public static function** *init*() {

$zf2ModulePaths = **array** (

dirname ( dirname ( \_\_DIR\_\_ ) )

);

**if** (($path = **static**::*findParentPath* ( 'vendor' ))) {

$zf2ModulePaths [] = $path;

}

**if** (($path = **static**::*findParentPath* ( 'module' )) !== $zf2ModulePaths [0]) {

$zf2ModulePaths [] = $path;

}

**static**::*initAutoloader* ();

// use ModuleManager to load this module and it's dependencies

$config = **array** (

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

'module\_paths' => $zf2ModulePaths

),

'modules' => **array** (

'User'

)

);

$serviceManager = **new** ServiceManager ( **new** ServiceManagerConfig () );

$serviceManager->setService ( 'ApplicationConfig', $config );

$serviceManager->get ( 'ModuleManager' )->loadModules ();

**static**::*$serviceManager* = $serviceManager;

}

**public static function** *chroot*() {

$rootPath = dirname ( **static**::*findParentPath* ( 'module' ) );

chdir ( $rootPath );

}

**public static function** *getServiceManager*() {

**return static**::*$serviceManager*;

}

**protected static function** *initAutoloader*() {

$vendorPath = **static**::*findParentPath* ( 'vendor' );

$zf2Path = getenv ( 'ZF2\_PATH' );

**if** (! $zf2Path) {

**if** (defined ( 'ZF2\_PATH' )) {

$zf2Path = ZF2\_PATH;

} **elseif** (is\_dir ( $vendorPath . '/ZF2/library' )) {

$zf2Path = $vendorPath . '/ZF2/library';

} **elseif** (is\_dir ( $vendorPath . '/zendframework/zendframework/library' )) {

$zf2Path = $vendorPath . '/zendframework/zendframework/library';

}

}

**if** (! $zf2Path) {

**throw new** RuntimeException ( 'Unable to load ZF2. Run ‘php composer.phar install‘ or' . ' define a ZF2\_PATH environment variable.' );

}

**if** (file\_exists ( $vendorPath . '/autoload.php' )) {

**include** $vendorPath . '/autoload.php';

}

**include** $zf2Path . '/Zend/Loader/AutoloaderFactory.php';

AutoloaderFactory::*factory* ( **array** (

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

'autoregister\_zf' => **true**,

'namespaces' => **array** (

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

)

)

) );

}

**protected static function** *findParentPath*($path) {

$dir = \_\_DIR\_\_;

$previousDir = '.';

**while** ( ! is\_dir ( $dir . '/' . $path ) ) {

$dir = dirname ( $dir );

**if** ($previousDir === $dir)

**return false**;

$previousDir = $dir;

}

**return** $dir . '/' . $path;

}

}

Bootstrap::*init* ();

Bootstrap::*chroot* ();

Các nội dung của tập tin bootstrap này có thể khó khăn ở cái nhìn đầu tiên, nhưng tất cả nó thực sự là đảm bảo rằng tất cả các file cần thiết là autoloadable của chúng tôi để thử nghiệm. Các dòng quan trọng nhất là dòng 38 trên mà chúng ta nói module nào, chúng tôi muốn tải cho thử nghiệm của chúng ta. Trong trường hợp này chúng tôi chỉ nạp module Album vì nó không phụ thuộc đối với các module khác. Bây giờ, nếu bạn điều hướng đến thư mục zf2-tutorial/module/Album/test/, và chạy phpunit, bạn sẽ nhận được một đầu ra tương tự như sau:

## Controller thử nghiệm đầu tiên của bạn

Thử nghiệm controller không bao giờ là nhiệm vụ dễ dàng, nhưng Zend Framework 2 đi kèm với các thành phần Zend\Test nên làm cho thử nghiệm ít cồng kềnh.

