Tutorial tự học CoderIgniter được coppy từ nguồn freetuts

Bài 1: Cấu Trúc Folder Codeigniter

Hẳn khi các bạn đã muốn học tới Codeigniter Framwork thì tôi tin chắc khái niệm Web Server bạn đã biết rồi nên tôi sẽ không hướng dẫn cài đặt làm gì. Nếu bạn chưa biết cài đặt thì đọc bài hướng dẫn cài đặt Vertrigo Server

1. Download Và Cài Đặt

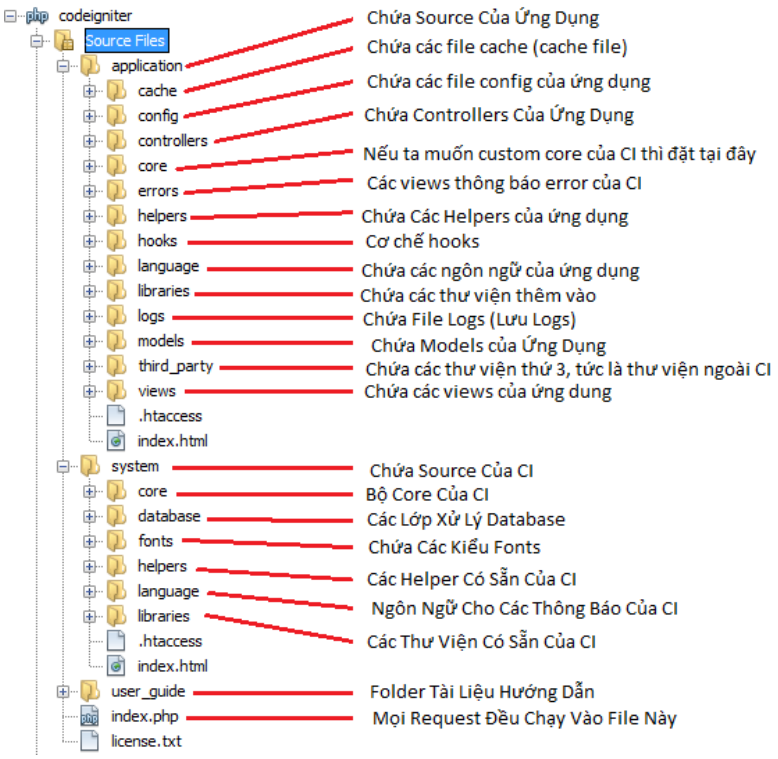
Việc **download và cài đặt Codeigniter** rất đơn giản. Trước tiên bạn vào trang này để download source (chọn version 3.x).

Sau khi download xong bạn sẽ có được 1 file nén `.rar`, bạn vào folder **WWW** của Web Server tạo một folder mới tên là Codeigniter sau đó copy file vừa download vào trong folder mới tạo này và giải nén tại đó.

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 đụng 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 tôi sẽ hướng dẫn các bạn sau.
Trong Application các bạn thấy có 3 folders quan trọng nhất đó là **Controllers, Models và Views**. Đây chính là mô hình MVC nổi tiếng ở thời điểm này, có lẽ có nhiều bạn nếu chưa từng làm qua mô hình MVC sẽ bở ngỡ, thì mình khuyên bạn nên tập viết ứng dụng bằng mô hình MVC viết bằng php OOP thuần thì qua đây các bạn sẽ rất dễ học.

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 tôi sẽ giới thiệu và các bạn sẽ năm ngay lúc đó.

Kết thúc

Trong bài này mình chỉ giới thiệu **sơ lược cấu trúc folder Codeigniter** CI thuần vừa mới download về. Trong seria này mình muốn các bạn đọc theo từng bài, chứ không nên nhảy bậc, đó là nguyên tắt viết tuts của mình

Bài 2: Tạo Controller Trong Codeigniter

Ở bài trước chúng ta đã tìm hiểu qua cấu trúc folder của Codeigniter, vậy thì trong bài này ta bắt đầu tìm hiểu qua mô hình MVC. Bài đầu tiên sẽ tìm hiểu đến controller trong codeigniter. 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 **welcom.php**, bạn xóa file này đi và tạo một file hello.php và điền nội dung vào là:

Trong đó:

```
if (!defined('BASEPATH')) exit('No direct script access allowed');
class Hello extends CI_Controller {
    public function index() {
        echo 'Freetuts.net Hello Controller';
    }
}
```

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 *"Freetuts.net Hello Controller"* 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 'Freetuts.net Hello Controller';
    }
    public function other() {
        echo 'Freetuts.net Other Controller';
    }
}
```

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 se xuất hiện dòng “*Freetuts.net Other Controller*”.

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 Framwork**, 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 'Freetuts.net ' . $message;

    }

}
```

