

# Bài 1: Cấu Trúc Folder Codeigniter

## 1. Download và Cài đặt

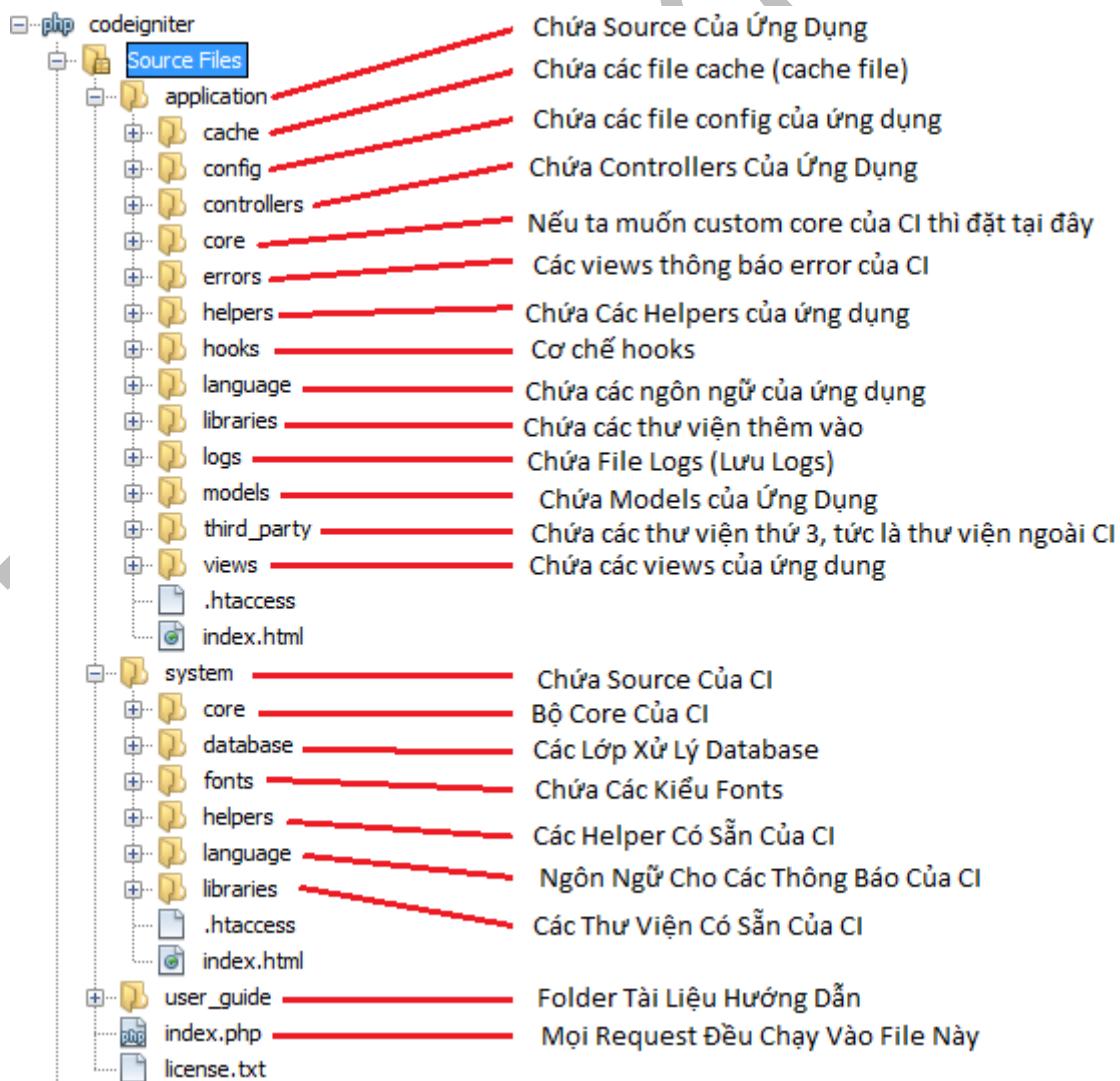
Việc **download và cài đặt Codeigniter** rất đơn giản. Trước tiên bạn vào trang <https://codeigniter.com/> để download source (chọn version hiện tại là version 3.1.7).

Sau khi download xong bạn sẽ có được 1 file nén .rar, bạn vào thư mục htdocs (đối với XAMPP) hoặc thư mục www (đối với Web Server), tạo một folder mới, sau đó giải nén file vừa download và copy vào folder mới tạo này.

Sau khi giải nén các bạn ra trình duyệt gõ localhost/codeigniter, nếu có kết quả xuất hiện thì bạn đã cài đặt thành công, ngược lại bạn cài đặt bị lỗi.

## 2. Cấu Trúc Folder Codeigniter

Sau khi bạn giải nén Codeigniter sẽ có các thư mục cấu trúc các folder codeigniter như sau:



Trong đó folder System là bộ core của CI, chúng ta không được dụng tới nó, chỉ được phép gọi ra và xài thôi.

Folder user\_guide chỉ là folder document, các bạn xóa nó đi vì không cần thiết cho ứng dụng, nếu các bạn muốn để lại tham khảo thì cũng không sao. Folder Application là folder chứa source web trong quá trình mình phát triển. Mọi file đều nằm trong folder này và tùy vào loại file mà lưu những vị trí khác nhau. Ở những bài tới, mình sẽ hướng dẫn các bạn sau.

Trong Application các bạn thấy có 3 folders quan trọng nhất đó là **Controllers**, **Models** và **Views**. Các folder còn lại chúng ta sẽ đề cập sau vì một lúc nói hết các folder các bạn cũng chưa chắc hiểu hết, chỉ khi nào cần dùng folder nào mình sẽ giới thiệu.

## Bài 2: Tạo Controller Trong CodeIgniter

Ở bài trước chúng ta đã tìm hiểu qua cấu trúc folder của Codeigniter (CI), trong bài này, chúng ta bắt đầu tìm hiểu qua mô hình MVC trong CodeIgniter. Đầu tiên, chúng ta sẽ tìm hiểu về controller. Nội dung bao gồm:

- Tạo mới controller trong codeigniter
- Truyền biến vào controller
- Xác định controller mặc định
- Hàm khởi tạo
- Xóa đường dẫn index.php

### 1. Tạo Mới Controller Trong Codeigniter

Tất cả các controller trong CodeIgniter đều được đặt trong thư mục **Application/Controllers** của CI. Mặc định khi cài đặt, CI đã tạo một controller tên là **welcome.php**, bạn xóa file này đi và tạo một file **hello.php** và điền nội dung vào là:

```
if (!defined('BASEPATH'))
    exit('No direct script access allowed');

class Hello extends CI_Controller {
    public function index() {
        echo 'TKPro CodeIgniter xin chào các bạn';
    }
}
```

Dòng **if (!defined('BASEPATH')) exit('No direct script access allowed');** là dòng Security bảo vệ file của các bạn, nó không cho truy cập thẳng vào file mà phải thông qua file index.php ở mức ngoài cùng.

Lớp Hello là tên Controller của chúng ta, nó kế thừa lớp **CI\_Controller** của hệ thống Codeigniter, tất cả các Controller đều phải **kế thừa** **CI\_Controller** thì mới sử dụng được các thư viện của CI và **tên Controller phải bắt đầu bằng chữ hoa**.

Hàm index là **Action** (method) của controller, đây là một hàm mặc định của tất cả các controller trong Codeigniter nghĩa là nếu đường dẫn bạn chỉ gõ **domain.com/hello** thì mặc định nó sẽ chạy file index

Bạn ra trình duyệt gõ đường dẫn **localhost/ten\_project/index.php/hello/index**, kết quả xuất ra màn hình là dòng "*'TKPro CodeIgniter xin chào các bạn'*" thì chúng ta đã tạo mới thành công rồi.

Bạn vào file Controller Hello thêm một hàm other mới như sau:

```
class Hello extends CI_Controller {  
  
    public function index() {  
        echo 'TKPro CodeIgniter xin chào các bạn';  
    }  
  
    public function other() {  
        echo 'TKPro CodeIgniter: Một hàm khác của controller Hello';  
    }  
}
```

Như vậy là ta đã tạo thêm một hàm (Action) mới trong controller Hello, bây giờ bạn ra trình duyệt gõ đường dẫn “localhost/codeigniter/index.php/hello/other” màn hình sẽ xuất hiện dòng “TKPro CodeIgniter: Một hàm khác của controller Hello”.

Qua hai ví dụ trên ta thấy **mỗi Controller trong Codeigniter** ta có thể tạo nhiều Action (hàm) trong đó, và mỗi action sẽ có những nhiệm vụ riêng biệt.

## 2. Truyền Biến Vào Controller

Trong mô hình MVC của các Framework, biến truyền vào theo phương thức GET đều có dạng “domain.com/controller/action/parameter1/parameter2/...”. Trong Codeigniter cũng vậy, để truyền biến vào Controller bạn sẽ có đường dẫn là “domain.com/index.php/controller/action/parameter1/parameter2/...”. Trong hàm (Action) của controller ta sẽ nhận nó bằng cách truyền những biến có vị trí tương ứng với từng parameter trên url.

### Ví dụ 1:

```
class Hello extends CI_Controller  
{  
    public function index($message = '')  
    {  
        echo 'TKPro CodeIgniter: ' . $message;  
    }  
}
```

Bạn ra trình duyệt gõ đường dẫn “localhost/codeigniter/index.php/hello/index/Hello Codeigniter Newbie” thì màn hình sẽ xuất hiện chữ “TKPro CodeIgniter: Hello Codeigniter Newbie”. Bạn hãy thử thay đổi biến truyền vào để kiểm tra.

Biến \$message truyền vào hàm (action) index mình gán nó giá trị khởi tạo bằng giá trị trống. Tại sao mình phải làm vậy? Tại vì theo nguyên tắc tất cả các hàm nếu truyền không đủ biến vào nó sẽ bị lỗi, nếu ta không gán giá trị mặc định thì nếu người dùng chỉ gõ “localhost/codeigniter/index.php/hello/index/” nó sẽ bị lỗi ngay, vì thế tất cả các biến truyền vào bạn phải gán giá trị mặc định để cho an toàn.

### Ví dụ 2:

```
class Hello extends CI_Controller
{
    public function index($id = 0, $message = '')
    {
        echo 'TKPro CodeIgniter: ID='.$id.', message =' . $message;
    }
}
```

Trong ví dụ này mình truyền 2 biến vào hàm (action) index và gán giá trị mặc định cho biến, bây giờ bạn thử ra trình duyệt gõ các đường dẫn sau:

- localhost/codeigniter/index.php/hello/index/12/Hello => kết quả là “TKPro CodeIgniter: ID = 12, message = Hello”
- localhost/codeigniter/index.php/hello/index/12 => kết quả là “TKPro CodeIgniter: ID = 12, message = ”
- localhost/codeigniter/index.php/hello/index//Hello => kết quả là “TKPro CodeIgniter: ID = , message = Hello”

Qua ba ví dụ trên ta thấy các biến truyền vào hàm nó tuân theo thứ tự trên URL.

### 3. Xác Định Controller Mặc Định

**Controller mặc định** là controller sẽ được gọi khi trên url bạn không gọi đến một controller nào. Bạn vào file **application/config/routes.php** kéo xuống phía dưới tìm đến dòng `$route['default_controller'] = "welcome";`. Tại đây bạn sửa giá trị của biến `$route['default_controller']` thành tên controller mà bạn muốn chạy mặc định, ví dụ tôi sẽ sửa thành `$route['default_controller'] = "hello/other";` sau đó ra trình duyệt gõ đường dẫn “**localhost/codeigniter**” thì mặc định nó sẽ chạy đến controller **“hello/other”**.

### 4. Hàm Khởi Tạo

Trong lập trình hướng đối tượng thì tất cả các lớp đối tượng có hàm khởi tạo, hàm này sẽ chạy đầu tiên khi bạn khởi tạo một đối tượng mới. Trong PHP hàm khởi tạo được quy ước là đặt trùng tên với tên Lớp hoặc là bạn đặt tên **\_\_construct()**. Nếu trong Controller bạn muốn sử dụng hàm khởi tạo thì bắt buộc bạn phải gọi đến hàm khởi tạo của cha nó (`CI_Controller`), vì trong **PHP** nếu hàm con kế thừa hàm cha mà hàm con có hàm khởi tạo thì nó sẽ chạy hàm khởi tạo của con chứ không chạy hàm khởi tạo của cha, mà trong hàm khởi tạo của cha lại chứa những đoạn code thiết lập hệ thống cho CI nên bắt buộc phải chạy nó.

```
class Hello extends CI_Controller
{
    // Hàm khởi tạo
    function __construct() {
```

```
// Gọi đến hàm khởi tạo của cha  
parent::__construct();  
}  
  
public function index()  
{  
    echo 'TKPro.edu.vn';  
}  
}
```

## 5. Xóa Đường Dẫn Index.php trong codeigniter

Trong các ví dụ trên URL để gọi Controller trong codeigniter luôn có file index.php nhìn rất là mất thẩm mỹ, để bỏ file index.php trên đường dẫn url trong codeigniter bạn làm như sau:

Tạo file .htaccess cùng cấp với file index.php, tức là ở ngoài cùng, sau đó copy nội dung này vào:

```
Options +FollowSymlinks  
RewriteEngine on  
RewriteCond %{REQUEST_FILENAME} !-f  
RewriteCond %{REQUEST_FILENAME} !-d  
RewriteRule ^(.+)/?$ index.php/$1 [L]
```

Sau đó bạn ra trình duyệt gõ URL “localhost/codeigniter/hello/other” thì CI tự động hiểu và gọi đến Controller **Hello** và hàm (Action) **other**.

## Bài 3: Load View Trong Codeigniter

Trong bài này, chúng ta sẽ tìm hiểu cách load view trong CI, nội dung bao gồm:

- Tạo Mới Một View Trong Codeigniter
- Load View Trong Controller

### 1. Tạo Mới Một View Trong Codeigniter

Trong CI, view được đặt trong thư mục application/views. Các bạn vào folder đó tạo một file tên là login\_view.php, trong view các bạn tạo một form login như sau:

```
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    </head>
    <body>
        <form method="POST" action="">
            Username : <input type="text" name="username" value="">
<br/>
            Password : <input type="password" name="password"
value=""> <br/>
            <input type="submit" name="submit_login" value="Login"/>
        </form>
    </body>
</html>
```

### 2. Load View Trong Controller

Có rất nhiều cách load view trong CI, dưới đây là những cách load thông dụng

#### *Load View*

Sau khi có view login\_view.php ở trên, bạn vào tạo một controller mới với tên là Login có nội dung như sau:

```
class Login extends CI_Controller
{
    // Hàm load form login
    public function load_form()
    {
        // Load view
        $this->load->view('login_view');
    }
}
```

Controller Login dùng để xử lý login, hàm (action) load\_form dùng để load form login cho người dùng nhập dữ liệu.

Cú pháp để load view là :

```
$this->load->view('tên view');
```

Bạn ra trình duyệt gõ URL “localhost/codeigniter/login/form\_login” nếu xuất hiện một form login thì bạn đã load view trong controller thành công rồi đấy.

### *Load Nhiều View*

Để load nhiều view bạn chỉ cần dùng cú pháp load view nhiều lần, ví dụ:

```
$this->load->view('view1');  
$this->load->view('view2');  
$this->load->view('view3');  
$this->load->view('view4');  
$this->load->view('view5');
```

### *Load View Ở Sub Folder*

Thông thường ta sẽ lưu các view liên quan với nhau trong một folder riêng, ví dụ:

- application/views/product/lists.php
- application/views/product/add\_form.php
- application/views/product/add\_edit.php

Vậy để load view nằm ở sub folder dạng này ta dùng cú pháp sau:

```
$this->load->view('subfolder/view1');
```

### *Load View Ở Dạng Biến*

Nếu bạn muốn load một view ở dạng biến thì bạn dùng cú pháp sau:

```
$var = $this->load->view('view_name', '', true);
```

### **Ví dụ:**

```
class Login extends CI_Controller  
{  
    // Hàm load form login  
    public function form()  
    {  
        // Load view lưu vào một biến  
        $login_form = $this->load->view('login_view', '', true);  
  
        // Xuất view ra màn hình  
        echo $login_form;  
    }  
}
```

### Truyền Dữ Liệu Qua View

Để truyền dữ liệu qua view thì tất cả dữ liệu bạn phải đưa vào một **mảng kết hợp \$data**, Controller sẽ tự động tạo các biến bên view tương ứng với các key và các value trong mảng \$data đó. Mỗi dữ liệu có thể ở các kiểu như: float, double, int, string, object, array.

#### Ví dụ:

*Trong controller Login: (File application/controllers/Login.php)*

```
class Login extends CI_Controller
{
    // Hàm load form login
    public function form()
    {
        // Data cần truyền qua view
        $data = array(
            'title' => 'Đây là trang login',
            'message' => 'Nhập Thông Tin Đăng Nhập'
        );

        // Load view và truyền data qua view
        $this->load->view('login_form', $data);
    }
}
```

*Trong View Login\_form (File application/views/login\_view.php):*

Bạn xóa hết nội dung cũ và gõ vào đoạn code PHP sau:

```
// Tương ứng với $data['title'] bên controller
echo $title;

// Tương ứng với $data['message'] bên controller
echo $message;
```

## Bài 4: Load Model Trong Codeigniter

Trong bài này chúng ta sẽ tìm hiểu cách tạo mới một model trong CI, nội dung bao gồm:

- Cấu hình database
- Tạo mới một model trong codeigniter
- Load model trong controller

### 1. Cấu Hình Database

Trong CI, để kết nối với Database chúng ta phải cấu hình thông tin cho đúng. Bạn mở file `application/config/database.php`, sau đó kéo xuống bên dưới chỉnh lại một số thông tin sau:

```
$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'Tên Đăng Nhập Database'; //Chỉnh ở đây
$db['default']['password'] = 'Mật Khẩu Đăng Nhập'; // Chỉnh ở đây
$db['default']['database'] = 'Tên Database'; // Chỉnh ở đây
$db['default']['dbdriver'] = 'mysql';
$db['default']['dbtextareafix'] = '';
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = '';
$db['default']['char_set'] = 'utf8';
$db['default']['dbcollat'] = 'utf8_general_ci';
$db['default']['swap_textarea'] = '';
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;
```

Sau khi chỉnh xong bạn lưu lại và vào một **Controller** nào đó chạy lệnh này: `$this->load->database();` Nếu không bị lỗi thì bạn config dababase đúng, còn không thì bạn kiểm tra lại thông tin nhé.

### 2. Tạo Mới Một Model Trong Codeigniter

Tất cả các file model đều nằm trong thư mục `application/models` nên bạn vào đó tạo một `file news_model.php` có nội dung như sau:

```
class News_model extends CI_Model
{
    public function getList(){
        // Code
    }
}
```

Trong đó News\_model là tên model, nó phải giống với tên file news\_model.php và ký tự đầu tiên phải ghi hoa. Mọi model đều phải kế thừa lớp CI\_Model thì mới hoạt động được. Vậy là bạn đã tạo xong model **news\_model** rồi đó.

### 3. Load Model Trong Controller

Để load model trong controller ta dùng cú pháp: **\$this->load->model('ten\_model')**, và để gọi các hàm trong model ta dùng cú pháp: **\$this->ten\_model->ham()**.

**Ví dụ:** Bạn tạo một mới Controller tên là News.php với nội dung như sau:

```
class News extends CI_Controller
{
    function news_list()
    {
        // Load model
        $this->load->model('news_model');

        // Gọi function trong model
        $news_list = $this->news_model->getList();
    }
}
```

Nếu bạn muốn load model và đặt cho nó một cái tên khác thì ta dùng cú pháp: **\$this->load->model('ten\_model','ten\_khac')**; và để gọi các hàm trong model ta dùng cú pháp: **\$this->ten\_khac->ham()**.