Đầu tiên, tạo IndexControllerTest.php dưới zf2-tutorial/module/Album/test/AlbumTest/Controller với các nội dung sau:

<?php

**namespace** AlbumTest\Controller;

**use** Zend\Test\PHPUnit\Controller\AbstractHttpControllerTestCase;

**class** AlbumControllerTest **extends** AbstractHttpControllerTestCase {

**public function** setUp() {

$this->setApplicationConfig ( **include** '/var/www/zf2tutorial/config/application.config.php' );

**parent**::*setUp* ();

}

}

Lớp AbstractHttpControllerTestCase chúng tôi mở rộng ở đây sẽ giúp chúng ta thiết lập các ứng dụng riêng của mình, giúp với điều phối và các nhiệm vụ khác xảy ra trong một yêu cầu, cũng cung cấp phương thức để khẳng định yêu cầu params, tiêu đề phản ứng, chuyển hướng và nhiều hơn nữa. Xem Zend\Test hướng dẫn kiểm tra để biết thêm. Một điều được cần thiết là để thiết lập cấu hình ứng dụng với phương thức setApplicationConfig.

Bây giờ, thêm các chức năng sau vào lớp AlbumControllerTest:

**public function** testIndexActionCanBeAccessed()

{

$this->dispatch('/album');

$this->assertResponseStatusCode(200);

$this->assertModuleName('Album');

$this->assertControllerName('Album\Controller\Album');

$this->assertControllerClass('AlbumController');

$this->assertMatchedRouteName('album');

}

Cấu hình quản lý dịch vụ cho các bài kiểm tra

## Test Model

Bây giờ chúng ta biết làm thế nào để kiểm tra bộ điều khiển của chúng tôi, chúng ta hãy di chuyển đến một phần quan trọng khác của ứng dụng - thực thể model.

Ở đây chúng ta muốn kiểm tra rằng trạng thái khởi đầu của thực thể là những gì chúng ta mong đợi nó được, chúng ta có thể chuyển đổi các tham số của model đến và đi từ một mảng, và nó có tất cả các bộ lọc đầu vào chúng ta cần.

Tạo tập tin AlbumTest.php trong /Album/test/AlbumTest/Model với nội dung sau:

<?php

**namespace** AlbumTest\Model;

**use** Album\Model\Album;

**use** PHPUnit\_Framework\_TestCase;

**class** AlbumTest **extends** PHPUnit\_Framework\_TestCase

{

**public function** testAlbumInitialState()

{

$album = **new** Album();

$this->assertNull(

$album->artist,

'"artist" should initially be null'

);

$this->assertNull(

$album->id,

'"id" should initially be null'

);

$this->assertNull(

$album->title,

'"title" should initially be null'

);

}

**public function** testExchangeArraySetsPropertiesCorrectly()

{

$album = **new** Album();

$data = **array**('artist' => 'some artist',

'id' => 123,

'title' => 'some title');

$album->exchangeArray($data);

$this->assertSame(

$data['artist'],

$album->artist,

'"artist" was not set correctly'

);

$this->assertSame(

$data['id'],

$album->id,

'"id" was not set correctly'

);

$this->assertSame(

$data['title'],

$album->title,

'"title" was not set correctly'

);

}

**public function** testExchangeArraySetsPropertiesToNullIfKeysAreNotPresent()

{

$album = **new** Album();

$album->exchangeArray(**array**('artist' => 'some artist',

'id' => 123,

'title' => 'some title'));

$album->exchangeArray(**array**());

$this->assertNull(

$album->artist, '"artist" should have defaulted to null'

);

$this->assertNull(

$album->id, '"id" should have defaulted to null'

);

$this->assertNull(

$album->title, '"title" should have defaulted to null'

);

}

**public function** testGetArrayCopyReturnsAnArrayWithPropertyValues()

{

$album = **new** Album();

$data = **array**('artist' => 'some artist',

'id' => 123,

'title' => 'some title');

$album->exchangeArray($data);

$copyArray = $album->getArrayCopy();

$this->assertSame(

$data['artist'],

$copyArray['artist'],

'"artist" was not set correctly'

);

$this->assertSame(

$data['id'],

$copyArray['id'],

'"id" was not set correctly'

);

$this->assertSame(

$data['title'],

$copyArray['title'],

'"title" was not set correctly'

);

}

**public function** testInputFiltersAreSetCorrectly()

{

$album = **new** Album();

$inputFilter = $album->getInputFilter();

$this->assertSame(3, $inputFilter->count());

$this->assertTrue($inputFilter->has('artist'));

$this->assertTrue($inputFilter->has('id'));

$this->assertTrue($inputFilter->has('title'));

}

}

**Chúng ta đang thử nghiệm cho 5 điều:**

Tất cả các thuộc tính của Album bước đầu thiết lập để NULL?