Bạn ra trình duyệt gõ đường dẫn “*localhost/codeigniter/index.php/hello/index/Hello Codeigniter Newbie*“ thì màn hình sẽ xuất hiện chữ “*Freetuts.net Hello Codeigniter*”. Bạn hãy thử thay đổi biến truyền vào để kiểm tra sự huyền diệu của nó.
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 'Freetuts.net ID = '.$id.' AND message  = '.$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 "Freetuts.net ID = 12 AND message = Hello"
- localhost/codeigniter/index.php/hello/index/12 => kết quả là "Freetuts.net ID = 12 AND message = "
- localhost/codeigniter/index.php/hello/index//Hello => kết quả là "Freetuts.net ID = AND 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

    public function __construct() {

        parent::__construct();// Gọi đến hàm khởi tạo của cha

    }

    public function index() {

        echo 'Freetuts.net';

    }

}
```

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**

Lời Kết:

Trong bài này tôi chỉ hướng dẫn các bạn cách sử dụng controller trong codeigniter đơn giản, còn rất nhiều kiến thức khác nhưng tôi nghĩ các bạn chưa cần tới vội nên tôi sẽ đề cập đến trong một bài nâng cao hơn. Bài tiếp theo chúng ta sẽ tìm hiểu cách load view trong codeigniter.

Bài 3: Load View Trong Codeigniter

Trong bài này tôi sẽ hướng dẫn các bạn cách load view trong codeigniter, nội dung bao gồm:

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

1. Tạo Mới Một View Trong Codeigniter

Trong Codeigniter tất cả các Views đều 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:

File application/views/login_view.php:

```
<!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>
```

Vậy là bạn đã tạo được một view trong codeigniter rồi đó.

2. Load View Trong Controller

Có rất nhiều cách load view trong codeigniter, 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:

File application/controllers/Login.php:

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

Controller Login dùng để xử lý login, hàm (action) load_form dùng để load form login cho người dùng nhập dữ liệu và các controller bạn đặt tên nên có động từ ở đầu thì sẽ hay hơn.

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);
        echo $login_form; // Xuất view ra màn hình
    }
}
```

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'

        );

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

    }

}
```

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;

echo '';

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

echo $message;
```

Lời Kết

Trong bài này chúng ta chỉ tìm hiểu một vài cú pháp xử lý view chứ chưa gọi là cao siêu gì, nhưng đó là nền tảng đấy nhé các bạn. Bài tiếp theo chúng ta sẽ tìm hiểu cách load model trong codeigniter.

Bài 4: Load Model Trong Codeigniter

Trong bài này tôi se hướng dẫn các bạn tạo mới một model trong codeigniter, nội dung bao gồm:

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

1. Cấu Hình Database

Trong Codeigniter để kết nối với Database chúng ta phải cấu hình thông tin cho đúng. Bạn ở 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 sẽ nằm trong scope {} của function
    }
}
```

Trong đó News_model là tên model của bạn, 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() {

        $this->load->model('news_model'); // Load model

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

    }

}
```

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() {

        $this->load->model('news_model', 'm_news'); // Load model và đặt tên khác

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

    }

}
```

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 (tuy nhiên tôi không khuyến khích bạn sử dụng việc autoload các file model với tất cả model của project, việc load như vậy sẽ khiến chương trình chạy chậm hơn do file config autoload này sẽ được đọc

trước và load toàn bộ nội dung của nó, chỉ ưu tiên những model được sử dụng lại nhiều lần trên toàn bộ chương trình của chúng ta)

```
class News extends CI_Controller {

    function news_list() {

        // model news_model đã được tự động load trong file config autoload

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

    }

}
```

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() {

        $this->load->model('news/news_model'); // Load model

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

    }

}
```

Lời Kết

Trong bài này chúng ta biết được cách tạo mới một model trong codeigniter, cách gọi model trong controller và cấu hình database trong codeigniter. Bài này vẫn chưa đi sâu vào cách xử lý dữ liệu nhưng bạn đừng chủ quan vì những bài sau tôi sẽ không nhắc đến nó nữa. Bài tiếp theo chúng ta sẽ tìm hiểu thư viện Session trong codeigniter.

Bài 5: Load Library Session

Trong bài này bạn sẽ được học:

1. Giới thiệu library session.
2. Các phương thức làm việc của session.
3. Hàm khởi tạo, sử dụng & hủy session.
4. Các vấn đề mở rộng trong session.

Lưu ý:

Tôi sử dụng Codeigniter version 2.1.4.Tên folder của tôi là citest cho bài

viết này.