**Ví dụ:**

```
class News extends CI_Controller
{
    function news_list()
    {
        // Load model
        $this->load->model('news_model', 'm_news');

        // Gọi function trong model
        $news_list = $this->m_news->getList();
    }
}
```

Trong ứng dụng nếu một model nào đó luôn luôn sử dụng thì bạn có thể dùng chức năng Autoload của Codeigniter, bạn mở file **application/config/autoload.php** và tìm đến dòng (hình như dòng cuối cùng) **\$autoload['model'] = array()**, sau đó bạn thêm model bạn cần load vào, ví dụ tôi thêm model news\_model: **\$autoload['model'] = array('news\_model')**.

Sau khi thêm model vào autoload thì ở controller bạn chỉ cần sử dụng nó mà không cần phải load model đó ra.

```
class News extends CI_Controller
{
    function news_list()
    {
        // Gọi function trong model
        $news_list = $this->news_model->getList();
    }
}
```

Nếu model của bạn nằm trong một folder. Ví dụ model của bạn nằm trong folder **application/models/news/news\_model.php** thì bạn dùng cú pháp sau: **\$this->load->model('news/news\_model')** và vẫn dùng cú pháp cũ để gọi hàm trong model: **\$this->news\_model->getList();**

**Ví dụ:** file model nằm trong thư mục: application/models/news/news\_model.php

```
class News extends CI_Controller
{
    function news_list()
    {
        // Load model
        $this->load->model('news/news_model');

        // Gọi function trong model
        $news_list = $this->news_model->getList();
    }
}
```

## Bài 5: Load Library Session Trong Codeigniter

Session là một thư viện rất quan trọng trong CI. Chúng ta sẽ sử dụng nó cho rất nhiều công việc như quản lý phiên làm việc, kiểm tra đăng nhập, xây dựng giỏ hàng, ...

### 1. Thế nào là session

Trong CI, session được viết dưới dạng class hay còn gọi là library session. Session trong CI được xây dựng dựa trên một khóa đặc biệt, còn gọi là **Encryption key**, để tăng tính tương tác và bảo mật cho ứng dụng. Để khai báo library session và gọi các phương thức bên trong nó, bạn cần phải vào thư mục file config, tìm file config.php và tìm đến dòng sau và gõ vào encryption key.

```
$config['encryption_key'] = '';
```

Trong cặp dấu nháy ('...'), bạn có thể điền bất cứ gì cũng được. Chẳng hạn như sau:

```
$config['encryption_key'] = 'tkpro.edu.vn';
```

**Lưu ý:** Đây là bước bắt buộc, nếu bạn để trống và gọi library session ra để sử dụng thì sẽ nhận thông báo lỗi như sau.

In order to use the Session class you are required to set an encryption key in your config file.

### 2. Load library session

Để có thể sử dụng library session, đầu tiên chúng ta phải load nó vào controller. Ví dụ ở đây tôi khởi tạo controller demo và action sẽ là index. Ngay tại hàm constructor tôi sẽ tiến hành gọi nó ra với cú pháp như sau.

```
$this->load->library("session");
```

#### Cấu trúc code:

```
class Demo extends CI_Controller{
    public function __construct(){
        parent::__construct();
        $this->load->library("session");
    }

    public function index(){
    }
}
```

Để sử dụng nó chúng ta sẽ dùng phương thức **\$this->session** tương tự như cách gọi Model.

### 3. Hàm Khởi tạo Session

Để có thể khởi tạo session một cách dễ dàng chúng ta sẽ sử dụng phương thức:

```
$this->session->set_userdata("ten", "gia tri")
```

Dữ liệu truyền vào đây có hai lựa chọn, một là truyền vào tên, hai là truyền vào giá trị của nó là gì. Ngoài ra chúng ta cũng có thể khởi tạo chúng theo dạng array (mảng):

```
$data=array(  
    "username" => "Huỳnh Tân Khải",  
    "email" => "lienhe@tkpro.edu.vn",  
    "website" => "tkpro.edu.vn",  
    "gender" => "Male",  
) ;
```

Và lúc này tôi đang có 4 giá trị cần lưu vào session theo dạng mảng theo cú pháp sau:

```
$this->session->set_userdata($data);
```

Khởi tạo xong, khi cần truy xuất, chúng ta sử dụng phương thức **userdata** của đối tượng session và truyền vào tên của thành phần cần lấy. Ví dụ, ở đây mình truyền vào tên của session là username.

```
$this->session->userdata("username")
```

## 4. Hàm hủy session

Hủy session có 2 cách tắt cả.

- Hủy một session được chỉ định bất kì ta dùng phương thức **unset\_data**. Ví dụ, mình hủy session có tên là username như sau:

```
$this->session->unset_data("username")
```

- Hủy hàng loạt session theo cấu trúc array. Ví dụ như sau:

```
$item = array('username' => '', 'email' => '', 'website' => '' );  
$this->session->unset_userdata($items);
```

- Với thao tác này, chúng ta có thể hủy toàn bộ session, nhưng làm thế thì quá dài dòng, bản thân CI cũng đã hỗ trợ một hàm hủy toàn bộ session, đó là hàm **sess\_destroy()**:

```
$this->session->sess_destroy();
```

## 5. Thực hành Session

Chúng ta sẽ thực hiện một ví dụ nhỏ để các bạn hình dung được quy trình làm việc của session. Tại thư mục **application/controller**, các bạn tạo một file có tên là **demo.php**. Vì ví dụ có sử dụng hàm **base\_url**, **redirect**, nên ngay tại constructor, chúng ta tiến hành gọi helper URL & library Session vào controller.

```
class Demo extends CI_Controller{  
    public function __construct(){  
        parent::__construct();  
        $this->load->helper("url");  
        $this->load->library("session");  
    }  
  
    public function index(){
```

```
$data=array(
    "username" => "Huỳnh Tân Khải",
    "email" => "lienhe@tkpro.edu.vn",
    "website" => "tkpro.edu.vn",
    "gender" => "Male"
);
$this->session->set_userdata($data);
redirect(base_url(),"index.php/demo/index2");
}
```

Ngay tại action index, chúng ta khai báo biến \$data dưới dạng array. Để khởi tạo session, chúng ta dùng phương thức **set\_userdata(\$data)** và khởi tạo toàn bộ 4 giá trị trên. Để kiểm tra xem code có hoạt động đúng không, chúng ta truy cập vào đường link *localhost/citest/index.php/demo/index2* nếu trình duyệt tự động chuyển sang index2 thì chúng ta đã khai báo và sử dụng session thành công, để việc kiểm tra tốt hơn thì ở action index2, chúng ta sẽ hiển thị từng session ra như sau:

```
public function index2() {
    $user=$this->session->userdata("username");
    $website=$this->session->userdata("website");
    $email=$this->session->userdata("email");
    echo "Username: $user, Email: $email, Website: $website";
}
```

Nếu màn hình chuyển sang link *localhost/citest/index.php/demo/index2* và xuất ra kết quả như sau thì xem như các bạn đã khởi tạo và sử dụng session thành công.

Để lấy tất cả giá trị có trong session, chúng ta sử dụng phương thức **all\_userdata** với cú pháp như sau.

```
public function index2() {
    $data=$this->session->all_userdata();
    echo "<pre>";
    print_r($data);
    echo "</pre>";
}
```

Kết quả thu được sẽ tương tự như sau:

```
Array
(
[session_id] => cf7966232a5fdb726653e240ef5cb8d4
[ip_address] => 127.0.0.1
[user_agent] => Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/33.0.1750.154 Safari/537.36
[last_activity] => 1395055268
[user_data] =>
[username] => Huỳnh Tân Khải
[website] => tkpro.edu.vn
[email] => lienhe@tkpro.edu.vn
[gender] => male
)
```

Sau khi khởi tạo & sử dụng session, chúng ta tiến hành hủy session. Để hủy session, chúng ta tạo thêm một action có tên là index3 dùng để hủy toàn bộ session. Để kiểm tra, chúng ta echo câu thông báo “**Huy session thành công**” chạy link <localhost/codeig/index.php/demo/index3> để kiểm tra.

```
public function index3() {
    $this->session->sess_destroy();
    echo "Huy session thành công.";
}
```

## 6. Các vấn đề mở rộng trong session

Cũng giống như các framework khác đều tồn tại một khái niệm gọi là **flashdata**, là một dạng dữ liệu không tồn tại lâu dài, nó chỉ xuất hiện ra một lần và sau đó nó sẽ bị hủy. Sử dụng nó trong thực tế như thế nào? nó hay được sử dụng trong các phiên đăng nhập, chẳng hạn như đăng nhập thành công sẽ có thông báo trả về ngay trên trang bạn đang truy cập, và thông báo đó sẽ biến mất nếu chúng ta refresh trình duyệt.

Để khởi tạo flashdata, chúng ta sử dụng cú pháp như sau:

```
$this->session->set_flashdata("ten", "gia tri");
```

Sử dụng nó cũng tương tự như sử dụng session:

```
$this->session->flashdata(ten);
```

Tên của flashdata luôn phải bắt đầu với key là **flash\_**, để cho các bạn hiểu rõ hơn về nó, chúng ta sẽ thực hiện một ví dụ nhỏ như sau. Chúng ta sẽ xuất ra câu thông báo “**Khởi tạo session thành công**” ngay khi chúng ta vừa chuyển sang action index2. Chúng ta khởi tạo flashdata ở action index tôi đặt tên là flash\_open, tại action index2 chúng ta hiển thị nó ra bằng phương thức flashdata(“flash\_open”) tương ứng với tên flash mà chúng ta khai báo ở action index.

```
public function index() {
    $data=array(
        "username" => "Huỳnh Trần Khải",
        "email" => "lienhe@tkpro.edu.vn",
        "website" => "tkpro.edu.vn",
        "gender" => "Male",
    );
    $this->session->set_userdata($data);
    $this->session->set_flashdata("flash_open", "Khởi tạo session thành công");
    redirect(base_url(),"index.php/demo/index2");
}

public function index2() {
    echo $this->session->flashdata("flash_open");
    $data=$this->session->all_data();
    echo "<pre>";
    print_r($data);
    echo "</pre>";
}
```

Để kiểm tra, chúng ta thấy trình duyệt chuyển sang trang index2 và xuất hiện câu thông báo "**Khởi tạo session thành công**". Tuy nhiên, nếu chúng ta refresh trình duyệt thì ngay lập tức cái flashdata này đã bị hủy và chúng ta sẽ không nhận được câu thông báo này nữa.

Vẫn đè mà mình muốn các bạn chú ý nhiều nhất chính là việc lưu trữ session trong database, có nghĩa là chúng ta sẽ tạo ra một table dùng để chứa các session. Giả sử chúng ta đang làm việc với giỏ hàng và chúng ta cần lưu hàng ngàn session ứng với hàng ngàn sản phẩm trong giỏ hàng. Tuy nhiên, sức chứa của session thì có hạn, và khi session bị quá tải thì nó không thể lưu trữ nhiều hơn được nữa. Nắm bắt được yêu cầu này của lập trình viên, CI hỗ trợ chúng ta giải quyết bài toán này một cách dễ dàng bằng cách chúng ta tạo ra một table tên là **ci\_sessions** với các cột như sau:

```
CREATE TABLE IF NOT EXISTS `ci_sessions` (
    session_id varchar(40) DEFAULT '0' NOT NULL,
    ip_address varchar(45) DEFAULT '0' NOT NULL,
    user_agent varchar(120) NOT NULL,
    last_activity int(10) unsigned DEFAULT 0 NOT NULL,
    user_data text NOT NULL,
    PRIMARY KEY (session_id),
    KEY `last_activity_idx` (`last_activity`)
);
```

Nhiệm vụ của table này là chứa tất cả dữ liệu mà chúng ta cần chứa, nó sẽ lưu trữ tất cả trong cột **user\_data**, nó lấy mọi giá trị của session sau đó convert sang json hoặc một chuẩn nào đó của CI, sau đó nó sẽ lưu trữ dữ liệu dưới dạng là text.

Ngoài việc tạo thêm table cho database, chúng ta còn phải cấu hình trong file config.php một vài thông tin như sau, bật tính năng sử dụng thao tác database trong session.

```
$config['sess_use_database'] = TRUE;
```

Và định nghĩa table name cho nó là ci\_sessions.

```
$config['sess_table_name'] = 'ci_sessions';
```

Thực tế là không nên lạm dụng thao tác database trong session vì nó sẽ khiến cho hệ thống của chúng ta bị nghẽn và chậm hơn vì phải xử lý thao tác đọc, ghi dữ liệu vào CSDL quá nhiều. CI chỉ đưa ra giải pháp này cho chúng ta tiện việc lưu trữ dữ liệu session có kích thước lớn. Chúng ta chỉ sử dụng nó khi nào phải làm việc với giỏ hàng, lưu trữ các thông tin của từng sản phẩm.

## Bài 6: Load Library Database Trong Codeigniter

Trong bài này bạn sẽ được học:

- Các thao tác xử lý database.
- Làm quen với Active Record.

### 1. Load library database

Đầu tiên, chúng ta cần tạo ra một controller có tên là **user** và action **index**, chúng ta sẽ gọi library database ngay tại constructor. Việc gọi library ngay tại constructor giúp chúng ta có thể tái sử dụng các library này trong các action sau đó.

```
class User extends CI_Controller{
    public function __construct(){
        parent::__construct();
        $this->load->library("database");
    }

    public function index(){
    }
}
```

Như thế này là chưa đủ để chúng ta có thể sử dụng library, các bạn cần phải khai báo database ở trong thư mục *application/config/database.php* (các bạn xem lại bài Load Model Trong Codeigniter). Tiếp theo, chúng ta cần có một table với tên là user (các bạn có thể đặt tùy ý).

```
CREATE TABLE `user` (
    `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
    `username` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
    `password` char(32) COLLATE utf8_unicode_ci DEFAULT NULL,
    `level` int(1) DEFAULT '1',
    PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci
```

### 2. Thao tác Database

Đầu tiên, chúng ta sẽ tìm hiểu cách viết câu truy vấn bình thường, sau đó mới thao tác tới các active record do CI cung cấp. Để viết được các câu truy vấn bình thường, chúng ta dùng thuộc tính như sau:

```
$this->db->query("SELECT * FROM USER order By id");
```

Để có thể lấy ra toàn bộ dữ liệu trong table user ta có thao tác như sau.

```
class User extends CI_Controller{
    public function __construct(){
        parent::__construct();
```

```
    $this->load->library("database");
}

public function index(){
    $query=$this->db->query("SELECT * FROM USER order By id");
    $data=$query->result_array();
    echo "<pre>";
    print_r($data);
    echo "</pre>";
}
```

Ở đây, chúng ta dùng biến \$data để chứa kết quả trả về, hàm **result\_array** tương ứng với hàm **mysql\_fetch\_assoc** của **PDO**. Để xem kết quả trả về, chúng ta chạy link *localhost/codeig/index.php/user/index*.

```
[0] => Array
(
[id] => 1
[username] => admin
[password] => 123456
[level] => 2
)
[1] => Array
(
[id] => 2
[username] => alibaba
[password] => 123456
[level] => 1
)
[2] => Array
(
[id] => 3
[username] => vicky
[password] => 123456
[level] => 1
)
[3] => Array
(
[id] => 4
[username] => jacky
[password] => dsads
[level] => 1
)
```

Ok, nếu kết quả trên trình duyệt hiển thị như thế này, xem như việc lấy toàn bộ record của bạn đã thành công rồi đấy. Tiếp theo, chúng ta sẽ sử dụng **active record** trong quá trình làm việc với database.

```
class User extends CI_Controller{
    public function __construct(){
        parent::__construct();
        $this->load->library("database");
    }
}
```

```
public function index(){
    $query=$this->db->get("user");
    $data=$query->result_array();
    echo "<pre>";
    print_r($data);
    echo "</pre>";
}
```

Khi dùng phương thức get(), chúng ta chỉ cần truyền vào tên bảng. Việc sử dụng phương thức này giúp chúng ta rút ngắn câu lệnh một cách khá là dễ dàng. Ngoài ra, chúng ta cũng thể cài đặt một số ràng buộc cho câu truy vấn thông qua một số phương thức sau:

```
$this->db->limit();
$this->db->select();
$this->db->join();
$this->db->like();
$this->db->select_min();
$this->db->select_max();
```

Để trả về số lượng dòng dữ liệu, chúng ta sử dụng phương thức `$query->num_rows.`

Để trả về chỉ một record đầu tiên, chúng ta dùng: `$query->row_array();`

Một số ví dụ như sau:

Chúng ta không muốn liệt kê tất cả các field mà chỉ liệt kê một số field nào đó, chúng ta dùng phương thức select:

```
class User extends CI_Controller{
    public function __construct(){
        parent::__construct();
        $this->load->library("database");
    }

    public function index(){
        $this->db->select("id, username, level");
        $query=$this->db->get("user");
        $data=$query->result_array();
        echo "<pre>";
        print_r($data);
        echo "</pre>";
    }
}
```

Để sắp xếp id theo chiều giảm dần, chúng ta dùng phương thức order\_by như sau:

```
class User extends CI_Controller{
    public function __construct(){
        parent::__construct();
        $this->load->library("database");
    }

    public function index(){
```

```
$this->db->select("id, username, level");
$this->db->order_by("id", "desc");
$query=$this->db->get("user");
$data=$query->result_array();
echo "<pre>";
print_r($data);
echo "</pre>";
}
```

Tương tự để giới hạn kết quả trả về, chúng ta sử dụng phương thức limit() với đối số thứ nhất là tổng số record trả về và đối số thứ 2 chính là vị trí bắt đầu để lấy record. (Nó khác câu truy vấn bình thường ở chỗ hoán đổi vị trí 2 đối số này).

```
class User extends CI_Controller{
    public function __construct(){
        parent::__construct();
        $this->load->library("database");
    }

    public function index(){
        $this->db->select("id, username, level");
        $this->db->order_by("id", "desc");
        $this->db->limit(7, 0);
        $query=$this->db->get("user");
        $data=$query->result_array();
        echo "<pre>";
        print_r($data);
        echo "</pre>";
    }
}
```

Để thực thi mệnh đề điều kiện, ví dụ trả về các user có level = 2 thì chúng ta có cú pháp như sau:

```
class User extends CI_Controller{
    public function __construct(){
        parent::__construct();
        $this->load->library("database");
    }

    public function index(){
        $this->db->select("id, username, level");
        $this->db->order_by("id", "desc");
        $this->db->limit(7, 0);
        $this->db->where("level", "2");
        $query=$this->db->get("user");
        $data=$query->result_array();
        echo "<pre>";
        print_r($data);
        echo "</pre>";
    }
}
```

Tiếp theo, để trả về phần tử lớn nhất, phần tử nhỏ nhất, chúng ta sử dụng phương thức select\_max và select\_min như ví dụ sau:

```
class User extends CI_Controller{
    public function __construct(){
        parent::__construct();
        $this->load->library("database");
    }

    public function index(){
        $this->db->select("id, username, level");
        $this->db->order_by("id", "desc");
        $this->db->limit(7, 0);
        $this->db->where("level", "2");
        $this->db->select_min("id");
        $query=$this->db->get("user");
        $data=$query->result_array();
        echo "<pre>";
        print_r($data);
        echo "</pre>";
    }
}
```

Tiếp theo, chúng ta sẽ tìm hiểu ba thao tác quan trọng khác khi làm việc với CSDL, đó là các thao tác: thêm dữ liệu (insert), cập nhật dữ liệu (update), xóa dữ liệu (delete).

Thêm dữ liệu:

Cú pháp: `$this->db->insert("table", tên biến truyền vào)`

Ví dụ:

```
public function index2() {
    $data=array(
        "username" => "Huỳnh Tân Khải",
        "password" => "1212445",
        "level"     => "2",
    );
    $this->db->insert("user", $data);
}
```

Kiểm tra bằng cách, chạy link `localhost/codeig/index.php/user/index2`, sau đó quay trở lại action index xem kết quả.