Thuộc tính của Album sẽ được thiết lập một cách chính xác khi chúng ta gọi exchangeArray ()?

Giá trị mặc định của NULL sẽ được sử dụng cho các thuộc tính có các khóa không có mặt trong mảng $data?

Chúng ta có thể nhận được một mảng sao chép của model chúng ta?

Tất cả phần tử có bộ lọc đầu vào hiện tại?

## Test Model table

Bước cuối cùng trong đơn vị unit testing hướng dẫn cho Zend Framework 2 ứng dụng là viết tests cho các bảng mode.

Kiểm tra này đảm bảo rằng chúng ta có thể có được một danh sách các album, hay một album bằng ID của nó, và chúng ta có thể lưu và xóa các album từ cơ sở dữ liệu.

Để tránh tương tác thực tế với cơ sở dữ liệu riêng của mình, chúng ta sẽ thay thế phần nào đó có mocks.

Tạo tập tin AlbumTableTest.php trong module/Album/test/AlbumTest/Model với nội dung sau:

<?php

**namespace** AlbumTest\Model;

**use** Album\Model\AlbumTable;

**use** Album\Model\Album;

**use** Zend\Db\ResultSet\ResultSet;

**use** PHPUnit\_Framework\_TestCase;

**class** AlbumTableTest **extends** PHPUnit\_Framework\_TestCase

{

**public function** testFetchAllReturnsAllAlbums()

{

$resultSet = **new** ResultSet();

$mockTableGateway = $this->getMock('Zend\Db\TableGateway\TableGateway'

,**array**('select'),**array**(),'', **false**);

$mockTableGateway ->expects($this->once())

->method('select')

->with()

->will($this->returnValue($resultSet));

$albumTable = **new** AlbumTable($mockTableGateway);

$this->assertSame($resultSet, $albumTable->fetchAll());

}

}

Vì chúng ta đang thử nghiệm AlbumTable tại đây và không phải là lớp TableGateway (mà đã được thử nghiệm trong Zend Framework), chúng tôi chỉ muốn chắc chắn rằng lớp AlbumTable chúng ta là tương tác với các lớp TableGateway cách mà chúng tôi mong đợi nó.

Ở trên, chúng tôi đang test để xem nếu fetchAl() của AlbumTable sẽ gọi Select() của thuộc tính $tableGateway không có tham số. Nếu có, nó sẽ trả về một đối tượng ResultSet. Cuối cùng, chúng tôi hy vọng rằng đối tượng ResultSet sẽ được trả lại cho phương thức gọi. Kiểm tra này sẽ chạy tốt, vì vậy bây giờ chúng ta có thể thêm các phần còn lại của phương thức test:

**public function** testCanRetrieveAnAlbumByItsId()

{

$album = **new** Album();

$album->exchangeArray(**array**('id' => 123,

'artist' => 'The Military Wives',

'title' => 'In My Dreams'));

$resultSet = **new** ResultSet();

$resultSet->setArrayObjectPrototype(**new** Album());

$resultSet->initialize(**array**($album));

$mockTableGateway = $this->getMock(

'Zend\Db\TableGateway\TableGateway',

**array**('select'),

**array**(),

'',

**false**

);

$mockTableGateway->expects($this->once())

->method('select')

->with(**array**('id' => 123))

->will($this->returnValue($resultSet));

$albumTable = **new** AlbumTable($mockTableGateway);

$this->assertSame($album, $albumTable->getAlbum(123));

}

**public function** testCanDeleteAnAlbumByItsId()

{

$mockTableGateway = $this->getMock(

'Zend\Db\TableGateway\TableGateway',

**array**('delete'),

**array**(),

'',

**false**

);

$mockTableGateway->expects($this->once())

->method('delete')

->with(**array**('id' => 123));

$albumTable = **new** AlbumTable($mockTableGateway);

$albumTable->deleteAlbum(123);

}

**public function** testSaveAlbumWillInsertNewAlbumsIfTheyDontAlreadyHaveAnId()

{

$albumData = **array**(

'artist' => 'The Military Wives',

'title' => 'In My Dreams'