Đây là một thư viện khá là quan trọng , trong framework CI (Codeigniter) . Bởi vì chúng ta sẽ sử dụng nó cho rất là nhiều công việc như, quản lý phiên làm việc trong các vấn đề thao tác với login.., sử dụng thao tác session trong giỏ hàng, nếu bạn chưa biết session là gì thì xem mời xem lại bài Session.

Thế nào là session

Trong CI nó được viết dưới dạng class hay còn gọi là library session, điểm nhấn là nó không sử dụng php thuần trong các thao tác. Nó đươc xây dựng dựa trên một cái khóa đặc biệt hay còn gọi là **Encryption key** để tăng tính tương tác và cấp độ bảo mật ứng dụng, để có thể khai báo library session và gọi các phương thức bên trong nó, bạn cần phải vào thư mục file config, tìm file config.php, ctrl+f gõ vào encryption key.

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

Để bắt đầu thao tác với session, ngay phần dấu ” bạn có thể điền bất cứ gì cũng được. ví dụ ở đây tôi điền vào ký tự gì đó chẳng hạn. Xem như tôi đã cấu hình thành công key đặc biệt trong session.

```
$config['encryption_key'] = 'freetuts.net';
```

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

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

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.

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"  =>  "Kaito", "email"  =>  "codephp2013@gmail.com",
            "website"  =>  "freetuts.net", "gender"  =>  "Male",
);
```

Và lúc này tôi đang có 4 giá trị để khởi tạo tòan bộ giá trị tôi dùng cú pháp sau.

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

Khởi tạo xong rồi, muốn sử dụng thì phải làm sao, cũng tương tự như phương thức set_userdata tôi có cú pháp sử dụng session như sau.Tại đây tôi truyền vào tên của session là username.

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

Hàm hủy session

Hủy session có 2 cách tất cả. để hủy một session được chỉ định bất kì ta dùng cú pháp sau,ở đây tôi hủy session username.

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

và cũng giống như cách khởi tạo session với array, tôi cũng có thể hủy session với cú pháp như sau.

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

Với thao tác này tôi 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 chỉ trong một nốt nhạc như sau.Cũng giống như hàm hủy session trong PHP thuần thôi phải không nào.

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

Thực hành Session

Tôi sẽ có một ví dụ nhỏ để cho các bạn hình dung được quy trình làm việc của nó diễn ra như thế nào. Tại thư mục *application/controller* tôi tạo một file có tên là demo.php, vì ví dụ có sử dụng hàm base_url, redirect, nên ngay tại constructor tôi 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" => "Kaito", "email" => "codephp2013@gmail.com",
            "website" => "freetuts.net", "gender" => "Male",
        );
        $this->session->set_userdata($data);
        redirect(base_url(),"index.php/demo/index2");
    }
}
```

Ngay tại action index tôi khai báo biến $data dưới dạng array, tôi sẽ sử dụng lại $data mà tôi khai báo ở phần lý thuyết. Để khởi tạo session, tôi dùng phương thức **set_userdata($data)**, tôi khởi tạo toàn bộ 4 giá trị trên. để kiểm tra xem code của tôi có đúng không, tôi sẽ test session ở action index2. Như vậy chúng ta sẽ tiến hành test bằng cách sau, truy cập vào đường link *localhost/citest/index.php/demo/index*

Nếu trình duyệt tự động chuyển sang index2 thì chúng ta đã khai báo và sử dụng session thành công, để việc test tốt hơn ngay tại index2 tôi sẽ tiến hành show

từng session ra để show session ra ta có phương thức **userdata("ten session")** , tôi sử dụng nội suy biến gọi $tenbien trong dấu "".

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

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.

Username: kaito, Website: freetuts.net, Email: codephp2013@gmail.com

Vấn đề đặt ra ở đây là, nếu tôi muốn lấy tất cả giá trị có trong session thì phải làm sao ? xin thưa rằng CI cũng đã cung cấp một hàm dùng để thao tác yêu cầu mà tôi vừa đưa ra, tại đây tôi sử dụng phương thức **all_data** tôi có cú pháp như sau.

```
public function index2(){
        $user = $this->session->userdata("username");
        $level = $this->session->userdata("level");
        $email = $this->session->userdata("email");
        echo "Username: $user, Email: $email, Level: $level";
        $data = $this->session->all_userdata();
        echo "<pre>";print_r($data);echo "</pre>";
}
```

Tôi in dữ liệu trong cập thẻ pre, nếu màn hình trả về kết quả như sau xem như các bạn đã thành công trong việc lấy toàn bộ giá trị trong session.