Cập nhật dữ liệu:

Cú pháp: `$this->db->update("table", tên biến truyền vào)`

Ví dụ:

```
public function index3() {
    $data=array(
        "username" => "Huỳnh Tân Khải",
        "password" => "123456",
        "level"     => "1",
    );
    $this->db->where("id", "5");
    if($this->db->update("user", $data)) {
```

```
        echo "Update Thanh cong";
    }else{
        echo "Update That bai";
    }
}
```

Chúng ta đã tiến hành update user có id = 5, nếu màn hình trình duyệt trả về là “Update Thanh Cong” thì việc cập nhật dữ liệu đã thành công. Kiểm tra bằng cách chạy link *localhost/codeig/index.php/user/index3*, quay trở lại action index để xem kết quả.

Cập nhật dữ liệu:

Cú pháp: `$this->db->delete("table")`

Ví dụ:

```
public function index4() {
    $this->db->where("id", "5");
    if($this->db->delete("user")){
        echo "Xoa thanh cong";
    }else{
        echo "Xoa that bai";
    }
}
```

Ở đây chúng ta đã xóa người dùng có id = 5, chạy link *localhost/codeig/index.php/user/index4* để tiến hành xóa và quay lại action index để xem kết quả.

Như vậy chúng ta đã tìm hiểu các thao tác làm việc với CSDL trong CI. Các bạn chú ý rằng các thao tác này không nên viết trong controller vì nó trái với nguyên tắc của CI, bản thân của controller không tương tác trực tiếp với database, để làm việc tốt với CSDL, chúng ta nên viết nó trong model.

### 3. Thao tác database trong model

Đầu tiên, chúng ta cần phải khởi tạo file model và chúng ta cũng cần ghi nhớ là tên file của model không được phép trùng tên với controller, nếu 2 file này trùng tên nhau (dù ở 2 thư mục khác nhau) thì CI sẽ báo lỗi ngay, tại thư mục **application/models**, chúng ta khởi tạo file UserModel.php và khai báo như sau:

```
<?php
class UserModel extends CI_Model{
    public function __construct(){
        parent::__construct();
        $this->load->database();
    }
}
?>
```

Để tiến hành lấy toàn bộ record trong table user, chúng ta tạo một function có tên là listUser. Function này thao tác với bảng user và trả kết quả về thông qua câu lệnh return.

```
<?php  
class UserModel extends CI_Model{  
    public function __construct(){  
        parent::__construct();  
        $this->load->database();  
    }  
  
    public function listUser(){  
        $this->db->select("id, username, level");  
        $query=$this->db->get("user");  
        return $query->result_array();  
    }  
}  
?>
```

Tại controller user, chúng ta tạo action index5, action này sẽ gọi model UserModel để lấy dữ liệu và truyền vào view để hiển thị. Để hiển thị dữ liệu, chúng ta tạo view có tên là **list\_view.php** (trong thư mục **application/views**) có nội dung như sau:

```
echo "<pre>";  
print_r($data);  
echo "</pre>";
```

Như vậy, nội dung của action index5 như sau:

```
public function index5() {  
    $this->load->model("UserModel");  
    $data['data']=$this->UserModel->listUser();  
    $this->load->view("user/list_view", $data);  
}
```

Các bạn truy cập link *localhost/codeig/index.php/user/index5* để kiểm tra.

## Bài 7: Phân trang trong Codeigniter

Trong bài này bạn sẽ được học:

- Các thao tác xử lý với phân trang.
- Các vấn đề mở rộng trong pagination.

Đây là một library cũng khá là phổ biến, hay được sử dụng trong quá trình xây dựng website, nó có nhiệm vụ giúp chúng ta phân trang một cách dễ dàng. Bản chất của library pagination là nó chỉ tạo ra những đường link, chứ nó không hỗ trợ chúng ta giới hạn kết quả trả về, để làm được điều đó chúng ta phải thao tác với model, tức là phải xử lý truy xuất database.

### 1. Load Library pagination

Cũng giống như các library trước, để có thể thao tác với nó, chúng ta phải load nó trong controller. Chúng ta tạo controller là Page và action là index. Trong controller Page chỉ có một action index nên chúng ta sẽ gọi library ngay trong action này. Trong constructor, chúng ta cũng phải load helper url ra để có thể khai báo đường dẫn đến controller và action muốn phân trang, helper là gì thì ở bài sau chúng ta sẽ tìm hiểu.

```
class Page extends CI_Controller{
    public function __construct(){
        parent::__construct();
        $this->load->helper("url");
    }

    public function index(){
        $this->load->library("pagination");
    }
}
```

Để bắt đầu tiến hành **phân trang** chúng ta cần khai báo một số tham số như sau:

```
$config['base_url'] = "Đường dẫn controller & action muốn phân trang";
$config['total_rows'] = "Tổng số record";
$config['per_page'] = "Số record trên một trang";
$this->pagination->initialize($config); // Chạy phân trang
```

Để giới hạn kết quả trả về chúng ta cần phải thao tác với model. Trong ví dụ này, chúng ta sẽ dùng lại file UserModel.php, cái đầu tiên mà chúng ta cần viết chính là phải tính tổng số record. Ở đây, chúng ta sử dụng hàm count\_all() dùng để tính tổng record.

```
public function countAll(){
    return $this->db->count_all($this->_table);
}
```

Tiếp theo, chúng ta sẽ **load model** ngay trong action index, thông tin đầu tiên mà chúng ta phải cấu hình chính là tổng số record trong database thông qua phương thức **countAll** mà chúng ta vừa viết trong model **UserModel**. Sau đó, cấu hình đường dẫn cho controller & action muốn phân trang, để nạp đường dẫn một cách chính xác, chúng ta sẽ dùng hàm **base\_url()** được hỗ trợ trong helper url. Và tất nhiên chúng ta cũng phải cấu hình bao nhiêu record trong một trang, tức là phần **per\_page**. Chúng ta phải truyền biến config vào library pagination để có thể thực thi việc phân trang. Cuối cùng, chúng ta in trực tiếp hàm **create\_links()** trong controller để hiển thị kết quả phân trang, nên nhớ việc đỗ dữ liệu chỉ nên thao tác trong view.

```
public function index() {
    $this->load->model("UserModel");
    $config['total_rows'] = $this->UserModel->countAll();
    $config['base_url'] = base_url()."index.php/page/index";
    $config['per_page'] = 3;

    $this->load->library('pagination', $config);
    echo $this->pagination->create_links();
}
```

Nếu màn hình kết quả trả về như sau xem như thao tác thành công.

1 2 3 > Last ›

Để hiển thị dữ liệu thì chúng ta phải xử lý thêm vài thao tác trong model, ví dụ như dùng **result\_array()** để lấy toàn bộ record và đỗ nó ra một mảng, truyền tham số vào view & show dữ liệu ngay trong view. Tất nhiên là phải giới hạn kết quả, muốn giới hạn kết quả thì chúng ta dùng hàm **limit()** và truyền vào số record và vị trí bắt đầu.

```
public function getList($total, $start){
    $this->db->limit($total, $start);
    $query=$this->db->get("user");
    return $query->result_array();
}
```

Muốn hiển thị dữ liệu, chúng ta phải gọi phương thức **getList()** trong controller, truyền vào nó 2 tham số, số record trong một trang & vị trí bắt đầu, muốn có được vị trí bắt đầu thì ta phải get được đường dẫn (chỉ số) trang hiện hành. Trong CI, chúng ta có một helper hỗ trợ việc lấy đường dẫn và khuyến cáo không nên sử dụng phương thức **\$\_GET** vì nó không an toàn. Phương thức này là **\$this->uri->segment(tham số)**. Để có thể lấy đường dẫn một cách chính xác, ví dụ link có dạng: localhost/citest/index.php/page/index/??? thì page = controller ứng với vị trí số 1, index = action ở vị trí thứ 2, và tham số trang \$start chính là vị trí thứ 3.

```
public function index() {
    $this->load->model("UserModel");
    $config['total_rows'] = $this->UserModel->countAll();
    $config['base_url'] = base_url()."index.php/page/index";
    $config['per_page'] = 3;
```

```
$start=($this->uri->segment(3)-1)*3;
$this->load->library('pagination', $config);
$data['data']= $this->UserModel->getList($config['per_page'], $start);
$this->load->view("page_view", $data);
}
```

Nhiệm vụ của getList là lấy dữ liệu về dưới dạng một mảng chứa thông tin user và chúng ta sẽ hiển thị trong view. Chúng ta tạo file page\_view.php đơn giản có nội dung như sau:

```
echo "<pre>";
print_r($data);
echo "<pre>";
echo $this->pagination->create_links();
```

Nếu kết quả trả về như hình bên dưới thì chúng ta đã tiến hành phân trang thành công.

```
Array
(
[0] => Array
(
[id] => 1
[username] => admin
[password] => 123456
[level] => 2
) [1] => Array
(
[id] => 2
[username] => alibaba
[password] => 123456
[level] => 1
)
[2] => Array
(
[id] => 3
[username] => vicky
[password] => 123456
[level] => 1
)
)
```

1 2 3 > Last >

## 2. Các vấn đề mở rộng trong pagination

- Làm nổi bật link hiện hành, ví dụ: khi bạn click vào link 3 thì phía nó hiển thị như sau: 1 2 **3** 4 5.

```
$config['num_links'] = 2;
```

- Sử dụng link theo chỉ số trang:

```
$config['use_page_numbers'] = TRUE;
```

## Bài 8: Tìm hiểu Helper Url và Form trong Codeigniter

Trong bài này, chúng ta sẽ tìm hiểu:

- Giới thiệu về helper
- Làm việc với form helper
- Các helper thông dụng

Codeigniter Framework tách biệt 2 khái niệm, đó là helper & library.

Helper là cái gì? Nó là những function được xây dựng sẵn nhằm hỗ trợ lập trình viên trong quá trình xây dựng website. Có rất là nhiều helper được khởi tạo sẵn.

Library khác helper ở điểm nào? Library được viết dựa trên những cái class dùng để định nghĩa các phương thức và thực thi các hành động ở bên trong lớp đó, dĩ nhiên nó khác biệt với helper. Helper là những thành phần mà chúng ta có thể gọi ở bất cứ nơi đâu và tùy ý sử dụng nó, ví dụ như gọi ra list menu ngay trong view chẳng hạn. Hai helper mà các bạn sẽ sử dụng nhiều nhất chính là **form & url**.

### 1. Load Helper

Cũng giống như library, để có thể sử dụng tất cả các hàm trong các helper thì việc đầu tiên là chúng ta phải gọi chúng ra. Chúng ta nên gọi helper ngay trong constructor của controller để có thể sử dụng helper đó ở mọi action. Có 2 cách để load helper: load lần lượt từng helper hoặc load một mảng các helper.

```
$this->load->helper('url');  
$this->load->helper(array('url', 'form'));
```

### 2. Url Helper

Helper này có rất là nhiều phương thức, tuy nhiên, kích thước của nó khá nặng, vì vậy chúng ta chỉ nên dùng một số phương thức phổ biến sau đây.

- *base\_url()*: Chỉ định đường dẫn tuyệt đối, khai báo trong file config.php.
- *site\_url()*: Nó gần giống với base\_url./li>
- *anchor()*: Khai báo một đường link, ví dụ echo anchor('news/local/123', 'My News', 'title="News title"').
- *url\_title()*: Hỗ trợ Url Friendly dạng hom-nay-troi-co-mua.
- *redirect()*: Hỗ trợ chuyển sang một trang được chỉ định.

Trong CI, chúng ta sẽ phải làm việc với router và khi phải làm việc với nó thì tất cả các đường dẫn sẽ được rewrite dưới dạng thư mục hoặc file, nếu chúng ta không chỉ định ra một đường dẫn tuyệt đối thì sẽ không thể nào định nghĩa được những đường dẫn

của thư mục images & css. Tránh lạm dụng phương thức trong helper vì làm như thế sẽ khiến ứng dụng website của các bạn chạy rất là chậm, bình thường chúng ta chỉ dùng 2 phương thức redirect & base\_url để làm tốt công việc cấu hình đường dẫn tuyệt đối, chuyển trang khi thao tác xong một hành động nào đó.

### 3. Form Helper

Đây là một helper khá hay, nó giúp các bạn nhanh chóng tạo ra các form dùng để nhập liệu một cách nhanh chóng. Tuy nhiên, chúng ta cũng chỉ nên sử dụng nó tùy lúc thôi nhé, còn lại thì vẫn cứ dùng html mà tạo form bình thường.

Chúng ta sẽ tạo ra một controller tên là Form, gọi sử dụng helper url và form. Do chúng ta gọi một lần 2 helper nên chúng ta sử dụng dạng mảng. Chúng ta sẽ gọi các phương thức trong form helper ngay trong phần view. Trước khi bắt đầu ví dụ, chúng ta sẽ liệt kê những phương thức mà chúng ta sắp sử dụng.

```
<?php
class Form extends CI_Controller{
    public function __construct() {
        parent::__construct();
        $this->load->helper(array('url', 'form'));
    }

    public function index() {
        $this->load->view("form");
    }
}
?>
```

- **form\_open:** Mở thẻ form.
- **form\_close:** Đóng thẻ form.
- **form\_input:** Tạo thẻ input với type là text.
- **form\_label:** Tương đương với thẻ label trong html.
- **form\_password:** Thẻ input với type là password.
- **form\_upload:** Thẻ input với type là file
- **form\_textarea:** Thẻ textarea.
- **form\_dropdown:** Thẻ select option trong form.
- **form\_radio:** Thẻ tạo nút chọn trong form.
- **form\_submit:** Thẻ input với type là submit.
- **form\_fieldset:** Thẻ mở fieldset trong form.
- **form\_fieldset\_close:** Thẻ đóng fieldset trong form.

Có rất nhiều cách tạo ra form cũng như khai báo thông số cho nó, nhưng cách thường được sử dụng là tạo ra một mảng chứa các thông số. Ví dụ, chúng ta tạo ra một form input với name là username, value là tkpro.edu.vn thì trong CI, chúng ta viết như sau:

```
<?php
$user=array(
    "name" => "username",
    "size" => "25",
);
$pass=array(
    "name" => "pass",
    "size" => "25",
);
$email=array(
    "name" => "email",
    "size" => "25",
);
$gender1=array(
    "name" => "gender",
    "value" => "m",
    "checked" => TRUE,
);
$gender2=array(
    "name" => "gender",
    "value" => "f",
);
$opt=array(
    "1" => "Viet Nam",
    "2" => "Cambodia",
    "3" => "Malaysia",
);
$note=array(
    "name" => "note",
    "cols" => "40",
    "rows" => "5",
);
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>freetuts.net</title>
</head>

<body>
<?php
    echo form_open(base_url()."index.php/form/index");
    echo form_fieldset("Member Register");
    echo form_label("Fullname: ").form_input($user)."<br />";
    echo form_label("Password: ").form_password($pass)."<br />";
```

```
echo form_label("Email: ").form_input($email)."<br />";  
echo form_label("Gender:  
").form_radio($gender1)."Male".form_radio($gender2)."Female<br />";  
echo form_label("Country: ").form_dropdown("Country: ", $opt, 1)."<br />";  
echo form_label("Note: ").form_textarea($note)."<br />";  
echo form_label(" " ).form_submit("ok", "Register");  
echo form_fieldset_close();  
echo form_close();  
?>  
</body>  
</html>
```

Như vậy, với các tạo ra các đối tượng trên form bằng cách sử dụng form helper và truyền mảng khai báo, chúng ta có thể định nghĩa nhiều thành phần cho form, mỗi thành phần có nhiều thuộc tính khác nhau như name, id, style, class, value, ....

### 4. Các Helper thông dụng

Mình sẽ liệt kê một số helper thông dụng thường được sử dụng trong CI. Có nhiều khuyến cáo là chúng ta sử dụng một cách hạn chế và không nên lạm dụng mà thay vào đó, chúng ta nên viết mã HTML thuần sẽ dễ kiểm soát và hiệu xuất làm việc của website cũng tốt hơn.

- CAPTCHA Helper
- Date Helper
- Download Helper
- Email Helper
- HTML Helper
- Language Helper
- Security Helper
- Text Helper
- String Helper

## Bài 9: Tìm Hiểu Library Form Validation

Trong bài này bạn sẽ được học:

- Sử dụng form validation kiểm tra dữ liệu
- Xuất các thông báo lỗi ra ngoài view

Trong bài này, chúng ta sẽ tìm hiểu việc kiểm soát dữ liệu người dùng nhập vào textbox, tạo ra những tập luật để kiểm duyệt dữ liệu, ví dụ dữ liệu ô nhập bắt buộc phải là con số, email phải đúng định dạng, ... Giống như các bài trước, để có thể thao tác với bất kỳ library nào trong CI, chúng ta phải load nó ra.

### 1. Load library form validation

```
$this->load->library('form_validation')  
//gọi phương thức  
//$this->form_validation->ten_phuong_thuc()
```

Để có thể nắm bắt vấn đề tốt hơn khi thao tác với **form validation**, chúng ta sẽ liệt kê một số phương thức hay dùng. Đầu tiên là phương thức dùng để tạo ra các luật kiểm tra, đó là **set\_rules**. Phương thức này nhận vào 3 giá trị: tên của textbox, tên đối tượng xuất ra thông báo lỗi, và giá trị quan trọng nhất chính là tập luật mà bạn muốn quy định, các luật cách nhau bằng dấu |.

```
$this->form_validation->set_rules('username', 'Username', 'required')
```

- **required**: Yêu cầu phải nhập liệu không được để trống
- **matches**: Yêu cầu password phải trùng khớp nhau
- **min\_length**: Giới hạn bao nhiêu ký tự khi nhập vào
- **max\_length**: Giới hạn bao nhiêu ký tự khi nhập vào
- **numeric**: Yêu cầu trong textbox phải nhập liệu là con số
- **valid\_email**: Email nhập liệu bắt buộc phải đúng định dạng
- **xss\_clean**: Xóa XSS của input, bảo mật

CI cung cấp rất nhiều luật, tuy nhiên, ở trên mình chỉ liệt kê một số luật thường dùng trong thực tế. Ngoài ra, còn một phương thức rất là quan trọng nếu thiếu phương thức này thì xem như các luật phía trên bị vô hiệu hóa, đó là phương thức **run()**. Phương thức **run()** có nhiệm vụ kiểm tra các luật trên có hợp lệ hay không, nếu không hợp lệ thì nó sẽ trả về FALSE, tức là nó sẽ kiểm tra phương thức **set\_rules** nếu một trong số các luật mà bạn khai báo bị lỗi thì nó sẽ trả về FALSE đồng thời trả về một cái view nào đó. Chúng ta có thể xuất câu thông báo lỗi ra bằng phương thức **validation\_errors()**.

## 2. Kiểm tra nhập liệu với form validation

Tôi sẽ dùng lại controller & view của bài form helper, trong đó, chúng ta sẽ kiểm tra 3 textbox gồm fullname, password & email.