);

$album = **new** Album();

$album->exchangeArray($albumData);

$mockTableGateway = $this->getMock(

'Zend\Db\TableGateway\TableGateway',

**array**('insert'),

**array**(),

'',

**false**

);

$mockTableGateway->expects($this->once())

->method('insert')

->with($albumData);

$albumTable = **new** AlbumTable($mockTableGateway);

$albumTable->saveAlbum($album);

}

**public function** testSaveAlbumWillUpdateExistingAlbumsIfTheyAlreadyHaveAnId()

{

$albumData = **array**(

'id' => 123,

'artist' => 'The Military Wives',

'title' => 'In My Dreams',

);

$album = **new** Album();

$album->exchangeArray($albumData);

$resultSet = **new** ResultSet();

$resultSet->setArrayObjectPrototype(**new** Album());

$resultSet->initialize(**array**($album));

$mockTableGateway = $this->getMock(

'Zend\Db\TableGateway\TableGateway',

**array**('select', 'update'),

**array**(),

'',

**false**

);

$mockTableGateway->expects($this->once())

->method('select')

->with(**array**('id' => 123))

->will($this->returnValue($resultSet));

$mockTableGateway->expects($this->once())

->method('update')

->with(

**array**(

'artist' => 'The Military Wives',

'title' => 'In My Dreams'

),

**array**('id' => 123)

);

$albumTable = **new** AlbumTable($mockTableGateway);

$albumTable->saveAlbum($album);

}

**public function** testExceptionIsThrownWhenGettingNonExistentAlbum()

{

$resultSet = **new** ResultSet();

$resultSet->setArrayObjectPrototype(**new** Album());

$resultSet->initialize(**array**());

$mockTableGateway = $this->getMock(

'Zend\Db\TableGateway\TableGateway',

**array**('select'),

**array**(),

'',

**false**

);

$mockTableGateway->expects($this->once())

->method('select')

->with(**array**('id' => 123))

->will($this->returnValue($resultSet));

$albumTable = **new** AlbumTable($mockTableGateway);

**try** {

$albumTable->getAlbum(123);

}

**catch** (\Exception $e) {

$this->assertSame('Could not find row 123', $e->getMessage());

**return**;

}

$this->fail('Expected exception was not thrown');

}

**Tài liệu tham khảo**

* ZF2 của Zend framwork
* Zend Framework 2.0 by Example Beginner's Guide
* Using Zend Framework 2

Mục lục

[Chương 1: TỔNG QUAN ZEND FRAMEWORK 2 2](#_Toc408853941)

[1.1. Giới thiệu ZF 3](#_Toc408853942)

[1.2. Giới thiệu ZF2 4](#_Toc408853943)

[1.3. Kiến trúc MVC của ZF2 5](#_Toc408853944)

[1.4. Zend MVC 5](#_Toc408853945)

[1.5. Cài đặt 7](#_Toc408853946)

[Chương 2: Module 9](#_Toc408853947)

[2.1. Zend Module 10](#_Toc408853948)

[2.2. Cấu trúc của module có dạng như sau: 11](#_Toc408853949)

[2.3. Namespaces 13](#_Toc408853950)

[2.4. Model-View-Controller (MVC layer) 14](#_Toc408853951)

[Chương 3: ZF2 module – configuration 16](#_Toc408853952)

[3.1. Cầu hình Module 17](#_Toc408853953)

[3.2. Routing (Định tuyến) 17](#_Toc408853954)

[3.2.1 SimpleRouteStack 18](#_Toc408853955)

[3.2.2 TreeRouteStack 18](#_Toc408853956)

[3.2.3 Các loại định tuyến 18](#_Toc408853957)

[Chương 4: Database và Model 22](#_Toc408853958)

[4.1. ZEND\DB\ADAPTER 23](#_Toc408853959)

[4.1.1 Tạo Adapter 23](#_Toc408853960)

[4.1.2 Chuẩn bị câu lệnh truy vấn thông qua Zend\Db\Adapter\Adapter::query() 24](#_Toc408853961)

[4.1.3 Thực thi câu lệnh truy vấn thông qua Zend\Db\Adapter\Adapter::query() 25](#_Toc408853962)