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] => kaito
[website] => freetuts.net
[email] => codephp2013@gmail.com

```
[gender]  =>  male
)
```

Sau khi khởi tạo & sử dụng session, tôi tiến hành hủy session như thế nào? rất là đơn giản chỉ với một dòng code bài toán được giai quyết ngay lập tức. Tôi tạo thêm một action có tên là index3 và tôi sẽ hủy toàn bộ session, để kiểm tra tôi echo câu thông báo **"Huy session thanh cong"** chạy link *localhost/citest/index.php/demo/index3* để kiểm tra.

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

Nếu màn hình trả về kết quả sau xem như bạn đã hủy session thành công.

```
Huy session thanh cong.
```

Các vấn đề mở rộng trong session

Cũng giống như các framework khác đều tồn tại một khái niệm gọi là **flashdata** nó có nhiệm vụ gì, bản chất nó là một dạng dữ liệu và nó 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 chung ta F5 trình duyệt.

Để khởi tạo flashdata tôi có cú pháp như sau.

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

Sử dụng nó cũng tương tư như sử dụng session thôi.

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

Tên của flashdata luôn phải bắt đầu với key là flash_ , để cho các bạn hiểu rõ hơn về nó, tôi sẽ có một ví dụ nhỏ như sau.

Tôi sẽ xuất ra câu thông báo **"Khoi tao session thanh cong"** ngay khi tôi vừa chuyển sang action index2. tôi khởi tạo flashdata ở action index tôi đặt tên là flash_open, ở tại action index2 tôi tiến hành show nó ra bằng phương thức flashdata("flash_open") tương ứng với tên flash tôi khai bảo ở action index.

```
public function index() {
    $data = array(
        "username"  =>  "Kaito", "email"  =>  "codephp2013@gmail.com",
        "website"  =>  "freetuts.net", "gender"  =>  "Male",
    );
    $this->session->set_userdata($data);
    $this->session->set_flashdata("flash_open", "Khoi tao session thanh cong");
    redirect(base_url(),"index.php/demo/index2");
}
public function index2() {
     echo $this->session->flashdata("flash_open");
     $user = $this->session->userdata("username");
     $level = $this->session->userdata("level");
     $email = $this->session->userdata("email");
     echo "Username: $user, Email: $email, Level: $level";
     $data = $this->session->all_data();
     echo "<pre>";print_r($data); echo "</pre>";
}
```

Nếu kết quả màn hình trả về như sau, thì các bạn đã thao tác thành công với hàm flashdata. Tôi F5 lại trình duyệt và ngay lập tức cái flashdata của tôi đã bị hủy, suy ra flashdata thường dùng trong các câu thông báo trả về cho người dùng họ đã thao tác thành công một điều gì đó.

```
Khoi tao session thanh cong
[flash:old:flash_open]  =>  Khoi tao session thanh cong
```

Vấn đề mà tôi muốn các bạn chú ý nhiều nhất chính là việc lưu trữ session trong database, có nghĩa là chúng ta sẽ tạo ra môt table dùng để chứa các session, giả sử tôi đang làm việc với giỏ hàng, và tôi cần lưu trữ hàng ngàn session ứng với từng sản phẩm, mà một khi session đã quá tải thì nó không thể lưu trữ nhiều hơn được nữa, nắm bắt được yêu cầu của lập trình viên, CI cũng hỗ trợ chúng ta giải quyết bài toàn này một cách dễ dàng. Chúng ta sẽ phải tạo ra một table tên là **ci_sessions** với các giá trị field như sau, đoạn code này trong user guide của CI có cung cấp sẵn nha các bạn, nhiệm vụ của table này là chứa tất cả dữ liệu mà chúng ta cần chứa, nó sẽ lưu trữ tất cả trong field user_data, nó lấy mọi giá trị của session sau đó convert sang json hoặc một cái chuẩn nào đó của CI, sau đó nó sẽ lưu trữ dữ liệu dưới dạng là text.

```
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`)
);
```

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 thao tác database trong session nha các bạn, nó khiến cho hệ thống của chúng ta bị nghẽn vì phải xử lý thao tác đọc, nhả, nạp dữ liệu quá nhiều, CI chỉ đưa ra giải pháp này cho chúng ta tiện việc thao tác với session hơn thôi. Tôi chỉ sử dụng nó khi nào phải làm việc với giỏ hàng, lưu trữ các thông tin của từng sản phẩm.

Kết

Hy vọng qua bài viết này, các bạn đã có thể hiểu rõ hơn về quy trình làm việc của library session được cung cấp từ CI, bài viết này có thể sẽ dài hơn và sẽ có một vài ví dụ thực tế nửa, nhưng tôi nghĩ nhiêu đây là quá đủ để các bạn có thể control được session một cách dễ dàng. Bài tiếp theo chúng ta sẽ tìm hiểu database trong CI

Bài 6: Load Library Database

Trong bài này bạn sẽ được học:

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

Lưu ý:

Tôi sử dụng Codeigniter version 2.1.4.Tên folder của tôi là citest cho bài viết này.