### Controller:

```
<?php  
class Form extends CI_Controller{  
    public function __construct(){  
        parent::__construct();  
        $this->load->helper(array('url', 'form'));  
    }  
  
    public function index(){  
        $this->load->view("form");  
    }  
}  
?>
```

### View

```
<?php  
$user=array(  
    "name" => "username",  
    "size" => "25",  
) ;  
$pass=array(  
    "name" => "pass",  
    "size" => "25",  
) ;  
$email=array(  
    "name" => "email",  
    "size" => "25",  
) ;  
$gender1=array(  
    "name" => "gender",  
    "value" => "m",  
    "checked" => TRUE,  
) ;  
$gender2=array(  
    "name" => "gender",  
    "value" => "f",  
) ;  
$opt=array(  
    "1" => "Viet Nam",  
    "2" => "Cambodia",  
    "3" => "Malaysia",  
) ;  
$note=array(  
    "name" => "note",  
    "cols" => "40",  
    "rows" => "5",
```

```
) ;  
?>  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
<title>freetuts.net</title>  
</head>  
  
<body>  
<?php  
    echo form_open(base_url()."index.php/form/index");  
    echo form_fieldset("Member Register");  
    echo form_label("Fullname: ").form_input($user)."  
    echo form_label("Password: ").form_password($pass)."  
    echo form_label("Email: ").form_input($email)."  
    echo form_label("Gender:  
").form_radio($gender1)."Male".form_radio($gender2)."Female  
    echo form_label("Country: ").form_dropdown("Country: ", $opt, 1)."  
    echo form_label("Note: ").form_textarea($note)."  
    echo form_label(" " ).form_submit("ok", "Register");  
    echo form_fieldset_close();  
    echo form_close();  
?>  
</body>  
</html>
```

Công việc bây giờ của chúng ta là dùng form validation để check lỗi nhập liệu, nếu người dùng đã nhấn nút submit thì chúng ta sẽ tiến hành yêu cầu họ phải thao tác theo đúng các luật mà chúng ta đã đưa ra. Do ví dụ này chỉ có một action nên việc đầu tiên là chúng ta sẽ load form validation ra, một khi đã load xong library thì chúng ta hoàn toàn có thể sử dụng được toàn bộ các phương thức của nó. Phương thức đầu tiên chúng ta sử dụng sẽ là phương thức **set\_rules()**, tên của textbox đầu tiên là fullname và đối tượng xuất ra thông báo lỗi sẽ là label Full Name. Chúng ta muốn người dùng bắt buộc phải nhập tên vào textbox, vì thế chúng ta sẽ dùng luật **required**. Chúng ta không muốn cho họ nhập vào một cái tên quá dài, chúng ta ép họ phải nhập không được ít hơn 6 ký tự, chúng ta có phương thức **min\_length[6]**. Khi sử dụng nhiều hơn một luật cho cùng một đối tượng, chúng ta phải dùng dấu | để phân cách. Để kiểm tra xem email có đúng định dạng không, chúng ta sẽ dùng luật **valid\_email**. Sau khi khai báo tập luật xong, chúng ta gọi phương thức run() kiểm tra xem các **set\_rules** có hợp lệ hay không, nếu một trong 3 luật trên bị lỗi thì ngay lập tức nó sẽ trả về là FALSE và thông báo lỗi sẽ xuất hiện bằng cách gọi phương thức **validation\_errors**.

```
<?php  
class Form extends CI_Controller{
```

```
public function __construct() {
    parent::__construct();
    $this->load->helper(array('url', 'form'));
}

public function index(){
    $this->load->library('form_validation');
    $this->form_validation->set_rules('username', 'Full Name',
'required|min_length[6]');
    $this->form_validation->set_rules('pass', 'Pass Word',
'required');
    $this->form_validation->set_rules('email', 'Email',
'required|valid_email');

    if($this->form_validation->run() == FALSE) {
        $this->load->view("form");
    }
}
?>
<?php
    echo validation_errors();
    echo form_open(base_url()."index.php/form/index");
    echo form_fieldset("Member Register");
    echo form_label("Fullname: ").form_input($user)."<br />";
    echo form_label("Password: ").form_password($pass)."<br />";
    echo form_label("Email: ").form_input($email)."<br />";
    echo form_label("Gender:");
") .form_radio($gender1)."Male".form_radio($gender2)."Female<br />";
    echo form_label("Country: ").form_dropdown("Country: ", $opt, 1)."<br />";
    echo form_label("Note: ").form_textarea($note)."<br />";
    echo form_label(" ").form_submit("ok", "Register");
    echo form_fieldset_close();
    echo form_close();
?>
```

Bây giờ, chúng ta hãy chạy link *localhost/citest/index.php/form* để kiểm tra, nếu chúng ta không nhập bất cứ thứ gì vào textbox mà nhấn nút submit thì ngay lập tức trình duyệt sẽ trả về kết quả sau.

The Full Name field is required.

The Password field is required.

The Email field is required.

Tức là yêu cầu phải nhập liệu, và nếu ở textbox fullname chúng ta nhập ít hơn 6 ký tự và email chúng ta điền không đúng định dạng thì thông báo lỗi sẽ trả về như sau.

The Full Name field must be at least 6 characters in length.

The Email field must contain a valid email address

Việc kiểm tra nhập liệu với form validation thì khá đơn giản, còn một vấn đề đặt ra ở đây là, nếu chúng ta muốn việt hóa các câu thông báo lỗi khi xuất ra thì làm thế nào?

### 3. Việt hóa language form validation

Bước Việt hóa thật ra là chúng ta sửa code trong phần system của CI. Rất đơn giản, tại folder system, các bạn tìm đến folder language, mở nó ra và tìm file form\_validation\_lang.php, mở file này ra và dịch các đoạn tiếng Anh ở bên trong bên. Chú ý ký hiệu %s chính là tên mà chúng ta đặt khi nó xuất ra thông báo lỗi.

```
<?php
$lang['required']      = "The %s field is required.";
$lang['isset']          = "The %s field must have a value.";
$lang['valid_email'] = "The %s field must contain a valid email address.";
$lang['valid_emails']= "The %s field must contain all valid email
addresses.";
$lang['valid_url']      = "The %s field must contain a valid URL.";
$lang['valid_ip']        = "The %s field must contain a valid IP.";
$lang['min_length']= "The %s field must be at least %s characters in
length.";
$lang['max_length']= "The %s field can not exceed %s characters in length.";
$lang['exact_length']="The %s field must be exactly %s characters in
length.";
$lang['alpha']    = "The %s field may only contain alphabetical characters.";
$lang['alpha_numeric']= "The %s field may only contain alpha-numeric
characters.";
$lang['alpha_dash'] = "The %s field may only contain alpha-numeric
characters, underscores, and dashes.";
$lang['numeric']     = "The %s field must contain only numbers.";
$lang['is_numeric']  = "The %s field must contain only numeric characters.";
$lang['integer']      = "The %s field must contain an integer.";
$lang['regex_match'] = "The %s field is not in the correct format.";
$lang['matches']       = "The %s field does not match the %s field.";
$lang['is_unique']    = "The %s field must contain a unique value.";
$lang['is_natural']   = "The %s field must contain only positive numbers.";
$lang['is_natural_no_zero']= "The %s field must contain a number greater
than zero.";
$lang['decimal']       = "The %s field must contain a decimal number.";
$lang['less_than']      = "The %s field must contain a number less than %s.";
$lang['greater_than']="The %s field must contain a number greater than %s.";
?>
```

## Bài 10: Tìm hiểu library upload trong codeigniter

Trong bài này bạn sẽ được học:

- Giới thiệu về Library upload
- Thao tác & làm việc với nó

### 1. Giới thiệu library upload

Trong CI, với **Library upload** chúng ta sẽ làm được những việc sau:

- Hỗ trợ định dạng file được chỉ định khi upload.
- Tùy chỉnh độ rộng & cao tấm hình được phép upload một cách dễ dàng.
- Giới hạn dung lượng file được upload.
- Hỗ trợ chúng ta resize ảnh theo nhu cầu.
- Đóng dấu bản quyền watermark với định dạng text hoặc chèn logo lên bất kể vị trí nào trong tấm hình.

Với 5 ưu điểm trên, chúng ta thấy rằng chức năng upload file, đặc biệt là upload file ảnh, đáp ứng hầu hết các yêu cầu upload của một ứng dụng web.

### 2. Cấu hình library upload

Cũng như các library khác, để có thể thao tác với nó thì chúng ta phải gọi nó ra bằng cú pháp sau:

```
$this->load->library('upload')
```

Sau đó, chúng ta cần cung cấp một số thông tin cấu hình:

```
$config['upload_path'] = './uploads/';  
$config['allowed_types'] = 'gif|jpg|png';  
$config['max_size'] = '100';  
$config['max_width'] = '1024';  
$config['max_height'] = '768';
```

Với các thông tin trên thì các bạn có thể dễ dàng nhận ra rằng chúng ta đang cấu hình đường dẫn, định dạng hình ảnh được phép upload, kích thước tấm hình, cũng như dung lượng tấm hình được phép upload. Sau khi cấu hình xong các thông tin trên, chúng ta sẽ có câu lệnh để thực thi việc upload như sau.

```
$this->upload->do_upload('Ten file')
```

Và tất nhiên là sau khi upload xong thì chúng ta có thể kiểm tra xem mình upload có thành công hay không. Để lấy về thông tin của tập tin vừa upload, chúng ta sử dụng lệnh sau:

```
$this->upload->data();
```

Trong quá trình upload file nếu tấm hình của chúng ta bị lỗi cũng như có một số chuẩn mực không hợp lệ thì cần xuất ra thông báo lỗi để người dùng biết, để hiển thị thông báo lỗi, chúng ta sử dụng:

```
$this->upload->display_errors();
```

Như vậy thì chúng ta chỉ cần ghi nhớ 3 phương thức và một số thông tin cấu hình ở phía trên thì chúng ta hoàn thành chức năng upload hình ảnh hay tập tin nói chung.

### 3. Thực hành upload file từ máy tính lên web server

Trong ví dụ này chúng ta sử dụng helper form & url nên chúng ta sẽ load nó ra ngay tại constructor. Tạo thêm một folder uploads ngang cấp với folder application, chúng ta cũng tạo ra action index và load view có chứa form để upload file.

```
class Upload extends CI_Controller{
    public function __construct(){
        parent::__construct();
        $this->load->helper(array("form", "url"));
    }

    public function index(){
        $this->load->view("upload_view");
    }
}
```

Tạo trong folder view một file có tên là upload\_view có nội dung như sau.

```
<?php
$upload=array(
    "name" => "img",
    "size" => "25",
);
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Freetuts.net</title>
</head>
<body>
<?php
    echo form_open_multipart(base_url()."index.php/upload/doupload");
    echo form_label("Avartar: ").form_upload($upload)."<br />";
    echo form_label(" ").form_submit("ok", "Upload");
    echo form_close();
?>
</body>
</html>
```

Hàm **form\_open\_multipart** tạo ra form có thuộc tính *multipart/form-data* trong html, tạo đối tượng file (**form\_upload**) với name là img, đối tượng nút bấm submit (**form\_submit**) có name là ok. Form này có action trả đến controller là upload & action

sẽ là doupload. Thực hiện xong 2 thao tác này, sau khi mở trang web thì các bạn có thể view source để xem phần hiển thị html của nó.

Chúng ta sẽ tạo ra action doupload và check xem người dùng đã nhấn nút submit hay chưa bằng thao tác `$this->input->post('ok')`. Nếu nó thỏa điều này thì chúng ta mới tiến hành cho thông tin cấu hình upload file vào được. Việc check như thế này sẽ kiểm tra xem người dùng có chọn hình chưa, nếu chưa chọn hình mà nhấn nút submit thì trình duyệt sẽ trả về thông báo lỗi.

Khi cấu hình các thông tin xong, chúng ta sẽ load cái library và cung cấp thông tin cấu hình này (biến `$config`). Sau đó, chúng ta gọi method để thực thi thao tác upload file. Upload xong thì chúng ta hiển thị toàn bộ thông tin của tấm ảnh nếu việc upload là thành công. Ngược lại, nếu quá trình upload thất bại thì chúng ta sẽ xuất ra câu báo lỗi, khi đó chúng ta phải tạo ra một biến để đổ câu thông báo lỗi sang view (`$data['errors'] = $this->upload->display_errors();`). Vậy ở phương thức index, chúng ta phải khai báo biến `$data['errors']` bằng rỗng để nó ẩn câu thông lỗi, chỉ khi nào có lỗi thì mới hiển thị ra. Nếu errors khác rỗng thì chúng ta mới xuất thông báo lỗi. Bay giờ, chúng ta hãy trình duyệt và gõ [localhost/citest/index.php/upload](http://localhost/citest/index.php/upload) để kiểm tra.

```
public function index() {
    $data['errors'] = '';
    $this->load->view("upload_view", $data);
}

public function doupload(){
    if($this->input->post("ok")){
        $config['upload_path'] = './uploads/';
        $config['allowed_types'] = 'gif|jpg|png';
        $config['max_size'] = '900';
        $config['max_width'] = '1024';
        $config['max_height'] = '768';
        $this->load->library("upload", $config);
        if($this->upload->do_upload("img")){
            echo 'Upload Ok';
            $check = $this->upload->data();
            echo "<pre>";
            print_r($check);
            echo "</pre>";
        }else{
            $data['errors'] = $this->upload->display_errors();
            $this->load->view("upload_view", $data);
        }
    }
}
```

Nếu upload thành công thì sẽ xuất ra các thông số như sau:

```
Upload Ok
Array
(

```

```
[file_name] => Koala.jpg
[file_type] => image/jpeg
[file_path] => D:/freetuts/www/cietest/uploads/
[full_path] => D:/freetuts/www/cietest/uploads/Koala.jpg
[raw_name] => Koala
[orig_name] => Koala.jpg
[client_name] => Koala.jpg
[file_ext] => .jpg
[file_size] => 762.53
[is_image] => 1
[image_width] => 1024
[image_height] => 768
[image_type] => jpeg
[image_size_str] => width="1024" height="768"
)
```

Để kiểm tra lỗi, chúng ta sẽ chọn thử một file .doc nào đó tiến hành nhấn nút submit xem kết quả.

The filetype you are attempting to upload is not allowed.

Kết quả là file này không đúng định dạng nên không được phép upload. Với việc cấu hình định dạng file cho phép upload thì các bạn đã phần nào hạn chế được việc hacker upload file shell (file thực thi) hoặc các mã độc khác lên website của mình.

## 4. Mã nguồn upload tập tin lên server

### **Controller Upload:**

```
<?php
class Upload extends CI_Controller{
    public function __construct(){
        parent::__construct();
        $this->load->helper(array("form", "url"));
    }

    public function index(){
        $data['errors'] = '';
        $this->load->view("upload_view", $data);
    }

    public function doupload(){
        if($this->input->post("ok")){
            $config['upload_path'] = './uploads/';
            $config['allowed_types'] = 'gif|jpg|png';
            $config['max_size'] = '900';
            $config['max_width'] = '1024';
            $config['max_height'] = '768';
            $this->load->library("upload", $config);
            if($this->upload->do_upload("img")){

```

```
        echo 'Upload Ok';
        $check = $this->upload->data();
        echo "<pre>";
        print_r($check);
        echo "</pre>";
    }else{
        $data['errors'] = $this->upload->display_errors();
        $this->load->view("upload_view", $data);
    }
}
}
```

### View:

```
<?php
$upload=array(
    "name" => "img",
    "size" => "25",
);
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Freetuts.net</title>
</head>

<body>
<?php
    if($errors != "") {
        echo $errors;
    }
    echo form_open_multipart(base_url()."index.php/upload/douupload");
    echo form_label("Avartar: ").form_upload($upload)."<br />";
    echo form_label(" ").form_submit("ok", "Upload");
    echo form_close();
?>
</body>
</html>
```

## Bài 11: Tìm Hiểu Library Image Trong Codeigniter

Trong bài này bạn sẽ được học:

- Giới thiệu về Library Image
- Thao tác & làm việc với nó

Trong bài trước, chúng ta đã hoàn thành việc upload hình ảnh trong CI, dựa vào library upload. Chúng ta có thể upload hình ảnh từ máy tính lên web server, cũng như có thể tạo ra những ràng buộc về kích thước hình ảnh, kiểm tra file có hợp lệ hay không, dung lượng tệp hình cho phép là bao nhiêu MB, .... Nhưng như vậy là chưa đủ, khi các bạn bắt tay vào xây dựng hoàn chỉnh một ứng dụng web thì sẽ phát sinh một số vấn đề như làm thế nào để tạo những cái hình nhỏ hơn, thay thế cho hình đại diện trong từng sản phẩm (gọi là các hình **thumbnail**). Để làm được điều này thì chúng ta phải tìm hiểu khái niệm **library image** trong CI.

### 1. Giới thiệu library Image

Với library image thì chúng ta có thể làm được những điều sau:

- Resize Image: Từ hình mặc định được upload lên sẽ cắt nhỏ thành hình thumbnail.
- Image Crop: Làm việc với thao tác này tức là mình sẽ cắt hình.
- Image Rotate: Làm việc với thao tác là xoay hình ảnh.
- Image Watermark: Chèn logo hoặc đoạn text bất kì vào tấm hình.

Thực tế thì 2 phương thức mà chúng ta thường sử dụng nhiều nhất chính là **resize & watermark**, đối với việc cắt hoặc xoay hình thì mình nghĩ photoshop sẽ làm tốt việc này hơn so với việc chúng ta dùng PHP để thao tác. Và trong bài này chúng ta sẽ tìm hiểu thao tác resize.

### 2. Cấu hình library image

Đầu tiên thì tất nhiên là sẽ phải gọi library với cú pháp.

```
$this->load->library('image_lib');
```

Và thiết lập một số thông tin cấu hình như sau.

```
$config['image_library'] = 'gd2';
$config['source_image'] = '/path/to/image/mypic.jpg';
$config['create_thumb'] = TRUE;
$config['maintain_ratio'] = TRUE;
$config['width'] = 75;
$config['height'] = 50;
```

Thuộc tính `$config['image_library']` chính là một dạng library image đặc biệt và chúng ta có rất nhiều library giống như thế, ví dụ **GD**, **GD2**, **ImageMagick**, **NetPBM**. Tuy nhiên thông dụng và phổ biến nhất vẫn là GD2, và mặc định thì library sẽ có cấu hình là GD2. Ngoài ra, chúng ta còn phải chỉ ra đường dẫn lưu tấm hình mà chúng ta upload lên.

Sau khi cấu hình đầy đủ các thông tin cần thiết, các bạn phải tiến hành gọi lệnh thực thi thao tác resize hình ảnh gì đó bằng cú pháp sau.

```
$this->image_lib->resize();
```

Và sau đây chúng ta sẽ có một ví dụ nhỏ, giúp các bạn nắm kỹ hơn về thư viện image.

### 3. Thực hành resize với library image

Chúng ta sẽ sử dụng lại controller upload đã làm ở bài trước, vì để có thể resize được thì bắt buộc phải upload được tấm hình lên web server.

```
<?php  
class Upload extends CI_Controller{  
    public function __construct(){  
        parent::__construct();  
        $this->load->helper(array("form", "url"));  
    }  
  
    public function index(){  
        $data['errors'] = '';  
        $this->load->view("upload_view", $data);  
    }  
  
    public function douupload(){  
        if($this->input->post("ok")){  
            $config['upload_path'] = './uploads/';  
            $config['allowed_types'] = 'gif|jpg|png';  
            $config['max_size'] = '900';  
            $config['max_width'] = '1024';  
            $config['max_height'] = '768';  
            $this->load->library("upload", $config);  
            if($this->upload->do_upload("img")){  
                echo 'Upload Ok';  
                $check = $this->upload->data();  
                echo "<pre>";  
                print_r($check);  
                echo "</pre>";  
            } else{  
                $data['errors'] = $this->upload->display_errors();  
                $this->load->view("upload_view", $data);  
            }  
        }  
    }  
}
```

Khi đã upload thành công, chúng ta sẽ có một số thao tác như resize hình ảnh. Để làm việc này, đầu tiên chúng ta load library image ra, sau đó tiến hành cấu hình thông tin cho nó. Do chúng ta cần phải có tên gốc của tấm hình mới có thể resize được, nên chúng ta dùng phương thức `$check = ['file_name']` nối chuỗi phía sau đường dẫn tấm hình. Sau đó, chúng ta sẽ cho độ rộng tấm hình khi resize sẽ là 150 và chiều cao của nó là 120, sau khi hoàn tất các thông số trên thì sẽ nạp tất cả thông tin này bằng cú pháp `$this->image_lib->initialize($config)`. Sau đó, chúng ta cần phải gọi lệnh thực thi thao tác resize với cú pháp `$this->image_lib->resize()`. Hàm này sẽ giúp chúng ta cắt hình lớn sang hình nhỏ và thỏa mãn toàn bộ yêu cầu mà chúng ta đã khai báo ở phần thông tin cấu hình.

```
public function doUpload(){
    if($this->input->post("ok")){
        $config['upload_path'] = './uploads/';
        $config['allowed_types'] = 'gif|jpg|png';
        $config['max_size'] = '900';
        $config['max_width'] = '1024';
        $config['max_height'] = '768';
        $this->load->library("upload", $config);
        if($this->upload->do_upload("img")){
            echo 'Upload Ok';
            $check = $this->upload->data();
            echo "<pre>";
            print_r($check);
            echo "</pre>";
            $this->load->library("image_lib");
            $config['image_library'] = 'gd2';
            $config['source_image'] =
            './uploads/'.$check['file_name'];
            $config['create_thumb'] = TRUE;
            $config['maintain_ratio'] = TRUE;
            $config['width'] = 150;
            $config['height'] = 120;
            $this->image_lib->initialize($config);
            $this->image_lib->resize();
        }else{
            $data['errors'] = $this->upload->display_errors();
            $this->load->view("upload_view", $data);
        }
    }
}
```

Kiểm tra bằng cách chạy link `localhost/citest/index.php/upload`, sau đó upload bất kỳ tấm hình nào đó và vào folder uploads xem điều gì sẽ xảy ra nhé. Ví dụ ở đây chúng ta upload tấm hình `lighthou.jpg`, và lúc kiểm tra tôi thấy ngoài hình gốc là `lighthou.jpg` thì còn xuất hiện thêm file `lighthou_thumb.jpg`, nó sẽ dựa vào độ rộng hoặc độ cao của tấm hình và xem xét đổi khi thực hiện thao tác resize.