[4.1.4 Tạo Statement 25](#_Toc408853963)

[4.2. ZEND\DB\SQL 25](#_Toc408853964)

[4.2.1 Zend\Db\Sql\Sql 25](#_Toc408853965)

[4.2.2 Sử dụng đối tượng Select 26](#_Toc408853966)

[4.3. ZEND\DB\TABLEGATEWAY 29](#_Toc408853967)

[4.3.1 Cấu hình Adapter 31](#_Toc408853968)

[4.3.2 Sử dụng ServiceManager để cấu hình table gateway 32](#_Toc408853969)

[4.3.3 Thực hiện Controller 33](#_Toc408853970)

[4.4. ZEND\DB\ROWGATEWAY 35](#_Toc408853971)

[Chương 5: FORM và ACTION 36](#_Toc408853972)

[5.1. Giới thiệu Zend/Form 37](#_Toc408853973)

[5.2. Xem xét form sau: 39](#_Toc408853974)

[5.3. Validation: InputFilter 41](#_Toc408853975)

[5.4. Quản lý lỗi 41](#_Toc408853976)

[5.4.1 Quản lý thông tin nhập 42](#_Toc408853977)

[Chương 6: File Transfer và Paginator 43](#_Toc408853978)

[6.1. File Transfer 44](#_Toc408853979)

[6.1.1 Đặt vấn đề: 44](#_Toc408853980)

[6.1.2 Để làm được việc này thì: 44](#_Toc408853981)

[6.1.3 Thực hiện: 44](#_Toc408853982)

[6.2. Paginator 45](#_Toc408853983)

[6.2.1 Giới thiệu Zend\Paginator 45](#_Toc408853984)

[6.2.2 Phân trang bộ dữ liệu 46](#_Toc408853985)

[6.2.3 Sử dụng Paginator để phân trang 48](#_Toc408853986)

[Chương 7: Captcha 50](#_Toc408853987)

[7.1. Giới thiệu: 51](#_Toc408853988)

[7.2. Captcha Operation: 51](#_Toc408853989)

[7.3. Captcha Adapter 52](#_Toc408853990)

[7.3.1 Các adapter dưới đây là các adapter mặc định của Zend Framework: 52](#_Toc408853991)

[7.3.2 Zend\Captcha\AbstractWord 52](#_Toc408853992)

[7.3.3 Zend\Captcha\Dump 52](#_Toc408853993)

[7.3.4 Zend\Captcha\Figlet 52](#_Toc408853994)

[7.3.5 Zend\Captcha\Image 52](#_Toc408853995)

[7.3.6 Zend\Captcha\Recaptcha 53](#_Toc408853996)

[7.4. Thực hiện Captcha 53](#_Toc408853997)

[7.4.1 Chuẩn bị: 53](#_Toc408853998)

[Chương 8: ZEND\CONFIG\READER và WRITER 58](#_Toc408853999)

[8.1. Zend\Config\Reader 59](#_Toc408854000)

[8.1.1 Zend\Config\Reader\Ini 59](#_Toc408854001)

[8.1.2 Zend\Config\Reader\Xml 59](#_Toc408854002)

[8.1.3 Zend\Config\Reader\Json 60](#_Toc408854003)

[8.1.4 Zend\Config\Reader\Yaml 61](#_Toc408854004)

[8.2. ZEND\CONFIG\WRITER 61](#_Toc408854005)

[8.2.1 Zend\Config\Writer\Ini 62](#_Toc408854006)

[8.2.2 Zend\Config\Writer\Xml 63](#_Toc408854007)

[8.2.3 Using Zend\Config\Writer\Json 64](#_Toc408854008)

[8.2.4 Zend\Config\Writer\Yaml 65](#_Toc408854009)

[Chương 9: Mail 66](#_Toc408854010)

[9.1. Giới thiệu 67](#_Toc408854011)

[9.2. Sử dụng mail transport 69](#_Toc408854012)

[9.2.1 Sendmail Transport 69](#_Toc408854013)

[9.2.2 SMTP Transport 69](#_Toc408854014)

[9.2.3 File Transport 69](#_Toc408854015)

[Chương 10: SESSION 71](#_Toc408854016)