Tôi sẽ không giới thiệu về library này mà sẽ xoáy sâu vào các thao tác của nó.Đầu tiên tôi cần tạo ra một controller có tên là user và action của tôi là index và tôi sẽ gọi library database ngay tại constructor. Việc tôi gọi lib ngay tại constructor nó giúp cho tôi có thể tái sử dụng lib trong từng action.

Load library database

```
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*, nếu bạn chưa rành về thao tác này xin mời xem lại bài Load Model Trong Codeigniter Tiếp theo tôi cần có một table với tên gọi là user, cái tên bảng tùy các bạn đặt nha.

```
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
```

Các bạn có thể copy đoạn mã này vào **phpmyadmin**, vào phần sql paste vào rồi run thôi nhen, nhưng trước đó hãy tạo ra database trước nhé, nếu không sẽ báo lỗi đấy. Làm đến bước này xem như chúng ta chuẩn bị xong phần kết nối database, tạo table và bước tiếp theo là xử lý nó.

Thao tác Database

Đầu tiên tôi sẽ trình bày 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 tôi dùng thuộc tính như sau.

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

Ứng với câu truy vấn thông thường chình là **mysql_query** đó các bạn. 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>";

    }

}
```

Tôi dung `$data` để hứng lấy kết quả trả về, ở đây result_array tương ứng với mysql_fetch_assoc dùng để lấy toàn bộ record, tôi in dữ liệu trong cặp thẻ textarea để xem kết quả trả về có đúng như sự kỳ vọng của tôi hay không. chạy link *localhost/citest/index.php/user/index*.

```
[0]  => Array (
    [id]  => 1
    [username]  => admin
    [password]  =>  123456
    [level]  =>  2
)
//… (các phần tử mảng sẽ hiển thị tại đây nhưng dài quá nên rút ngắn bớt đi)
```

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, rất là đơn giản phải không nào, tiếp theo tôi 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>";
   }
}
```

Tôi dùng phương thức get tương ứng với câu truy vấn thông thường mà tôi đã viết ở trên, việc sử dụng phương thức này giúp tôi rút ngắn câu lệnh một cách khá là dễ dàng, tôi cũng đếm luôn tổng số record trả về là bao nhiêu bằng phương thức num_rows , và nếu tôi chỉ muốn lấy một record thì làm như thế nào?

```
$query->row_array();
```

Thay vì dùng result_array để lấy tất cả record , thì tôi dùng hàm **row_array** để lấy về một record duy nhất.Các bạn có thể test bằng cách thay đổi giữa row & result. Ngoài ra còn khá là nhiều phương thức hỗ trợ các bạn làm việc tốt với database, tôi sẽ liệt kê một số phương thức mà chung ta hay dùng để xây dựng ứng dụng website.

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

Ví dụ tôi không muốn liệt kê tất cả các field và chẳng hạn như tôi muốn ẩn đi cột password.Tôi sẽ dùng phương thức select trong active record, liệt kê ra các field mà tôi muốn lấy.

```
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>";
   }
}
```

Kết quả trả về như sau, xem như thao tác đã thành công trong việc lấy ra các field được chỉ định.

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

Để sắp xếp id theo chiều giảm dần tôi có cú pháp như sau. order_by tương ứng với order by id desc trong câu truy vấn thông thường.

```
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, tôi cũng có cú pháp như sau. limit() , vị trí đầu tiên là tổng số record tôi muốn hiển thị trong một trang, vi trí thứ 2 chính là vi trí bắt đầu để lấy record. nó khác câu truy vấn bình thường ở chỗ hoán đổi vị trí một tí nhé. ví dụ tôi muốn lấy 7 record, và vị trí bắt đầu của tôi là 0, tôi có limit(7, 0).

```
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 một mệnh đề điều kiện, liệt kê va lấy ra user nào có level = 2 thì tôi có cú pháp như sau where(“level”, “2”), level tương ứng với field, level muốn lấy đương nhiên là bằng 2 rồi.

```
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>";
   }
}
```

Kết quả trả về như sau xem như thao tác đã thành công.

```
Array (
    [0]  =>  Array (
        [id]  =>  1
        [username]  =>  admin
        [level]  =>  2
    )
)
```

Tiếp theo tôi sẽ thực thi mệnh đề điều kiện, để lấy ra id nhỏ nhất & lớn nhất. nếu ở php thuần chúng ta phải viết như sau.

```
SELECT MIN(id) as id FROM user

SELECT MAX(id) as id FROM user

Active Record

$this->db->select_min()

$this->db->select_max();
```

Với active record thì mọi thứ trở nên rất là đơn giản như cân đường hộp sữa ấy. với phương thức select_min(“id”) bài toán xem như được giải quyết.