Lý do tại sao chúng ta phải sử dụng hình thumbnail? Đó là vì nếu một trang web mà chỉ sử dụng tám hình gốc ban đầu sẽ làm cho website load rất chậm, dung lượng của hình gốc là rất nặng, và chúng ta có hàng chục tám hình cho sản phẩm, mà sản phẩm nào cũng phải load hình ảnh gốc thì rõ ràng website sẽ load rất chậm. Đó là lý do chúng ta buộc phải sử dụng hình ảnh thumbnail để cải thiện hiệu quả tải website.

## 4. Full code resize image

### Controller

```
<?php  
class Upload extends CI_Controller{  
    public function __construct(){  
        parent::__construct();  
        $this->load->helper(array("form", "url"));  
    }  
  
    public function index(){  
        $data['errors'] = '';  
        $this->load->view("upload_view", $data);  
    }  
  
    public function douupload(){  
        if($this->input->post("ok")){  
            $config['upload_path'] = './uploads/';  
            $config['allowed_types'] = 'gif|jpg|png';  
            $config['max_size'] = '900';  
            $config['max_width'] = '1024';  
            $config['max_height'] = '768';  
            $this->load->library("upload", $config);  
            if($this->upload->do_upload("img")){  
                echo 'Upload Ok';  
                $check = $this->upload->data();  
                echo "<pre>";  
                print_r($check);  
                echo "</pre>";  
                $this->load->library("image_lib");  
                $config['image_library'] = 'gd2';  
                $config['source_image'] =  
                    './uploads/'.$check['file_name'];  
                $config['create_thumb'] = TRUE;  
                $config['maintain_ratio'] = TRUE;  
                $config['width'] = 150;  
                $config['height'] = 120;  
                $this->image_lib->initialize($config);  
                $this->image_lib->resize();  
            }else{  
                $data['errors'] = $this->upload->display_errors();  
                $this->load->view("upload_view", $data);  
            }  
        }  
    }  
}
```

```
        }
    }
}
```

### View

```
<?php
$upload=array(
    "name" => "img",
    "size" => "25",
);
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Freetuts.net</title>
</head>

<body>
<?php
    if($errors != "") {
        echo $errors;
    }
    echo form_open_multipart(base_url()."index.php/upload/douupload");
    echo form_label("Avartar: ").form_upload($upload)."<br />";
    echo form_label(" ").form_submit("ok", "Upload");
    echo form_close();
?
</body>
</html>
```



## Bài 12: Đóng dấu watermark image trong CodeIgniter

Trong bài này chúng ta sẽ cùng tìm hiểu một tính năng trong thư viện image, đó chính là watermark (đóng dấu ảnh). Các bước chính trong quá trình upload & đánh dấu ảnh vẫn là 3 công việc: upload hình ảnh, resize ảnh, watermark cho tấm ảnh đó.

Lý do tại sao chúng ta cần phải có watermark cho tấm hình? Đó là vì khi bạn upload một tấm hình nào đó là do công sức của chính tay các bạn làm ra, và không muốn nó bị người khác copy một cách dễ dàng thì chúng ta cần phải bảo hộ tấm hình bằng một logo nào đó, hoặc chèn text để lại địa chỉ website của các bạn. Làm điều đó chỉ để bảo vệ quyền sở hữu trí tuệ của các bạn.

### 1. Cấu hình watermark image

Với watermark do CI cung cấp thì chúng ta có 2 giải pháp, đó là đóng dấu bằng text hoặc chèn logo lên hình. Trong bài này, chúng ta sẽ tìm hiểu cách chèn một trong hai kiểu đóng dấu này.

Sau đây là những thông tin cần thiết cho việc cấu hình để đóng dấu theo dạng text, mình sẽ giải thích một số giá trị quan trọng như là kiểu text, màu sắc của text, vị trí text sẽ hiển thị trên tấm hình, kiểu font cho text, và nếu bạn muốn chọn font khác cho text thì vào trỏ đường dẫn đến font mà bạn lựa chọn. Chúng ta chỉ định vị trí cũng như canh lề cho watermark trên tấm hình. Trong đoạn mã bên dưới, chúng ta chỉ định vị trí của watermark là nằm ở cuối tấm hình và được canh giữa. Một thông tin cấu hình nữa là chỉnh khoảng cách (giá trị padding) nên là 0. Sau khi khai báo hoàn tất thì chúng ta tiến hành thực thi thao tác watermark bằng lệnh `$this->image_lib->watermark()`.

```
$config['wm_text'] = 'Copyright 2014 - freetuts.net';
$config['wm_type'] = 'text';
$config['wm_font_path'] = './system/fonts/textb.ttf';
$config['wm_font_size'] = '30';
$config['wm_font_color'] = 'ffff00';
$config['wm_vrt_alignment'] = 'bottom';
$config['wm_hor_alignment'] = 'center';
$config['wm_padding'] = '0';
$this->image_lib->initialize($config);
$this->image_lib->watermark();
```

#### Full Code:

```
public function doupload(){
    if($this->input->post("ok")){
        $config['upload_path'] = './uploads/';
        $config['allowed_types'] = 'gif|jpg|png';
        $config['max_size'] = '900';
        $config['max_width'] = '1024';
        $config['max_height'] = '768';
```

```
$this->load->library("upload", $config);
if($this->upload->do_upload("img")){
    echo 'Upload Ok';
    $check = $this->upload->data();
    echo "<pre>";
    print_r($check);
    echo "</pre>";
    $this->load->library("image_lib");
    /*$config['image_library'] = 'gd2';
    $config['source_image'] =
'./uploads/'.$check['file_name'];
    $config['create_thumb'] = TRUE;
    $config['maintain_ratio'] = TRUE;
    $config['width']      = 150;
    $config['height']     = 120;*/

    $config['wm_text'] = 'Copyright 2014 - freetuts.net';
    $config['wm_type'] = 'text';
    $config['wm_font_path'] = './system/fonts/texb.ttf';
    $config['wm_font_size'] = '30';
    $config['wm_font_color'] = 'ffff00';
    $config['wm_vrt_alignment'] = 'bottom';
    $config['wm_hor_alignment'] = 'center';
    $config['wm_padding'] = '0';
    $this->image_lib->initialize($config);
    $this->image_lib->watermark();
} else{
    $data['errors'] = $this->upload->display_errors();
    $this->load->view("upload_view", $data);
}
}
```

Chạy link `localhost/citest/index.php/upload` test bằng cách upload tấm hình và kiểm tra hình trong folder có add text màu vàng chưa, nếu có thì các bạn đã thành công, còn chưa thấy tức là các bạn khai báo sai ở chỗ nào đó, xem kỹ code hoặc copy code của bài giảng bỏ vào chạy thử. Tiếp theo chúng ta sẽ tìm hiểu và cấu hình chèn logo cho watermark. Chúng ta tham khảo đoạn mã cấu hình bên dưới, trong đó, kiểu type sẽ là `$config['wm_type'] = 'overlay'` tức là kiểu định dạng chèn logo, `$config['wm_overlay_path'] = './uploads/logo.png'` chính là đường dẫn chứa logo mà chúng ta sẽ chèn vào tấm hình, `$config['wm_opacity'] = '50'` độ tương phản màu sắc của tấm hình. Logo sẽ xuất hiện ở dưới cùng tấm hình và nó sẽ nằm ở bên phải, tương tự như chèn text vào hình.

```
$config['wm_type'] = 'overlay';
$config['wm_overlay_path'] = './uploads/logo.png';
$config['wm_vrt_alignment'] = 'bottom';
$config['wm_hor_alignment'] = 'right';
$config['wm_padding'] = '0';
$config['wm_opacity'] = '50';
```

### Full Code:

```
public function doupload(){
    if($this->input->post("ok")){
        $config['upload_path'] = './uploads/';
        $config['allowed_types'] = 'gif|jpg|png';
        $config['max_size'] = '900';
        $config['max_width'] = '1024';
        $config['max_height'] = '768';
        $this->load->library("upload", $config);
        if($this->upload->do_upload("img")){
            echo 'Upload Ok';
            $check = $this->upload->data();
            echo "<pre>";
            print_r($check);
            echo "</pre>";
            $this->load->library("image_lib");

            /*$config['image_library'] = 'gd2';
            $config['source_image'] = './uploads/'.$check['file_name'];
            $config['create_thumb'] = TRUE;
            $config['maintain_ratio'] = TRUE;
            $config['width'] = 150;
            $config['height'] = 120; */
            $config['wm_type'] = 'overlay';
            $config['wm_overlay_path'] = './uploads/logo.png';
            $config['wm_vrt_alignment'] = 'bottom';
            $config['wm_hor_alignment'] = 'right';
            $config['wm_padding'] = '0';
            $config['wm_opacity'] = '50';
            $this->image_lib->initialize($config);
            $this->image_lib->watermark();
        }else{
            $data['errors'] = $this->upload->display_errors();
            $this->load->view("upload_view", $data);
        }
    }
}
```

## 2. Vấn đề mở rộng & tối ưu trong library image

Sau khi tiến hành resize xong thì chúng ta phải xóa các thông tin mà chúng ta đã cấu hình ở phía trên, library image có cung cấp cho chúng ta câu lệnh [`\$this->image\_lib->clear\(\)`](#). Sau khi clear xong thì chúng ta phải hủy luôn bằng hàm `unset($config)`, vì chúng sẽ được cấu hình lại ở phần watermark. Do cả resize lẫn watermark đều phải sử dụng lại đường dẫn gốc của tấm hình nên chúng ta sẽ copy phần đường dẫn ở resize đưa xuống chỗ watermark, và điều quan trọng mà các bạn cần lưu ý là khi chúng ta bắt đầu kết hợp resize & watermark chỉ trong một lần upload, thì chúng ta phải cho `$config['create_thumb'] = FALSE` ngay tại vị trí cấu hình của watermark. Vì nếu không sửa thành FALSE thì chúng ta sẽ không thể làm một lúc cả 2 thao tác được.

**Full Code resize & watermark:**

```
public function doupload(){
    if($this->input->post("ok")){
        $config['upload_path'] = './uploads/';
        $config['allowed_types'] = 'gif|jpg|png';
        $config['max_size'] = '900';
        $config['max_width'] = '1024';
        $config['max_height'] = '768';
        $this->load->library("upload", $config);
        if($this->upload->do_upload("img")){
            echo "Upload thanh cong";
            echo "<pre>";
            print_r($this->upload->data());
            echo "</pre>";
            $check=$this->upload->data();
            $this->load->library("image_lib");
            $config['image_library'] = 'gd2';
            $config['source_image'] = './uploads/'.$check['file_name'];
            $config['create_thumb'] = TRUE;
            $config['maintain_ratio'] = TRUE;
            $config['width'] = 150;
            $config['height'] = 120;
            $this->image_lib->initialize($config);
            $this->image_lib->resize();
            $this->image_lib->clear();
            unset($config);

            $config['source_image'] = './uploads/'.$check['file_name'];
            $config['create_thumb'] = FALSE;
            $config['wm_type'] = 'overlay';
            $config['wm_overlay_path'] = './uploads/logo.png';
            $config['wm_vrt_alignment'] = 'bottom';
            $config['wm_hor_alignment'] = 'right';
            $config['wm_padding'] = '0';
            $config['wm_opacity'] = '50';
            $this->image_lib->initialize($config);
            $this->image_lib->watermark();
        }else{
            $data['errors'] = $this->upload->display_errors();
            $this->load->view("upload_view", $data);
        }
    }
}
```

## Bài 13: Tìm Hiểu Helper Date Trong CodeIgniter

Trong bài này bạn sẽ được học:

- Giới thiệu về helper date.
- Các hàm thông dụng.

Hôm nay chúng ta sẽ tiếp tục tìm hiểu sâu hơn về các **helper** mà CI đã cung cấp. Bài này chỉ tập trung vào các hàm thông dụng của **helper date**.

### 1. Giới thiệu về helper date

Khi các bạn bắt tay xây dựng hoàn chỉnh một website thì ít nhất chúng ta phải xử lý một số chức năng có liên quan đến thời gian & ngày tháng. Đối với PHP thuần thì chúng ta cũng có vài hàm xử lý vấn đề này, nhưng chúng ta đang làm việc trên nền tảng framework CI thì helper date của CI sẽ cung cấp cho chúng ta đầy đủ các hàm để xử lý các vấn đề về thời gian.

### 2. Cấu hình helper date

Để sử dụng được toàn bộ các hàm bên trong nó, việc đầu tiên vẫn phải load helper vào **controller** bằng cú pháp sau.

```
$this->load->helper('date');
```

### 3. Các hàm thông dụng trong helper date

#### *now()*

Hàm này trả về thời gian hiện tại trong hệ thống Unix. Kiểu thời gian trả về có thể là thời gian local của vị trí đặt Server của hệ thống hoặc thời gian chuẩn GMT, chúng ta có thể thiết lập kiểu trả về bằng cách cấu hình lại ở vị trí `$config['time_reference']` trong file `config.php`. Nếu thiết lập giá trị nó là local thì hàm `now()` này không khác gì hàm `time()` trong php thuần.

#### *mdate()*

Hàm này tương đương với hàm `date()` của php thuần, chỉ khác là mỗi thành phần phải bắt đầu bằng prefix `%`.

#### *Ví dụ:*

```
$dates = "Nam: %Y Thang: %m Ngay: %d - %h:%i %a";
$time = time();
echo mdate($dates, $time);
```

### *days\_in\_month()*

Hàm này giúp chúng ta đếm số ngày trong một tháng, ví dụ nếu các bạn muốn biết trong tháng 5 có bao nhiêu ngày thì chúng ta có cú pháp sau.

```
echo days_in_month(05, 2014);
```

### *timezone\_menu()*

Hàm này giúp chúng ta có được danh sách toàn bộ múi giờ thế giới với cú pháp sau.

```
echo timezone_menu('UM8');
```

Danh sách múi giờ trên thế giới.

- *UM12 - (UTC - 12:00) Eniwetok, Kwajalien*
- *UM11 - (UTC - 11:00) Nome, Midway Island, Samoa*
- *UM10 - (UTC - 10:00) Hawaii*
- *UM9 - (UTC - 9:00) Alaska*
- *UM8 - (UTC - 8:00) Pacific Time*
- *UM7 - (UTC - 7:00) Mountain Time*
- *UM6 - (UTC - 6:00) Central Time, Mexico City*
- *UM5 - (UTC - 5:00) Eastern Time, Bogota, Lima, Quito*
- *UM4 - (UTC - 4:00) Atlantic Time, Caracas, La Paz*
- *UM25 - (UTC - 3:30) Newfoundland*
- *UM3 - (UTC - 3:00) Brazil, Buenos Aires, Georgetown, Falkland Is.*
- *UM2 - (UTC - 2:00) Mid-Atlantic, Ascension Is., St Helena*
- *UM1 - (UTC - 1:00) Azores, Cape Verde Islands*
- *UTC - (UTC) Casablanca, Dublin, Edinburgh, London, Lisbon, Monrovia*
- *UP1 - (UTC + 1:00) Berlin, Brussels, Copenhagen, Madrid, Paris, Rome*
- *UP2 - (UTC + 2:00) Kaliningrad, South Africa, Warsaw*
- *UP3 - (UTC + 3:00) Baghdad, Riyadh, Moscow, Nairobi*
- *UP25 - (UTC + 3:30) Tehran*
- *UP4 - (UTC + 4:00) Abu Dhabi, Baku, Muscat, Tbilisi*
- *UP35 - (UTC + 4:30) Kabul*
- *UP5 - (UTC + 5:00) Islamabad, Karachi, Tashkent*
- *UP45 - (UTC + 5:30) Bombay, Calcutta, Madras, New Delhi*
- *UP6 - (UTC + 6:00) Almaty, Colombo, Dhaka*
- *UP7 - (UTC + 7:00) Bangkok, Hanoi, Jakarta*
- *UP8 - (UTC + 8:00) Beijing, Hong Kong, Perth, Singapore, Taipei*
- *UP9 - (UTC + 9:00) Osaka, Sapporo, Seoul, Tokyo, Yakutsk*
- *UP85 - (UTC + 9:30) Adelaide, Darwin*
- *UP10 - (UTC + 10:00) Melbourne, Papua New Guinea, Sydney, Vladivostok*
- *UP11 - (UTC + 11:00) Magadan, New Caledonia, Solomon Islands*
- *UP12 - (UTC + 12:00) Auckland, Wellington, Fiji, Marshall Island*

Với helper date, chúng ta có được rất nhiều hàm dùng để xử lý vấn đề thời gian. Ở đây, mình chỉ liệt kê một số hàm thông dụng, nếu các bạn muốn hiểu sâu hơn thì mở user guide của CI và xem thêm.

## Bài 14: Tìm Hiểu Helper Text Trong Codeigniter

Trong bài này bạn sẽ được học:

- Giới thiệu về helper text.
- Các hàm thông dụng.

Việc xử lý văn bản trong website là việc được thực hiện thường xuyên, **helper text** trong CI cung cấp cho chúng ta rất nhiều hàm để làm việc này.

Đầu tiên để chúng ta cần phải tiến hành khai báo helper bằng cú pháp sau.

```
$this->load->helper('text');
```

Sau khi load xong, chúng ta sẽ tìm hiểu từng hàm thông dụng có sẵn trong helper text.

### **word\_limiter()**

Hàm này xóa bỏ các chuỗi với số lượng từ được giữ lại và bổ sung dấu 3 chấm (...) thay cho phần bị xóa bỏ.

```
$this->load->helper("text");
$string = "Day la vi du ve word_limiter";
$string = word_limiter($string, 4);
echo $string; // Ket qua tra ve la: Day la vi du...
```

### **character\_limiter()**

Giống hàm word\_limit() nhưng tính theo số ký tự chứ không phải số từ.

### **word\_censor()**

Hàm này dùng để ẩn đi một số từ xấu, ví dụ như hạn chế người dùng nói tục trên website của bạn. Tham số đầu tiên là chuỗi của văn bản gốc, tham số thứ hai là mảng các từ sẽ bị ẩn đi, tham số thứ ba là từ/ ký hiệu thay thế, nếu không nó sẽ hiển thị mặc định là ####, ở đây chúng ta sử dụng ký tự \*\*\* cho tham số thứ ba.

```
$string = "Got damn it shit";
$badword = array('damn', 'fuck', 'shit', 'funny');
echo $string = word_censor($string, $badword, '***');
```

### **highlight\_phrase()**

Hàm này sẽ định nghĩa màu sắc cho một phần trong đoạn văn bản được quy định. Hàm này có 4 tham số: Tham số thứ nhất là phần văn bản gốc, Tham số thứ hai là đoạn văn bản muốn tô màu, Tham số thứ ba & thứ tư sẽ chứa phần đóng mở html định dạng màu sắc cho văn bản.

```
$string = "Trang web của trung tâm đào tạo lập trình TKPro";
echo $string = highlight_phrase($string, "TKPro", "<span
style='color:red;'>", "</span>");
//Kết quả: Trang web của trung tâm đào tạo lập trình TKPro.
```

## Bài 15: Tìm Hiểu Helper Language Trong Codeigniter

Trong bài này bạn sẽ được học:

- Giới thiệu về helper language.
- Các hàm thông dụng.