[10.1. Session Config 72](#_Toc408854017)

[10.1.1 Standard Config 72](#_Toc408854018)

[10.2. Session Config 73](#_Toc408854019)

[10.2.1 Tùy chọn cấu hình cơ bản 73](#_Toc408854020)

[10.2.2 Cách sử dụng cơ bản 74](#_Toc408854021)

[10.3. SESSION CONTAINER 74](#_Toc408854022)

[10.3.1 Cách sử dụng cơ bản 74](#_Toc408854023)

[10.3.2 Thiết lập mặc định Session Manager 74](#_Toc408854024)

[10.4. SESSION MANAGER 74](#_Toc408854025)

[10.4.1 Khởi tạo Session Manager 75](#_Toc408854026)

[10.5. Session Save Handlers 77](#_Toc408854027)

[10.5.1 Bộ nhớ đệm 77](#_Toc408854028)

[10.5.2 Cách sử dụng cơ bản 77](#_Toc408854029)

[10.6. Session Storage 78](#_Toc408854030)

[10.6.1 Array Storage 78](#_Toc408854031)

[10.6.2 Session Storage 78](#_Toc408854032)

[10.6.3 Session Array Storage 78](#_Toc408854033)

[Chương 11: Cart 80](#_Toc408854034)

[11.1. Quy trình kiểm tra được trình bày như sau: 81](#_Toc408854035)

[11.2. Xây dựng Module Cart 81](#_Toc408854036)

[11.3. Xây dựng Controller 81](#_Toc408854037)

[11.3.1 Phương thức thêm mặt hàng vào giỏ hàng 81](#_Toc408854038)

[11.3.2 Phương thức cập nhật Giỏ hàng 82](#_Toc408854039)

[11.3.3 Phương thức xóa một mặt hàng trong Giỏ hàng 82](#_Toc408854040)

[11.3.4 Phương thức xóa Giỏ hàng 82](#_Toc408854041)

[Chương 12: Authentication 84](#_Toc408854042)

[12.1. Authentication 85](#_Toc408854043)

[12.2. Zend\Authentication\Adapter\DbTable 85](#_Toc408854044)

[12.2.1 Các tùy chọn cấu hình có sẵn bao gồm: 85](#_Toc408854045)

[12.2.2 Cách thực hiện 86](#_Toc408854046)

[12.2.3 Một số phương thức 86](#_Toc408854047)

[Chương 13: MVCEVENT 87](#_Toc408854048)

[13.1. MvcEvent thêm bộ truy xuất và sự biến đổi sau đây. 87](#_Toc408854049)

[13.2. Các phương thức định nghĩa nó là: 87](#_Toc408854050)

[13.3. Thứ tự của các sự kiện 87](#_Toc408854051)

[Chương 14: ZEND\PERMISSIONS\ACL 90](#_Toc408854052)

[14.1. Giới thiệu 91](#_Toc408854053)

[14.2. Resource 91](#_Toc408854054)

[14.3. Role 91](#_Toc408854055)

[14.4. Privilege 91](#_Toc408854056)

[14.5. Kiểm tra quyền 92](#_Toc408854057)

[Chương 15: GIỚI THIỆU VỀ Zend\JSON 93](#_Toc408854058)

[15.1. SỬ DỤNG CƠ BẢN 94](#_Toc408854059)

[15.2. SỬ DỤNG NÂNG CAO Zend\JSON 94](#_Toc408854060)

[15.3. Đối tượng Encoding PHP 95](#_Toc408854061)

[15.4. Internal Encoder/Decoder 95](#_Toc408854062)

[15.4.1 JSON Expressions 96](#_Toc408854063)

[15.4.2 XML TO JSON CONVERSION 96](#_Toc408854064)

[Chương 16: UNIT TESTING A ZF2 100](#_Toc408854065)

[16.1. Thiết lập các thư mục kiểm tra 101](#_Toc408854066)

[16.2. Khởi tạo các bài kiểm tra của bạn 101](#_Toc408854067)

[16.3. Controller thử nghiệm đầu tiên của bạn 104](#_Toc408854068)

[16.4. Test Model 105](#_Toc408854069)

[16.5. Test Model table 108](#_Toc408854070)