```
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, tôi sẽ hướng dẫn các bạn ba thao tác quan trọng nhất trong active record, chính là insert, update, delete. Đầu tiên tôi sẽ hướng dẫn cách bạn

cách insert. Để insert một record ta cần phải thao tác với mảng, và cú pháp của câu truy vấn như sau.

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

Đoạn code insert:

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

Kiểm tra bằng cách, chạy link *localhost/citest/index.php/user/index2* , sau đó quay trở lại action index xem kết quả, nếu kết quả trả về như sau, xem như thao tác thành công.

```
Array (
[0]  =>  Array (
    [id]  =>  1
    [username]  =>  admin
    [level]  =>  2
)
[1]  =>  Array (
    [id]  =>  5
    [username]  =>  kaito
    [level]  =>  2
)
```

Tiếp theo để update một record ta cần phải làm như thế nào, đầu tiên để có thể update ta cần phải có một cái mảng, và một mệnh đề điều kiện chỉ định id cần update.

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

Đoạn code update:

```
public function index3() {
    $data = array(
        "username" => "kaito","password" => "kaito123","level" => "1",
    );
    $this->db->where("id", "5");
    if($this->db->update("user", $data)) {
        echo "Update Thanh cong";
    } else {
        echo "Update That bai";
    }
}
```

Tôi tiến hành update id = 5, ứng với username là kaito, tôi cho level của nó là 1, nếu màn hình trình duyệt trả về cho tôi là "Update Thanh Cong" xem như tôi đã hoàn thành việc update một record. kiểm tra bằng cách chạy link *localhost/citest/index.php/user/index3*, quay trở lại action index để xem kết quả và cuối dùng muốn xóa một record tôi cần phải dùng mệnh đề điều kiện, chỉ định ra id cần xóa, ta có cú pháp như sau.

```
$this->db->delete("table")
```

Đoạn code Delete:

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

Tôi xóa id = 5, chạy link **localhost/citest/index.php/user/index4** để tiến hành xóa id tương ứng với username là kaito, quay lại action index xem kết quả nhé, username kaito đã biến mất. Như vậy tôi và các bạn đã cùng nhau tìm hiểu các thao tác active record trong CI, và điều tôi muốn các bạn lưu ý rằng, các thao tác này không nên viết trong controller nó trái với nguyên tắc của CI, bản thân của controller không tương tác trực tiếp với database, để làm việc tốt với nó chúng ta sẽ thao tác trong model nhé.

Thao tác database trong model

Đầu tiên ta cần phải khởi tạo file model nhé, tuyệt đối nhớ dùm tôi là tên file của model không được phép trùng tên với controller, nếu 2 thằng trùng nhau sẽ báo lỗi ngay lập tức, tại thư mục **application/models** tôi khởi tạo file muser.php và khai báo như sau.

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

Xem như khai báo xong model muser rồi nhé, để tiến hành lấy toàn bộ record trong table user ta sẽ viết như thế nào. Tôi tạo một function có tên là listUser. thao tác database thì y chang phần action index của controller user, khác có tí là lấy kết quả thì ta cần return để trả kết quả về, y chang cách viết function trong php thuần.

```
<?php
class Muser 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 tôi tạo action index5 do làm việc với model nên ta cũng sẽ phải thao tác với view để show dữ liệu, tại thư mục **application/views** tôi tạo file list_view.php. nội dung code trong file list_view chỉ là test dữ liệu trong cặp thẻ pre mà thôi.

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

Trong action index5 tôi có đoạn code khá là basic nhen, đầu tiên sẽ tiến hành load model ra, sau đó gọi model muser lên sử dụng, truyền tham số vào view

thế là xong,tham số khi truyền từ controller sang sẽ bị chuyển 1 lần, vì thế cần phải lưu dữ liệu dưới dạng mảng.

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

Kết quả y chang action index nhen, không tin các bạn chạy link *Localhost/citest/index.php/user/index5* để test.

Những điều cần lưu ý

Để có thể nắm bắt và hình dung được cách xử lý database trong CI các bạn cần lưu ý các vấn đề sau giúp tôi.

- Trong bất kì controller không được tạo action có tên là list, nó ứng với core system của CI.
- Thực thi câu truy vấn:

```
$this->db->query()
```

- Đếm số record:

```
$this->db->num_rows()
```

- Lấy tất cả record:

```
$this->db->result_array()
```

- Lấy 1 dòng record:

```
$this->db->row_array()
```

Active Record:

- Thực thi câu truy vấn:

```
$this->db->get("table")
```

- Liệt kê những cột muốn hiển thị:

```
$this->db->select("cols1", "cols2")
```

- Sắp xếp kết quả:

```
$this->db->order_by("id desc")
```

- Giới hạn kết quả:

```
$this->db->limit(7, 0)
```

- Liệt kê dữ liệu với điều kiện:

```
$this->db->where("cols", "var")
```

- Thêm record:

```
$this->db->insert("table", $tenbien)
```

- Sửa record:

```
$this->db->update("table", $tenbien)
```

- Xóa record:

```
$this->db->delete("table")
```

Kết:

Sorry các bạn vì bài viết khá là dài, cũng dễ hiểu thôi đây chính là một library quan trọng mà các bạn cần phải nắm vững nó để có thể xây dựng hoàn chỉnh một ứng dụng website. tôi không thể nào liệt kê và hướng dẫn các bạn toàn bộ các phương thức mà active record cung cấp, để có thể nắm bắt sâu hơn, các bạn có thể tìm hiểu thêm thông qua user guide mà CI cung cấp.

Bài 7: Load Library Pagination

Trong bài này bạn sẽ được học:

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

Lưu ý:

Tôi sử dụng Codeigniter version 2.1.4.

Tên folder của tôi là citest cho bài viết này.

Đây là một library cũng khá là phổ biến, hay được sử dụng trong quá trình xây dựng ứng dụng 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 thực thi thao tác với model, tức là phải đụng tới database & sử dụng active record đấy các bạn.

Load Library pagination

Cũng giống như các library trước, để có thể thao tác với nó chúng ta phải load nó trong controller. Ở bài viết này chắc các bạn đã hiểu rõ cách tạo controller cũng như action trong CI rồi nhé, nên tôi sẽ không nhắc lại nửa, tôi có controller là page và action là index, và trong ví dụ này tôi chỉ sử dụng duy nhất một action , nên tôi sẽ gọi library ngay tại action index, và ngay tại constructor tôi 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 tôi sẽ giải thích sau.

```
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 phải khai báo một số tham số như sau. Và tất nhiên với các tham số này chỉ là basic thôi nhé các bạn. Để áp dụng vào thực tế thì chúng ta cần phải kết hợp thêm rất nhiều thứ để có thể hoàn chỉnh phân trang cho một action trong controller.

```
$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 tôi sẽ dùng lại file muser.php, cái đầu tiên mà chúng ta cần viết, chính là phải tính tổng số record có bao nhiêu trong database. Ở đây active record hỗ trợ chúng ta hàm count_all() dùng để tính tổng record.

```
public function countAll() {

    return $this->db->count_all($this->_table);

}
```

Tiếp theo tôi sẽ load model ngay trong action index, thông tin đầu tiên mà tôi phải cấu hình chính là tổng số record trong database thông qua phương thức countAll mà tôi vừa viết trong model muser, sau đó cấu hình đường dẫn cho controller & action muốn phân trang, để nạp đường dẫn một cách 9 xác, tôi sẽ dùng hàm base_url() được hỗ trợ trong helper url. Và tất nhiên tôi cũng phải cấu hình bao nhiêu record trong một trang tức là phần per_page đó các bạn. Chúng ta phải bắt buộc truyền biến config vào library pagination để có thể thực thi việc phân trang. Cuối cùng tôi echo trực tiếp hàm create_links() trong controller để show kết quả xem chúng ta phân trang có thành công chưa, nên nhớ việc đổ dữ liệu chỉ nên thao tác trong view thôi nhé.

```
public function index() {
        $this->load->model("Muser");
        $config['total_rows']  =  $this->Muser->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 hình dung cho việc tại sao không thấy danh sách thành viên ở đâu cả, trả lời để có thể thấy được danh sách thì chúng ta phải xử lý thêm vài thao tác trong model ví dụ như dùng result_array() lấy toàn bộ record đổ nó ra một cái mảng, truyền tham số vào view & show dữ liệu ngay trong view. Tất nhiên là phải giới hạn kết quả, mà muốn giới hạn kết quả thì phải dùng limit() và nhớ rằng trong CI phần limit nó khác một xíu chính là lấy từ tổng số record rồi mới đến vị trí bắt đầu.

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

Muốn thao tác trong việc show danh sách thành viên 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 đươc vị trí bắt đầu thì ta phải get được cái đường dẫn của nó, chính là get cái id đó các bạn, trong CI nó có một helper hỗ trợ chúng ta lấy được đường dẫn và khuyến cáo không được sử dụng phương thức $_GET, vì nó không an toàn. phương thức thay thế nó chính là $this->uri->segment(tham số), để có thể lấy đường dẫn một cách chính xác tôi có cách đếm như sau , ví dụ link **localhost/citest/index.php/page/index/???** , page = controller ứng với vị trí số 1, index = action , và tham số vị trí $start chính là vị trí thứ 3.