Việc tùy chỉnh ngôn ngữ trên website rất là cần thiết trong quá trình toàn cầu hóa, nếu các bạn đang làm việc với php thuần thì việc này sẽ khá vất vả. Tuy nhiên, với CI thì mọi chuyện trở nên dễ dàng, bộ helper language cung cấp cho chúng ta đầy đủ các hàm thông dụng.

### 1. Giới thiệu helper language

Website song ngữ là gì? tức là nó có nhiều hơn một ngôn ngữ, ngoài tiếng Việt thì nó có thêm các ngôn ngữ khác như tiếng Anh, tiếng Pháp chẳng hạn. Chúng ta sẽ tìm hiểu helper này qua một ví dụ cụ thể như sau.

### 2. Cấu hình & sử dụng helper language

Đầu tiên chúng ta phải vào folder *application/language* và tạo ra một folder mới tên là *vietnamese*, đây sẽ là folder dùng để lưu trữ file ngôn ngữ chúng ta sẽ sử dụng.

Sau khi tạo folder trên, chúng ta tạo tiếp file tên là *vi\_lang.php* dùng để chứa các thông số ngôn ngữ, chú ý là phải có hậu tố *lang* ở đằng sau để CI có thể hiểu được.

Chúng ta mở file *vi\_lang* ra và thêm vào đoạn code sau.

```
<?php  
$lang['fullname'] = "Họ tên";  
$lang['email'] = "Hòm thư";  
$lang['phone'] = "Điện thoại";  
$lang['address'] = "Địa chỉ";  
?>
```

Đoạn mã trên rất đơn giản, đó là chúng ta tạo biến mảng \$lang có cú pháp *\$lang['key'] = 'value'*. Key tức là cái khóa, value là giá trị ngôn ngữ cần hiển thị, như vậy chúng ta hoàn toàn dễ dàng làm chủ ngôn ngữ riêng cho mình.

Tiếp theo, chúng ta tạo một controller tên là *demolang*, ngay tại action *index*, chúng ta tiến hành load helper với cú pháp như sau.

```
<?php  
class Demolang extends CI_Controller{  
    public function __construct(){  
        parent::__construct();  
    }  
  
    public function index(){
```

```
    $this->lang->load("vi", "vietnamese");
    $this->load->view("demolang");
}
}
```

Với cách khai báo này, chúng ta dễ dàng hiểu rằng, ngôn ngữ cần load có tên là `vi` và nó nằm trong folder `vietnamese`. Tiếp theo chúng ta sẽ gọi chúng ra ngay tại view.

```
<?php
echo $this->lang->line('fullname')." : TKPro<br />";
echo $this->lang->line('email')." : lienhe@tkpro.edu.vn<br />";
echo $this->lang->line('address')." : tkpro.edu.vn<br />";
echo $this->lang->line('phone')." : 0913694918";
?>
```

Chúng ta gọi đoạn text với ngôn ngữ tương ứng thông qua key theo cú pháp `$this->lang->line('tham_so')`.

TKPro.edu.vn

## Bài 16: Kỹ thuật master layout trong codeigniter

Đây là một vấn đề mở rộng mà CI không đề cập trong user guide, đó là khái niệm về **master layout**. Trong CI chúng ta phải truy cập từng controller, và trong từng **controller** sẽ có nhiều action, và nhiệm vụ của từng action sẽ làm công việc hiển thị thông tin cần thiết, ví dụ như trong controller category, có add, edit, ... Trong từng action như vậy, chúng ta phải đổ nội dung ra view. Bất kỳ khi các bạn xây dựng một trang web nào đó, thì ở từng action sẽ có phần layout cố định như là header, footer, đó là 2 phần trong action nào chúng ta cũng bắt buộc phải có, nếu như cứ mỗi action/mỗi view lại phải copy lại đoạn html để sử dụng lại thì không hợp lý tí nào cả. Vì khi cần điều chỉnh lại thông tin, chúng ta phải điều chỉnh ở rất nhiều nơi. Đó là lý do mà chúng ta phải có một giải pháp để xử lý vấn đề layout trong CI.

### 1. Cấu hình master layout

Đầu tiên chúng ta phải tạo ra một khung layout cố định, mặc định của nó phải có header & footer. Tất cả các view trong từng action sẽ nằm ở một vị trí nào đó trong layout cố định này. Nếu các bạn thay đổi banner ở phần header thì tất cả các action đều sẽ thay đổi theo. Đó là lý do mà chúng ta buộc phải dùng master layout, để có thể dùng chung cho toàn bộ action. Chúng ta sẽ ví dụ thông qua 2 controller là admin & home, hiển thị 2 layout hoàn toàn khác nhau.

### 2. Thực hành master layout - xây dựng layout admin

Như đã trình bày ở trên, chúng ta cần có một khung layout cố định, tức là một file chứa phần header & footer. Chúng ta vào folder application/views tạo ra 2 folder là admin & home. Trong mỗi folder sẽ có một file tên là main.php, đó là file master. Chúng ta tạo thêm 2 file là index\_view.php cho từng folder, đây sẽ là 2 file hiển thị view trong action admin & home.

Mở file main.php lên, chúng ta tiến hành khai báo html cho nó và tạo ra layout cơ bản gồm header, footer, content, 2 phần kia sẽ hiển thị cố định. Như vậy ở ngay vị trí của content chính là nơi mà chúng ta sẽ phải xử lý load view của toàn bộ action trong từng controller.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title></title>
</head>

<body>
```

```
<div id="top">Banner</div>
<div id="content">

</div>
<div id="footer">© 2018, TKPro.edu.vn</div>
</body>
</html>
```

Chúng ta tạo controller admin có action index và load view ra:

### *Admin Controller - Action Index*

```
<?php
class Admin extends CI_Controller{
    public function __construct() {
        parent::__construct();
    }

    public function index(){
        $this->load->view('admin/index_view');
    }
}
?>
```

Tiếp theo, nếu chúng ta muốn sử dụng master layout, có nghĩa là phần view của controller admin sẽ phải xuất hiện ở vị trí content trong file main.php. Để có thể làm được điều này thì chúng ta phải truyền view sang file main. Để có thể **load view lồng view**, thì chúng ta phải tạo ra một trường để chúng ta truyền tham số sang file main. Và ngay tại đó các bạn phải load tiếp một cái view nữa. Chúng ta sẽ tạo ra một biến \$data['subview'] = 'View của action' sau đó truyền tham số vào file main bằng cú pháp:

```
<?php
class Admin extends CI_Controller{
    public function __construct() {
        parent::__construct();
    }

    public function index(){
        $data['subview'] = 'admin/index_view';
        $this->load->view('admin/main', $data);
    }
}
?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title><?php echo $title; ?></title>
<style>
```

```
*{ margin: 0px; padding: 0px; }
body{ width: 780px; margin: 0px auto; }
#top{ background: blue; margin-top:10px; color:white; height: 100px; line-height: 100px; font-weight: bold; text-align: center; }
#footer{ background: black; color:white; height: 30px; line-height: 30px; font-weight: bold; text-align: center; }
#content{ padding: 5px; }
</style>
</head>

<body>
    <div id="top">Banner</div>
    <div id="content">
        <?php echo $subview; ?>
    </div>
    <div id="footer">© 2018, TKPro.edu.vn</div>
</body>
</html>
```

Sau đó mở file main.php ra, ngay tại vị trí content, chúng ta echo \$subview;, tham số truyền sang view là một cái khóa, sau khi từ action truyền sang nó sẽ bỏ đi cái mảng, tức là chỉ còn lại \$subview mà thôi. Chúng ta thêm 1 số style css cho layout dễ nhìn hơn. Để kiểm tra, chúng ta ra trình duyệt và gõ. localhost/citest/index.php/admin.

Chúng ta không muốn hiển thị chữ admin/index\_view ra, mà chúng ta muốn hiển thị nội dung của index\_view, để làm được điều này thì ngay tại vị trí mà chúng ta echo, chúng ta tiến hành load view với tham số là index\_view, cú pháp như sau:

```
<?php echo $this->load->view($subview); ?>
```

Với master layout, chúng ta có thể dễ dàng thay đổi title page cho từng action, bằng cách truyền tham số title sang file main.php như sau:

```
<?php
class Admin extends CI_Controller{
    public function __construct() {
        parent::__construct();
    }

    public function index(){
        $data['subview'] = 'admin/index_view';
        $data['title'] = 'Admin System';
        $this->load->view('admin/main', $data);
    }
}
?>
```

Với **kỹ thuật master layout**, chúng ta hoàn toàn có thể sử dụng cho toàn bộ action. Để minh họa, chúng ta sẽ ví dụ thông qua controller category và action là add. Mình cũng muốn thông qua ví dụ này để trình bày việc hiển thị dữ liệu thông qua master layout. Chúng ta có biến \$info chứa thông tin cần hiển thị ở addcate\_view, đầu tiên,

chúng ta tiến hành kiểm tra bằng cách in dữ liệu trong cặp thẻ pre, sau đó chúng ta dùng vòng lặp foreach xuất dữ liệu trong mảng vừa khai báo.

```
<?php
class Category extends CI_Controller{
    public function __construct() {
        parent::__construct();
    }

    public function add(){
        $data['subview'] = 'admin/addcate_view';
        $data['info'] = array(
            'name' => 'Hasegawa kaito',
            'website' => 'freetuts.net',
            'email' => 'hoaiminhit1990@gmail.com',
            'phone' => '1234567894556',
        );
        $data['title'] = 'Add A Category';
        $this->load->view('admin/main', $data);
    }
}
?>
```

```
<h1>Category Controller - Action Add</h1>
<?php
echo "<pre>";
print_r($info);
echo "</pre>";

foreach($info as $k => $v) {
    echo "$k : $v<br />";
}
?>
```

### 3. Thực hành master layout - xây dựng layout home

Chúng ta tạo controller home & action index, copy các thông số từ controller admin, thay đổi tên view là xong:

```
<?php
class Home extends CI_Controller{
    public function __construct() {
        parent::__construct();
    }

    public function index(){
        $data['subview'] = 'home/index_view';
        $data['title'] = 'Homepage';
        $this->load->view('home/main', $data);
    }
}
?>
```

## Bài giảng CodeIgniter Framework 3

---

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title><?php echo $title; ?></title>
<style>
*{ margin: 0px; padding: 0px; }
body{ width: 780px; margin: 0px auto; }
#top{ background: orange; color:white; height: 100px; line-height: 100px;
font-weight: bold; text-align: center; }
#footer{ background: black; color:white; height: 30px; line-height: 30px;
font-weight: bold; text-align: center; }
#content{ padding: 5px; }
</style>
</head>

<body>
<div id="top">Banner</div>
<div id="content">
    <?php echo $this->load->view($subview); ?>
</div>
<div id="footer">2014 © by freetuts.net</div>
</body>
</html>
```

Mở trình duyệt và gõ `localhost/citest/index.php/home` để kiểm tra kết quả.



## Bài 17: Chức năng thêm – xóa – sửa trong CodeIgniter

Để có thể hiểu rõ cách thức thêm – xóa – sửa dữ liệu trong CodeIgniter, chúng ta sẽ thực hiện chức năng quản lý thành viên.

### 1. Xây dựng database

Chúng ta xây dựng bảng User có cấu trúc như sau.

```
CREATE TABLE `user` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `password` char(32) COLLATE utf8_unicode_ci DEFAULT NULL,
  `email` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `level` int(1) DEFAULT '1',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=23 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
```

Sau đó thêm vài dòng dữ liệu mẫu như sau:

```
insert into `user`(`id`, `username`, `password`, `email`, `level`) values
(1, 'admin', '123', 'admin@tkpro.edu.vn', 2),
(2, 'htkhai', '123456', 'lienhe@tkpro.edu.vn', 1);
```

Sau khi hoàn tất việc tạo dữ liệu, thì chúng ta tiến hành khai báo cấu hình database trong CI như sau (trong file *application/config/database.php*).

```
$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'root';
$db['default']['password'] = '';
$db['default']['database'] = 'codeigdb';
$db['default']['dbdriver'] = 'mysql';
$db['default']['dbprefix'] = '';
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = '';
$db['default']['char_set'] = 'utf8';
$db['default']['dbcollat'] = 'utf8_general_ci';
$db['default']['swap_pre'] = '';
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;
```

Các bạn sửa lại thông số cho đúng với máy tính của mình.

Hoàn thành công việc cấu hình database xong, tiếp theo chúng ta cho nó autoload, đỡ phải mất công khai báo trong file model. Với chức năng chúng ta đang xây dựng, chúng ta sẽ dùng tới session, form validation, database, nên chúng ta autoload các thư viện này. Để autoload thư viện, trong folder config, mở file autoload.php lên và tìm đến dòng 55 thêm vào như sau:

```
$autoload['libraries'] = array('database', 'session', 'form_validation');
```

Như vậy là xong bước 1, tiếp theo chúng ta sẽ cấu hình **master layout** để tiện việc thao tác với giao diện hơn.

## 2. Cấu hình master layout

Vào folder application/views, tạo folder user và thêm 4 file vào folder này. File main.php là file chứa giao diện cố định. Tiếp theo chúng ta tạo controller user và action index, tiến hành khai báo thông tin cấu hình master layout như sau.

```
class User extends CI_Controller {

    protected $_data;

    public function __construct() {
        parent::__construct();
        $this->load->helper('url');
    }

    public function index() {
        $this->_data['subview'] = 'user/index_view';
        $this->_data['titlePage'] = 'List All User';
        $this->load->view('user/main.php', $this->_data);
    }
}
```

Chúng ta khởi tạo biến `$_data` để có thể gọi trong mọi action. Tiếp theo chúng ta vào file main.php xử lý html và load `$subview` vào div content. Chúng ta sẽ phải dùng tới hàm `base_url()` nên các bạn gọi `helper url` ngay trong constructor để có thể tái sử dụng lại trong mọi action.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title><?php echo $titlePage; ?></title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <meta name="author" content="Hasegawa Kaito" />
        <link rel="stylesheet" type="text/css" href="<?php echo base_url();
?>asset/admin/css/style.css" charset="utf-8" />
        <script language="javascript">
            function xacnhan() {
                if (!window.confirm('You want delete user ?')) {
                    return false;
                }
            }
        </script>
    </head>
    <body id="manage">
        <div id="header">
```

```
<ul id="menu">
    <li><a href="php echo base_url(); index.php/user"?&gt;User&lt;/a&gt;&lt;/li&gt;
        &lt;li&gt;&lt;a href="#"&gt;Chuyên mục&lt;/a&gt;&lt;/li&gt;
        &lt;li&gt;&lt;a href="#"&gt;Bài viết&lt;/a&gt;&lt;/li&gt;
        &lt;li&gt;&lt;a href="#"&gt;Comment&lt;/a&gt;&lt;/li&gt;
    &lt;/ul&gt;
    &lt;p&gt;
        Xin chào &lt;span&gt;Admin&lt;/span&gt;
        &lt;a href="#"&gt;[ Thoát ]&lt;/a&gt;
    &lt;/p&gt;
&lt;/div&gt;&lt;!-- End header --&gt;
&lt;div id="content"&gt;&lt;!-- ===== Begin #content ===== --&gt;
    &lt;?php echo $this-&gt;load-&gt;view($subview); ?&gt;
&lt;/div&gt;&lt;!-- End #content --&gt;
&lt;div id="footer"&gt;
    &amp;copy; 2018, TKPro.edu.vn
&lt;/div&gt;&lt;!-- End #footer --&gt;
&lt;/body&gt;
&lt;/html&gt;</pre
```

Chúng ta tạo thêm folder asset/admin/css dùng để chứa file css của layout.

```
*{margin: 0px; padding: 0px; font: normal 12px/120% tahoma; color: #333;}
ul{list-style: none;}
a{text-decoration: none;}
body{ width: 960px; margin: 0px auto;}
a:hover, a:active, a:visited{color: #268;}
h1,h2,h3,h4,h5,h6{font: 12px/140% tahoma;}
html{background: #f0f0f0; }

/*==== login ===*/
#header p{text-align: center; margin: 10px; color: #f1f1f1;}
#header p a{color: #f1f1f1; font-weight: 900;}
#header p span{color: #ffffcf; font-weight: 900;}
#header p a:hover{text-decoration: underline; }

.show{padding: 20px 10px; margin-bottom: 10px; background: #dfdfdf;
border-left: 1px solid #bfbfbf; border-top: 1px solid #bfbfbf;
border-right: 1px solid #fff; border-bottom: 1px solid #fff;
}
.show legend{color: #268; font: 900 14px arial; padding: 0px 10px;}
.show label{display: block; float: left; width: 110px; margin-top: 2px; }

.input,select{margin-bottom: 5px; height: 20px; padding-left: 3px; }
.input:focus{background: #ffffcf; border: 1px solid #cfcfcf; }

.button{display: block; margin: auto; }
.btn{background: green; color: white; border: none; padding: 5px;
font-weight: bold; }
.btn:hover{ background: orange; cursor: pointer; }
```

## Bài giảng CodeIgniter Framework 3

---

```
/*==== manage ===*/
#header{background: #999; border-bottom: 1px solid #fff;
padding-bottom: 5px; margin: 10px 0px 10px 0px;}

#menu{overflow: hidden; margin: 0px auto; width: 400px;}
#menu li{float: left;}
#menu li a{font: 900 12px/30px arial; color: #f5f5f5; width: 98px;
display: block; text-align: center;
border-top: 1px solid #afafaf; border-bottom: 1px solid #afafaf;
border-left: 1px solid #afafaf; border-right: 1px solid #777;}
#menu li a:hover{color: #fff; background: #888; }

#footer{background: #999; height: 30px; line-height: 30px;
text-align: center; color: #f5f5f5; }

#user,#categories,#articles,#comment{margin: 20px auto; width: 400px; }

#articles{width: 960px; }
#comment{width: 500px; }

#user fieldset .input{width: 220px; }
#articles fieldset textarea,#comment fieldset textarea{margin-bottom: 5px; }
#articles fieldset textarea:focus,#comment fieldset
textarea:focus{background: #ffffcf; border: 1px solid #cfcfcf; }

ul.err{padding: 10px; }
ul.err li{color: #f00; }

.message{color: green; text-align: center; line-height: 30px;
font-size: 14px; }

.list{width: 960px; margin: 0px auto 20px; border: 1px solid #cfcfcf; }
.list tr:hover{background: #ffffcf; }
.list th{font: 900 12px/30px tahoma; background: #cfcfcf;
border-left: 1px solid #efefef; border-right: 1px solid #afafaf; }
.list td{text-align: center; line-height: 26px;
border-top: 1px solid #cfcfcf; }
.list_page{height: 26px; }
.list_page span{color: #333; padding: 2px 5px; margin: 0px 5px;
border: 1px solid #cfcfcf; font-weight: 900; }
.list_page a{margin: 0px 4px; }

#list_user,#list_articles,#list_comment{width: 960px; }
#list_cate{width: 600px; }

.admin{ color:red; font-weight: bold; }

.mess_succ{background: #99CC00;color: #FFFFFF;text-align: center;
font-weight: bold; margin: 15px; }
```

```
.mess_succ ul{list-style-type:none;padding:0px;margin:0px;}  
.mess_succ ul li{padding:10px 10px 10px 15px;}  
.mess_error{background: #FF6600; color: #FFFFFF; margin:15px;}  
.mess_error ul{list-style-type:none; padding:0px; margin:0px;}  
.mess_error ul li{padding:10px 10px 10px 15px;}
```

Tại file `index_view.php`, chúng ta cho nội dung đơn giản để test xem cấu hình master layout thành công chưa, chạy link `localhost/citest/index.php/user`.

Sau khi cấu hình thành công master layout, việc tiếp theo là chúng ta sẽ liệt kê danh sách các thành viên ra layout.

### 3. Viết chức năng liệt kê thành viên

Nhắc tới liệt kê danh sách thành viên thì chúng ta sẽ nghĩ ngay tới việc phải thao tác với model & active record, vậy trong folder models, chúng ta tạo file `muser.php`.

```
<?php  
class Muser extends CI_Model{  
    protected $_table = 'user';  
    public function __construct() {  
        parent::__construct();  
    }  
  
    public function getList(){  
        $this->db->select('id, username, email, level');  
        return $this->db->get($this->_table)->result_array();  
    }  
  
    public function countAll(){  
        return $this->db->count_all($this->_table);  
    }  
}
```

Chúng ta không muốn liệt kê toàn bộ cột trong bảng user mà chỉ hiển thị các cột: `id`, `email`, `username`, `level`. Do chúng ta đang viết hàm vì thế cần phải trả kết quả về với cú pháp như sau: `return $this->db->get($this->_table)->result_array()`. Với phương thức `result_array()`, chúng ta sẽ liệt kê toàn bộ record có trong table. Chúng ta cũng có thể đếm tổng số record trong table với phương thức `count_all()`.

Quay lại controller user, tiến hành load model ra, gán biến `$this->_data['info']` trả tới phương thức `getList` trong model, sau đó sang `index_view` in dữ liệu ra

```
echo "<pre>";  
print_r($info);  
echo "</pre>";
```

```
Array  
(  
    [0] => Array  
        (
```

```
[id] => 1
[username] => admin
[email] => admin@tkpro.edu.vn
[level] => 2
)

[1] => Array
(
    [id] => 2
    [username] => htkhai
    [email] => lienhe@tkpro.edu.vn
    [level] => 1
)
)
```

Chạy link `localhost/citest/index.php/user`, nếu trình duyệt trả về kết quả như hình trên thì chúng ta đã lấy dữ liệu thành, tiếp theo chúng ta sẽ dùng vòng lặp foreach để dữ liệu vào table để hiển thị đẹp mắt hơn nhé.

```
<table cellpadding="0" cellspacing="0" border="0" width="100%">
<tr>
    <td colspan="6" align="center"><a href="php echo base_url(); ?&gt;index.php/user/add"&gt;Add User&lt;/a&gt;&lt;br /&gt;&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
    &lt;th width="20"&gt;STT&lt;/th&gt;
    &lt;th width="80"&gt;Name&lt;/th&gt;
    &lt;th width="80"&gt;Email&lt;/th&gt;
    &lt;th width="30"&gt;Level&lt;/th&gt;
    &lt;th width="10"&gt;Edit&lt;/th&gt;
    &lt;th width="10"&gt;Delete&lt;/th&gt;
&lt;/tr&gt;
&lt;tr&gt;
    &lt;?php
        $stt=0;
        foreach($info as $item) {
            $stt++;
            echo "&lt;tr&gt;";
            echo "&lt;td&gt;$stt&lt;/td&gt;";
            echo "&lt;td&gt;$item[username]&lt;/td&gt;";
            echo "&lt;td&gt;$item[email]&lt;/td&gt;";
            if($item['level'] == 2){
                echo "&lt;td class='admin'&gt;Administrator&lt;/td&gt;";
            }else{
                echo "&lt;td&gt;Member&lt;/td&gt;";
            }
            echo "&lt;td&gt;&lt;a href=".base_url()."index.php/user/edit/$item[id]&gt;Edit&lt;/a&gt;&lt;/td&gt;";
            echo "&lt;td&gt;&lt;a href=".base_url()."index.php/user/del/$item[id] onclick='return xacnhuan();'&gt;Delete&lt;/a&gt;&lt;/td&gt;";
        }
    &lt;/?php&gt;
&lt;/tr&gt;</pre
```

```
        echo "</tr>";
    }
?>
</tr>
<tr>
    <td colspan="6" align="center">Có tổng cộng <?php echo
$total_user; ?> thành viên.<br /></td>
</tr>
</table>
```

Chúng ta có 6 cột tất cả, khởi tạo \$stt = 0 ở phía ngoài vòng lặp sau đó cho nó tăng dần \$stt++, check ngay chỗ hiển thị level, nếu \$info['level'] = 2 thì là administrator, ngược lại bằng 1 thì là member, \$total\_user là biến truyền từ controller sang thông qua hàm countAll trong model.

```
<table cellpadding="0" cellspacing="0" border="0" width="100%" id="list_cate" class="list">
    <tr><th width="20">STT</th>
        <th width="80">Name</th>
        <th width="80">Email</th>
        <th width="30">Level</th>
        <th width="10">Edit</th>
        <th width="10">Delete</th>
    </tr>
    <tr>
        <?php
            $stt=0;
            foreach($info as $item) {
                $stt++;
                echo "<tr>";
                echo "<td>$stt</td>";
                echo "<td>$item[username]</td>";
                echo "<td>$item[email]</td>";
                if($item['level'] == 2){
                    echo "<td class='admin'>Administrator</td>";
                }else{
                    echo "<td>Member</td>";
                }
                echo "<td><a href=".base_url()."index.php/user/edit/$item[id]> Edit</a></td>";
                echo "<td><a href=".base_url()."index.php/user/del/$item[id] onclick='return
xacnhan(); '>Delete</a></td>";
                echo "</tr>";
            }
        ?>
    </tr>
    <tr><td colspan="6" align="center">Có tổng cộng <?php echo
$total_user; ?> thành viên.<br /></td>
    </tr>
</table>
```

Vậy chúng ta có action index như sau.

```
public function index() {  
    $this->_data['subview'] = 'user/index_view';  
    $this->_data['titlePage'] = 'List All User';  
    $this->load->model('Muser');  
  
    $this->_data['info'] = $this->Muser->getList();  
    $this->_data['total_user'] = $this->Muser->countAll();  
    $this->load->view('user/main.php', $this->_data);  
}
```

Tiếp theo chúng ta sẽ thao tác với một chức năng khá đơn giản với bất kỳ ứng dụng nào, đó là chức năng Delete User.

### 4. Viết chức năng delete user

Chúng ta tạo thêm action del, đây là action thao tác đến việc xóa user.

```
public function del($id) {  
    $this->load->model('Muser');  
    $this->Muser->deleteUser($id);  
    $this->session->set_flashdata("flash_mess", "Delete Success");  
    redirect(base_url() . "index.php/user");  
}
```

Giải thích cho việc truyền \$id vào hàm del: thay vì chúng ta dùng uri->segment để xác định vị trí của id, vì thật chất nó là **phương thức GET** trong php thuận nên nó không thật sự an toàn, vì thế chúng ta có thể đếm vị trí id bằng cách sau: Ví dụ link của chúng ta có cấu trúc là *localhost/citest/index.php/user/del/id...* thì user là 1, del là 2, id là 3. Vậy ngay tại action del ta truyền trực tiếp \$id vào nó vẫn hiểu id đang ở vị trí thứ 3.

Với việc delete một user, chúng ta cũng cần phải xuất ra thông báo là delete thành công hay gì đó, và chúng ta lựa chọn giải pháp dùng **flashdata** trong **session** xuất ra thông báo, vì flashdata chỉ xuất hiện 1 lần, sau khi F5 thì nó sẽ biến mất.

Vào file model muser, viết phương thức **deleteUser()**, muốn xóa một id nào đó chúng ta phải so sánh id bằng với \$id mà chúng ta đang muốn truyền vào. Sau đó mình sẽ sử dụng **\$this->db->delete(\$this->\_table)**, ngay tại controller chúng ta có cú pháp như sau: **\$this->Muser->deleteUser(\$id)**. Sau khi xóa xong chúng ta phải xuất ra thông báo, vậy ngay tại action del, chúng ta khởi tạo flashdata bằng cú pháp **\$this->session->set\_flashdata("flash\_mess", "Delete Success")**, flash\_mess là khóa chúng ta đặt tên, còn tham số thứ 2 là câu thông báo lỗi muốn xuất ra.

Khởi tạo xong, tại action index, chúng ta gọi nó ra bằng cú pháp **\$this->\_data['mess'] = \$this->session->flashdata('flash\_mess')**. Tại index\_view, chúng ta thêm đoạn code sau để có thể xuất ra thông báo lỗi, xóa xong tự về trang list user với hàm **redirect**.

```
<?php  
if(isset($mess) && $mess != '') {
```

```
        echo "<div class='mess_succ'>";
        echo "<ul>";
            echo "<li>$mess</li>";
        echo "</ul>";
        echo "</div>";
    }
?>
```

Nếu tồn tại \$mess và \$mess khác rỗng thì xuất ra thông báo là Delete Success. Bay giờ chúng ta refresh trình duyệt và click xóa 1 user nào đó và xem kết quả.

Đã xong chức năng delete user, tiếp theo chúng ta sẽ viết tiếp chức năng add user.

### 5. Viết chức năng add user

Vào controller user tạo thêm action add, với các thông số như sau.

```
public function add() {
    $this->_data['titlePage'] = 'Add A User';
    $this->_data['subview'] = 'user/add_view';
    $this->load->view('user/main.php', $this->_data);
}
```

Vào trong file add\_view.php thêm vào html như sau.

```
<form action="php echo base_url(); ?&gt;index.php/user/add" method="post"
id="categories"&gt;
    &lt;?php
        echo "&lt;div class='mess_error'&gt;";
        echo "&lt;ul&gt;";
            if(validation_errors() != ''){
                echo "&lt;li&gt;".validation_errors()."&lt;/li&gt;";
            }
        echo "&lt;/ul&gt;";
        echo "&lt;/div&gt;";
    ?&gt;
    &lt;fieldset class="show"&gt;
        &lt;legend align="center"&gt;Username Infomations&lt;/legend&gt;
        &lt;label&gt;Username:&lt;/label&gt;&lt;input type="text" name="username" size="28" class="input"/&gt;
        &lt;label&gt;Email:&lt;/label&gt;&lt;input type="text" name="email" size="28" class="input"/&gt;
        &lt;label&gt;Password:&lt;/label&gt;&lt;input type="password" name="password" size="28" class="input"/&gt;
        &lt;label&gt;Re-Pass:&lt;/label&gt;&lt;input type="password" name="password2" size="28" class="input"/&gt;
        &lt;br /&gt;
        &lt;label&gt;Level:&lt;/label&gt;&lt;select name="level"&gt;
            &lt;option value="1" selected&gt;Member&lt;/option&gt;
            &lt;option value="2" &gt;Administrator&lt;/option&gt;
        &lt;/select&gt;&lt;br /&gt;
        &lt;label&gt;&amp;nbsp;&lt;/label&gt;&lt;input type="submit" name="ok" value="Add User" class="btn" /&gt;
    &lt;/fieldset&gt;
&lt;/form&gt;</pre
```

Hàm `validation_errors()` dùng để xuất ra thông báo lỗi kiểm soát dữ liệu nhập vào. Chúng ta chạy link `/localhost/citest/index.php/user/add` và xem kết quả.

Sau khi tạo form thành công, việc tiếp theo là chúng ta phải xử lý kiểm tra xem người dùng có nhấn nút submit hay chưa, nếu chưa thì báo lỗi, sử dụng **form validation** tạo ra các luật theo ý của các bạn.

Ở đây mình không giải thích lại toàn bộ các luật trên, ví chúng ta đã tìm hiểu chúng trong các bài trước rồi nhé, mình chỉ nói về hai luật mới, đó là `matches` & `callback`. Luật `matches` kiểm tra xem password có trùng với repass hay không, giá trị truyền vào là tên form của re-pass. Luật `callback` sẽ kiểm tra sự trùng lặp dữ liệu trong database.

Sau khi khai báo các luật validation, chúng ta cần kiểm tra xem các `set_rules` ở trên có hợp lệ hay không bằng phương thức `$this->form_validation->run()`, nếu nó bằng `TRUE` thì chúng ta tiến hành insert dữ liệu vào database. Nhưng trước khi insert, chúng ta phải kiểm tra xem user đó đã có trong csdl hay chưa, để làm được thao tác đó chúng ta sẽ có phương thức `check_user` ngay tại controller như sau.

```
public function check_user($user) {
    $this->load->model('Muser');
    $id=$this->uri->segment(3);
    if ($this->Muser->checkUsername($user) == FALSE) {
        $this->form_validation->set_message("check_user", "Your
username has been registered, please try again!");
        return FALSE;
    } else {
        return TRUE;
    }
}
```

Tương tự như `username`, chúng ta cũng cần check luôn `email` xem có trùng lặp hay không, 2 phương thức khá giống nhau:

```
public function check_email($email) {
    $this->load->model('Muser');
    $id=$this->uri->segment(3);
    if ($this->Muser->checkUsername($email) == FALSE) {
        $this->form_validation->set_message("check_email", "Your email
has been registered, please try again!");
        return FALSE;
    } else {
        return TRUE;
    }
}
```

Như vậy chúng ta phải có một phương thức trong model để thực thi `check_user` & `check_email`, vào file `muser` tạo ra 2 phương thức, `checkUsername` & `checkEmail`, với các thông số sau.

```
public function checkUsername($user) {
    $this->db->where('username', $user);
```

```
$query=$this->db->get($this->_table);
if($query->num_rows() > 0) {
    return FALSE;
} else{
    return TRUE;
}

public function checkEmail($email){
    $this->db->where('email',$email);
    $query=$this->db->get($this->_table);
    if($query->num_rows() > 0) {
        return FALSE;
    } else{
        return TRUE;
    }
}
```

Nếu nó lớn hơn 0 tức là nó tồn tại rồi thì chúng ta chỉ return FALSE, nếu nó tồn tại thì xuất ra thông báo lỗi.

Sau khi hoàn tất việc kiểm tra trùng lặp dữ liệu, bước tiếp theo chúng ta tiến hành insert. Quay lại file muser viết phương thức insert như sau.

```
public function insertUser($data_insert){
    $this->db->insert($this->_table,$data_insert);
}
```

Tạo ra một cái array, chứa các thông tin mà chúng ta cần thêm vào CSDL, chúng ta dùng phương thức \$this->input->post() để thêm dữ liệu. Như vậy, insert xong sẽ chuyển về trang index, đồng thời xuất ra thông báo Added.

```
public function add() {
    $this->_data['titlePage'] = 'Add A User';
    $this->_data['subview'] = 'user/add_view';

    $this->form_validation->set_rules("username", "Username",
"required|xss_clean|trim|min_length[4]|callback_check_user");
    $this->form_validation->set_rules("password", "Password",
"required|matches[password2]|trim|xss_clean");
    $this->form_validation->set_rules("email", "Email",
"required|trim|xss_clean|valid_email|callback_check_email");

    if ($this->form_validation->run() == TRUE) {
        $this->load->model("Muser");
        $data_insert = array(
            "username" => $this->input->post("username"),
            "password" => $this->input->post("password"),
            "email"     => $this->input->post("email"),
            "level"     => $this->input->post("level"),
        );
        $this->Muser->insertUser($data_insert);
    }
}
```

```
        $this->session->set_flashdata("flash_mess", "Added");
        redirect(base_url() . "index.php/user");
    }
    $this->load->view('user/main.php', $this->_data);
}
```

Như vậy là chúng ta đã hoàn tất việc thêm một thành viên vào CSDL, tiếp theo chúng ta sẽ viết chức năng edit user.

### 6. Viết chức năng edit user

Để edit được, các bạn phải nhấn vào link edit và khi đó thông qua một id mà chúng ta muốn sửa thể hiện trên đường dẫn, chúng ta load lên thông tin của user cần sửa. Sau đó, chúng ta sẽ cập nhập dữ liệu từ trong form ngược trở lại với CSDL, và dĩ nhiên phải trải qua thao tác validation trước khi tiến hành update vào CSDL. Đó là mục đích mà chúng ta phải cần làm ở trong phần edit này.

Phần này khá giống với các thao tác mà chúng ta đã làm ở phần add user, cho nên phần này mình không giải thích nhiều, chỉ tập trung vào sự khác biệt giữa add & edit.

Chúng ta tạo action edit và truyền \$id vào, để có thể edit bất kì record nào thì chúng ta phải lấy ra được cái id của record đó. Chúng ta vào file muser viết phương thức **getUserById** như sau (row\_array là hàm hỗ trợ chúng ta lấy ra một record):

```
public function getUserById($id) {
    $this->db->where("id", $id);
    return $this->db->get($this->_table)->row_array();
}
```

Sau khi có phương thức trên, chúng ta sẽ tiến hành lấy dữ liệu ra form ra bằng cú pháp `$this->_data['info'] = $this->Muser->getUserById($id)`. Sau đó, chúng ta sang edit\_view copy phần form bên add\_view vào đổi lại đường dẫn trong form như sau:

```
<form action="<?php echo base_url(); ?>index.php/user/edit/<?php echo
$info['id']; ?>" method="post" id="categories">
    <?php
    echo "<div class='mess_error'>";
    echo "<ul>";
        if(validation_errors() != ''){
            echo "<li>".validation_errors()."</li>";
        }
    echo "</ul>";
    echo "</div>";
    ?>
    <fieldset class="show">
        <legend align="center">Edit Username: <?php echo $info['username'];
?></legend>

        <label>Username:</label><input type="text" name="username"
value="<?php echo $info['username']; ?>" size="28" class="input"/>
```

## Bài giảng CodeIgniter Framework 3

```
<label>Email:</label><input type="text" name="email" size="28" value=<?php echo $info['email']; ?>" class="input"/>
    <label>Password:</label><input type="password" name="password" size="28" class="input"/>
    <label>Re-Pass:</label><input type="password" name="password2" size="28" class="input"/><br />

    <label>Level:</label><select name="level">
        <option value='1' <?php if ($info['level'] == 1) echo ' selected'; ?> >Member</option>
        <option value='2' <?php if ($info['level'] == 2) echo ' selected'; ?> >Administrator</option>
    </select><br />
    <label>&ampnbsp</label><input type="submit" name="ok" value="Edit User" class="btn" />
</fieldset>
</form>
```

Tiếp theo, chúng ta quay lại file muser và viết phương thức **updateUser** như sau:

```
public function edit($id) {
    $this->load->library("form_validation");
    $this->load->model('Muser');
    $this->_data['titlePage'] = "Edit A User";
    $this->_data['subview'] = "user/edit_view";

    $this->_data['info'] = $this->Muser->getUserById($id);
    $this->form_validation->set_rules("username", "Username",
"required|xss_clean|trim|min_length[4]|callback_check_user");
    $this->form_validation->set_rules("password", "Password",
"matches[password2]|trim|xss_clean");
    $this->form_validation->set_rules("email", "Email",
"required|trim|xss_clean|valid_email|callback_check_email");
    if ($this->form_validation->run() == TRUE) {
        $data_update = array(
            "username" => $this->input->post("username"),
            "email" => $this->input->post("email"),
            "level" => $this->input->post("level"),
        );
        if ($this->input->post("password")) {
            $data_update['password'] = $this->input->post("password");
        }
        $this->Muser->updateUser($data_update, $id);
        $this->session->set_flashdata("flash_mess", "Update Success");
        redirect(base_url() . "index.php/user");
    }
    $this->load->view('user/main.php', $this->_data);
}
```

Khá giống với action add, chỉ khác ở vài chỗ: ví dụ như các bạn chỉ muốn change email, username không muốn change password thì check như sau: nếu tác động vào textbox password thì mới update, còn không vẫn giữ nguyên password cũ.

```
if ($this->input->post("password")) {  
    $data_update['password'] = $this->input->post("password");  
}
```

Vấn đề chúng ta mắc phải ở đây là, chúng ta đang dùng lại 2 phương thức `check_user` & `check_email`, và sự khác biệt đối với edit thì chúng ta phải kiểm tra thêm cái id nữa. Tuy nhiên chúng ta hoàn toàn có thể sử dụng một lúc cả 2 phương thức cho add & edit bằng cách: quay trở lại file muser ngay tại vị trí của 2 phương thức, chúng ta thêm vào `$id = ""`, đối với add thì không có id và với edit thì có id. Nếu `$id != ""` tức là nó có giá trị thì chúng ta sẽ có `$this->db->where("id !=", $id)`, tương tự ở email cũng vậy.

```
public function checkUsername($user, $id="") {  
    if($id != ""){  
        $this->db->where("id !=", $id);  
    }  
    $this->db->where('username',$user);  
    $query=$this->db->get($this->_table);  
    if($query->num_rows() > 0){  
        return FALSE;  
    }else{  
        return TRUE;  
    }  
}  
  
public function checkEmail($email,$id="") {  
    if($id != ""){  
        $this->db->where("id !=", $id);  
    }  
    $this->db->where('email',$email);  
    $query=$this->db->get($this->_table);  
    if($query->num_rows() > 0){  
        return FALSE;  
    }else{  
        return TRUE;  
    }  
}
```

Như vậy, trong controller user, tại vị trí `check_user` chúng ta chỉnh sửa lại như sau:

```
public function check_user($user, $id) {  
    $this->load->model('Muser');  
    $id=$this->uri->segment(3);  
    if ($this->Muser->checkUsername($user, $id) == FALSE) {  
        $this->form_validation->set_message("check_user", "Your  
username has been registered, please try again!");  
        return FALSE;  
    } else {  
        return TRUE;  
    }  
}  
  
public function check_email($email,$id) {
```

```
$this->load->model('Muser');
$id=$this->uri->segment(3);
if ($this->Muser->checkEmail($email, $id) == FALSE) {
    $this->form_validation->set_message("check_email", "Your email
has been registered, please try again!");
    return FALSE;
} else {
    return TRUE;
}
}
```

### Toàn bộ action edit:

```
public function edit($id) {
    $this->load->model('Muser');
    $this->_data['titlePage'] = "Edit A User";
    $this->_data['subview'] = "user/edit_view";

    $this->_data['info'] = $this->Muser->getUserById($id);
    $this->form_validation->set_rules("username", "Username",
"required|xss_clean|trim|min_length[4]|callback_check_user");
    $this->form_validation->set_rules("password", "Password",
"matches[password2]|trim|xss_clean");
    $this->form_validation->set_rules("email", "Email",
"required|trim|xss_clean|valid_email|callback_check_email");
    if ($this->form_validation->run() == TRUE) {
        $data_update = array(
            "username" => $this->input->post("username"),
            "email" => $this->input->post("email"),
            "level" => $this->input->post("level"),
        );
        if ($this->input->post("password")) {
            $data_update['password'] = $this->input->post("password");
        }
        $this->Muser->updateUser($data_update, $id);
        $this->session->set_flashdata("flash_mess", "Update Success");
        redirect(base_url() . "index.php/user");
    }
    $this->load->view('user/main.php', $this->_data);
}
```

## Bài 18: Tìm hiểu Library Shopping Cart trong CodeIgniter

Bài này sẽ giúp các bạn tìm hiểu một library khá quan trọng, đó là shopping cart. Vì thư viện Shopping Cart trong CodeIgniter được tích hợp sẵn Session nên các bạn phải đặt giá trị cho key encryption\_key trong file application/config.php nhé.

Trong CodeIgniter, Shopping Cart được lưu dưới dạng mảng và được mã hóa để lưu vào Session, mà Session trong codeigniter gắn liền với cookie (cookie lưu được 4Kb) nên bạn không thể lưu với dung lượng lớn được. Chính vì thế ta chỉ có thể lưu id sản phẩm và một số thông tin như giá cả chứ không thể lưu toàn bộ thông tin như hình ảnh, tiêu đề, tóm tắt.

### 1. Cấu hình library shopping cart

Muốn thao tác với shopping cart trong CI thì trước tiên phải load nó theo cú pháp:

```
<?php  
class Shop extends CI_Controller{  
    public function __construct(){  
        parent::__construct();  
        $this->load->library("cart");  
    }  
}
```

### 2. Thêm sản phẩm

Shopping Cart trong CI lưu dưới dạng mảng và có những key bắt buộc không được sửa như như id, qty, price, name. Ngoài ra nó có thêm key options chứa các thông tin con như size, color của sản phẩm nên nếu những key này không giống với database của bạn thì hãy xử lý trước khi insert để đồng bộ. Các bạn xem đoạn code dưới đây:

```
public function insert(){  
    $data=array(  
        "id" => "1",  
        "name" => "Áo thun",  
        "qty" => "1",  
        "price" => "100000",  
        "option" => array("color" => "màu đỏ"),  
    );  
    // Them san pham vao gio hang  
    if($this->cart->insert($data))  
        echo "Them san pham thanh cong";  
    else  
        echo "Them san pham that bai";  
}
```

Trong đoạn code này tôi đã khởi tạo một sản phẩm mới chứa trong mảng \$data. Sau đó sử dụng hàm insert của **thư viện Shopping Cart** để tiến hành thêm sản phẩm vào giỏ hàng. Các bạn hãy chạy code và sẽ thấy kết quả thành công hay thất bại.

### 3. Hiển thị danh sách sản phẩm

Trong đoạn code hiển thị danh sách sản phẩm, chúng ta dùng hàm `contents` trong thư viện shopping cart để tiến hành xem toàn bộ thông tin sản phẩm.

```
public function show() {
    //Show thong tin chi tiet gio hang
    $data=$this->cart->contents();
    echo "<pre>";
    print_r($data);
    echo "</pre>";
}
```

### 4. Xóa một sản phẩm

Muốn xóa một sản phẩm thì chúng ta phải biết được cái rowid của sản phẩm đó. Khi các bạn insert sản phẩm thì trường rowid tự động phát sinh, trường này có nhiệm vụ giúp chúng ta phân biệt từng sản phẩm một cách toàn diện.

Muốn có được giá trị **rowid** thì chúng ta bắt buộc phải sử dụng [vòng lặp foreach](#) để xuất dữ liệu. Và có thể xóa được một sản phẩm thì trước tiên chúng ta cần phải có hơn một sản phẩm trong giỏ hàng.

Chúng ta có đoạn code insert mới như sau.

```
public function insert(){
    $data = array(
        array(
            'id' => '1',
            'name' => 'Áo thun',
            'price' => '10000',
            'qty' => '1',
            'options' => array(color => 'màu đỏ')
        ),
        array(
            'id' => '2',
            'name' => 'Quần Jeans',
            'price' => '20000',
            'qty' => '1',
            'options' => array('color=> 'màu xanh')
        ),
        array(
            'id' => '3',
            'name' => 'Mũ luõi trai',
            'price' => '30000',
            'qty' => '1',
            'options' => array('color' => 'màu vàng')
        ),
    );
    // Them san pham vao gio hang
    if($this->cart->insert($data))
```

```
        echo "Them san pham thanh cong";
    else
        echo "Them san pham that bai";
}
```

Kết quả:

```
Array
(
    [c4ca4238a0b923820dcc509a6f75849b] => Array
        (
            [rowid] => c4ca4238a0b923820dcc509a6f75849b
            [id] => 1
            [name] => Áo thun
            [qty] => 1
            [price] => 100000
            [option] => Array
                (
                    [color] => màu đỏ
                )
            [subtotal] => 100000
        )

    [72ea95e2650aca6c0707014d757b7ce4] => Array
        (
            [rowid] => 72ea95e2650aca6c0707014d757b7ce4
            [id] => 2
            [name] => Quần Jeans
            [price] => 20000
            [qty] => 1
            [options] => Array
                (
                    [color] => màu xanh
                )
            [subtotal] => 20000
        )

    [9acc658922ae3040b17a5ddaf35a4dbf] => Array
        (
            [rowid] => 9acc658922ae3040b17a5ddaf35a4dbf
            [id] => 3
            [name] => Mũ lưỡi trai
            [price] => 30000
            [qty] => 1
            [options] => Array
                (
                    [color] => màu vàng
                )
            [subtotal] => 30000
        )
)
```

Để xóa sản phẩm nào thì ta thiết lập qty = 0, ví dụ tôi muốn xóa sản phẩm có id là 1 thì việc trước tiên tôi phải lấy được rowid của sản phẩm đó bằng cách sử dụng vòng lặp for để lấy giá trị rowid. Sau đó trường rowid sẽ ứng với phần tử rowid mà chúng ta vừa lấy được.

Các bạn xem đoạn code xóa một sản phẩm.

```
public function deleteOne(){
    $data=$this->cart->contents();
    foreach($data as $item){
        if($item['id'] == "1"){
            $item['qty'] = 0;
            $del = array("rowid" => $item['rowid'], "qty" => $item['qty']);
        }
    }
    if($this->cart->update($del))
        echo "Xoa san pham thanh cong";
    else
        echo "Xoa san pham that bai";
}
```

## 5. Xóa hết sản phẩm

Hàm xóa hết sản phẩm của Shopping Cart trong CI giống với hàm hủy toàn bộ session. Chúng ta dùng hàm **destroy** để xóa hết toàn bộ sản phẩm tồn tại trong giỏ hàng.

```
public function del(){
    $this->cart->destroy();
    echo "Done";
}
```

## 6. Cập nhật sản phẩm

Giống như thao tác xóa một sản phẩm, để cập nhật một sản phẩm thì chúng ta cần phải có rowid của sản phẩm đó, rowid giúp chúng ta xác định sản phẩm cần cập nhật.

Để cập nhật sản phẩm có id là 2 thì chúng ta sẽ thiết lập qty cho nó lớn hơn 0, trường rowid sẽ nhận lấy giá trị phần tử rowid mà chúng ta vừa lấy được thông qua vòng lặp.

Các bạn xem đoạn code cập nhật sản phẩm.

```
public function update(){
    $data=$this->cart->contents();
    foreach($data as $item){
        if($item['id'] == "2"){
            $item['qty'] = 10;
            $update = array("rowid" => $item['rowid'], "qty" =>$item['qty']);
        }
    }
    if($this->cart->update($update))
        echo "Update san pham thanh cong";
    else echo "Update san pham that bai";
}
```

## 7. Các hàm khác

Hàm đếm tổng số sản phẩm trong giỏ hàng là hàm `total_items`.

```
public function total(){
    echo 'Hien co '.$this->cart->total_items().' san pham trong gio hang';
}
```

Hàm tính tổng số tiền có trong giỏ hàng là hàm `total`.

```
public function totalmoney(){
    echo 'Tong tien '.$this->cart->total().'$ trong gio hang';
}
```

Hàm `has_options(key)` dùng để kiểm tra rowid có tồn tại hay không, nếu tồn tại thì chúng ta mới lấy được các thông số ở bên trong. Hàm này thường được dùng chung với hàm `product_options(key)` có nhiệm vụ lấy ra các giá trị bên trong trường options. Cả 2 hàm đều phải thông qua vòng lặp foreach mới xuất dữ liệu được.

```
public function product(){
    $data=$this->cart->contents();
    foreach($data as $item){
        if($this->cart->has_options($item['rowid'])){
            foreach($this->cart->product_options($item['rowid']) as $option_name => $option_value){
                echo "<b>$option_name</b>: $option_value<br />";
            }
        }
    }
}
```

## Bài 19: Rewrite URL trong Codeigniter

Trong bài này chúng ta sẽ tìm hiểu cách viết lại đường dẫn trong CI (Rewrite URL). Chức năng này khá quan trọng khi bạn làm web bởi vì nó thân thiện với người dùng và tốt cho SEO. Nếu gặp khách hàng hiểu biết SEO thì có khi họ yêu cầu bạn Rewrite URL theo ý muốn của họ.

Trong bài này mình hướng dẫn các vấn đề chính như sau:

- Tìm hiểu Route trong Codeigniter
- Rewrite URL cho trang sản phẩm
- Rewrite URL cho trang chi tiết
- Rewrite URL cho trang tag
- Rewrite URL cho trang chuyên mục

### 1. Tìm hiểu Route trong Codeigniter

Mọi thứ liên quan đến **Rewrite URL** trong CI đều nằm trong file `application/config/routes.php`, bạn mở file này lên nó sẽ có một số dữ liệu như sau:

```
$route['default_controller'] = 'welcome';
$route['404_override'] = '';
$route['translate_uri_dashes'] = FALSE;
```

Trong đó:

- **default\_controller**: là controller mặc định, nếu bạn truy cập với URL ko có khai báo controller thì nó sẽ gọi tới controller mặc định này. Ví dụ:  
`$route['default_controller'] = 'product/index';`
- **404\_override**: Nếu bạn không khai báo thì nó sẽ lấy trang mặc định lỗi của CI, còn nếu bạn khai báo thì nó sẽ gọi đến controller và action trong đó. Ví dụ:  
`$route['404_override'] = 'error/show404';`
- **translate\_uri\_dashes**: Cái này không phải là route mà nó là một option với giá trị là true hoặc false. Nếu true thì CI sẽ chuyển đổi ký tự gạch ngang (-) thành gạch dưới (\_) và ráp vào controller. Ví dụ bạn có controller tên Product\_shop thì bạn sẽ gõ URL là `http://domain.com/product-shop` sẽ sai, nhưng nếu bạn khai báo `$route['translate_uri_dashes'] = true;` thì sẽ được.

#### Thêm mới route như thế nào?

Để thêm mới một route thì bạn chỉ cần thêm bên dưới cùng của file với cấu trúc:

```
$route['url-tren-trinh-duyet'] = 'controller/action/param1/param2...';
```

Để xử lý tốt **route** trong CI thì bạn phải biết khái niệm về biểu thức chính quy (regular expression). Ví dụ ta có chuỗi 'san-pham/([a-zA-Z0-9]+)/([0-9]+)' thì chúng ta có ba group:

- Group0: toàn bộ chuỗi 'san-pham/([a-zA-Z0-9]+)/([0-9]+)' ký hiệu \$0
- Group1: '([a-zA-Z0-9]+)' ký hiệu \$1
- Group2: '([0-9]+)' ký hiệu \$2

Nghĩa là nếu ta khai báo cặp dấu () thì nó sẽ hiểu là một group. Tuy nhiên route trong CI **không sử dụng Group0**.

### Các ký hiệu hay dùng trong Route

Route trong CI có các ký hiệu như sau:

- (:any) - đại diện cho các ký tự bất kì
- (:num) - đại diện cho các số tự nhiên

Sau đây là một số ví dụ sử dụng routes trong codeigniter:

```
$route['blog'] = 'blog/user';
```

Trong route này thì nếu trình duyệt là <http://domain.com/blog> thì nó sẽ gọi tới blog controller và user action.

```
$route['blog/(:num)'] = 'blog/user/$1';
```

Trong ví dụ này action user trong controller blog sẽ có một tham số truyền vào và giá trị của nó là đằng sau blog/.

- Nếu bạn gõ URL là <http://domain.com/blog/thehalfheart> thì sẽ **sai** vì ký tự đằng sau được khai báo là (:num)
- Nếu bạn gõ URL là <http://domain.com/blog/12> thì **đúng**

Nhưng nếu bạn khai báo như sau thì cả hai URL trên đều đúng:

```
$route['blog/(:any)'] = 'blog/user/$1';
```

### Sử dụng Regular Expression trong route

Bạn có thể sử dụng Regular Expression trong route. Thông thường chúng ta sử dụng các chuỗi regex sau:

- '([0-9]+)' tương đương với (:num)
- '([a-zA-Z0-9]+)' gần tương đương với (:any)

Ở đây chúng ta không liệt kê hết được, các bạn tự tìm hiểu về regular expression nhé.

### Route Callback trong Codeigniter

Chức năng này mới được thêm từ Version CI3X, các versions trước không có.

Thay vì bạn khai báo là \$route['url'] = 'value' thì bạn sẽ khai báo function cho nó với cú pháp như sau:

```
$route['url'] = function() {
    return 'controller/action';
};
```

Tới đây bạn sẽ có thắc mắc là nếu có tham số truyền vào controller thì sao? Đơn giản là tuân theo capturing group nhé. Nghĩa là **group1 tương đương tham số 1, group2 tương đương tham số 2.**

```
$route['product/(:any)-(:num)'] = function($any, $num) {
    return 'controller/action/' . $any . '/' . $num;
};
```

Sau đây là phần áp dụng vào thực tế:

## 2. Rewrite UR cho trang sản phẩm

Trước tiên bạn tạo một **controller** Product như sau:

```
class Product extends CI_Controller{
    public function index($page = ''){
        echo '<h1>Home page</h1>';
        echo 'Page: ', $page;
    }

    public function category($cate_slug = '', $page = ''){
        echo '<h1>Category page</h1>';
        echo 'Slug: ', $cate_slug, '<br/>';
        echo 'Page: ', $page, '<br/>';
    }

    public function tag($tag_slug = '', $page = ''){
        echo '<h1>Tag page</h1>';
        echo 'Slug: ', $tag_slug, '<br/>';
        echo 'Page: ', $page, '<br/>';
    }

    public function detail($post_slug = '', $post_id = ''){
        echo '<h1>Detail page</h1>';
        echo 'Slug: ', $post_slug, '<br/>';
        echo 'ID: ', $post_id, '<br/>';
    }
}
```

Trong controller này, chúng ta tạo 4 action:

- **index**: trang chủ
- **category**: trang sản phẩm theo chuyên mục
- **tag**: trang sản phẩm theo tag
- **detail**: trang chi tiết sản phẩm

### Rewrite trang chủ

Trang chủ ở đây không phải là trang home mà là trang hiển thị danh sách tất cả các sản phẩm nên nó sẽ có URL như sau:

- domain.com/san-pham
- domain.com/san-pham/page/1

Chúng ta sẽ viết hai routes như sau:

```
$route['san-pham/page/(:num)'] = 'product/index/$1';
$route['san-pham'] = 'product/index';
```

Tại sao mình lại đặt route có phân trang ở trên? Tại vì theo quy luật nó sẽ lặp danh sách các URL từ trên xuống dưới, nếu cái nào khớp thì nó sẽ ngưng. Chính vì vậy ta nên đặt có phân trang ở trên và ko có phân trang ở dưới. Bây giờ bạn chạy hai URL trên sẽ thấy kết quả.

### Rewrite trang category

Trang danh sách sản phẩm có dạng sau:

- domain.com/chuyen-muc
- domain.com/chuyen-muc/page/1

Trong controller mình có nhận sẵn hai tham số rồi nên bây giờ ta chỉ viết routes thôi.

```
$route['(:any)/page/(:num)'] = 'product/category/$1/$2';
$route['(:any)'] = 'product/category/$1';
```

Các bạn để ý các ký hiệu \$1 và \$2 nhé, nó là **capturing group** đấy.

### Rewrite trang tag

Tương tự như trang danh sách chuyên mục, nhưng để phân biệt giữa tag và chuyên mục thì trên URL mình sẽ thêm chữ tag nữa.

- domain.com/tag
- domain.com/tag/page/1

Và đây là routes:

```
$route['tag/page/(:num)'] = 'product/tag/$1/$2';
$route['tag'] = 'product/tag/$1';
```

Chúng ta cần đặt route này trên route của category, bởi vì trong route của category có \$route['(:any)'] = 'product/category/\$1'; nên nó sẽ đúng với bất kì trường hợp nào, chính vì vậy nó sẽ dừng và ko duyệt tiếp nên chúng ta phải đặt lên trên category.

```
// Home
$route['san-pham/page/(:num)'] = 'product/index/$1';
$route['san-pham'] = 'product/index';

// Tag
```

```
$route['tag/(:any)/page/(:num)'] = 'product/tag/$1/$2';
$route['tag/(:any)'] = 'product/tag/$1';

// Category
$route['(:any)/page/(:num)'] = 'product/category/$1/$2';
$route['(:any)'] = 'product/category/$1';
```

### Rewrite trang detail

Trang chi tiết sản phẩm ở controller mình nhận hai tham số nên URL sẽ có dạng:

- domain.com/{tieu-de-cua-san-pham}-{id}.html

Chúng ta viết route như sau:

```
$route['(:any)-(:num)\.html'] = 'product/detail/$1/$2';
```

Tương tự bạn phải đặt route này trên cùng nhé, bởi vì nó là cái đặc biệt nên dễ bị trùng với category.

```
// Detail
$route['(:any)-(:num)\.html'] = 'product/detail/$1/$2';

// Home
$route['san-pham/page/(:num)'] = 'product/index/$1';
$route['san-pham'] = 'product/index';

// Tag
$route['tag/(:any)/page/(:num)'] = 'product/tag/$1/$2';
$route['tag/(:any)'] = 'product/tag/$1';

// Category
$route['(:any)/page/(:num)'] = 'product/category/$1/$2';
$route['(:any)'] = 'product/category/$1';
```

Vậy là bạn đã rewrite URL hoàn chỉnh.