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

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

Nhiệm vụ của getList là đổ về dữ liệu những cái mảng chứa thông tin user và chúng ta sẽ phải làm việc trong view, tôi tạo file page_view.php, và bên trong nó tôi cũng in dữ liệu trong cặp thẻ pre.

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

```
echo $this->pagination->create_links();
```

Nếu kết quả trả về như trên, chúng ta đã tiến hành phân trang và kiểm soát record trả về 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 ›

Các vấn đề mở rộng trong pagination

- Độ phân giản của link: khi bạn click vào link 3 thì phía nó hiển thị như sau 1 2 **3** 4 5.

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

- Sử dụng theo từng số trang: Bình thường link sẽ lấy tham số là vị trí bắt đầu tức là 3 , sau đó click trang 2 nó sẽ nhảy ra tham số là 9, sử dung phương thức này khi bạn click vào trang 2 nó sẽ hiển thị link như sau localhost/citest/index.php/page/index/2. tức là sử dụng 9 xác vị trí của trang.

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

Ngoài ra còn rất nhiều phương thức hỗ trợ chúng ta trong việc phân trang, các bạn tự tìm hiểu sâu hơn nhé, tôi chỉ hướng dẫn đến đây thôi.

Kết:

Để có thể phân trang cho một action nào đó, các bạn cần phải nắm rõ những phương thức mà tôi vừa nêu ở trên, thật ra nó rất là đơn giản, nhưng cái user guide của nó viết chung chung quá, khiến quá trình tìm hiểu của chúng ta gặp khó khăn, nhưng chỉ cần làm giống những gì tôi vừa trình bày ở trên là có thể phân trang bất cứ action nào mà bạn mong muốn.

Bài 8: Tìm hiểu Helper Url Và Form

Trong bài này bạn sẽ được học:

1. Giới thiệu về helper
2. Làm việc với form helper
3. Các helper thông dụng

Lưu ý:

Tôi sử dụng Codeigniter version 2.1.4.Tên folder của tôi là citest cho bài viết này.

Trong Codeigniter Framework nó chia ra 2 khái niệm đó là helper & library, vậy câu hỏi đặt ra ở đây. Helper nó là cái gì ? nó là những function được xây dựng sẵn nhằm hỗ trợ lập trình viên trong quá trình xây dựng ứng dụng website bằng nền tảng CI, có rất là nhiều helper được khởi tạo sẵn. Vậy library khác

helper ở điểm nào, xin thưa rằng 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 là với helper chúng ta có thể gọi nó 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**

Load Helper

Cũng giống như library, để có thể sử dụng tất cả các hàm trong các helper thì việc đầu tiên chúng ta phải gọi chúng ra, và dĩ nhiên là phải gọi ngay constructor trong controller để có thể tái sử dụng helper đó với toàn bộ action, có 2 cách để load helper, từ đó chúng ta hoàn toàn có thể gọi các hàm ở ngay controller và view một cách dễ dàng.

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

Url Helper

Trong helper này có rất là nhiều phương thức, nhưng không phải cái nào là cũng xài đâu nhé, bản thân framework nó rất là nặng, cho nên vì thế tôi khuyên các ban chỉ nên dùng vài phương thức sau đây.

1. base_url(): Chỉ định đường dẫn tuyệt đối, khai báo trong file config.php.
2. site_url(): Nó gần giống với base_url./li>
3. anchor(): Khai báo một đường link
   vd: **link -> echo anchor(‘news/local/123’, ‘My News’, ‘title = ”News title”‘).**
4. url_title(): Hỗ trợ Url Friendly dạng hom-nay-troi-co-mua.
5. 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 đươc 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 tôi chỉ dùng 2 phương thức redirect & base_url để làm tốt công việc cấu hình đường dẫn tuyệt đối , chuyển trang khi tôi thao tác xong một hành động nào đó.

Form Helper

Đây là một helper cũng khá là hay nó giúp các bạn nhanh chóng tạo ra các form dùng để nhập liệu một cách nhanh chóng, đọc cái user guide của nó cũng khiến tôi phải chóng mặt vì nó khá là rối và khó hình dung, tôi cũng phải mất vài ngày mới tìm hiểu xong nó nhưng theo quan điểm của tôi, hãy sử dụng nó tùy lúc thôi nhé, còn không cứ dùng html mà tạo form bình thường thôi tại sao tôi lại nói vậy, hãy đọc tiếp bài viết tự các bạn sẽ hiểu.

Tôi sẽ tạo ra môt controller tên là form, gọi helper url & form ra nhé, vì giờ đang tìm hiểu cách sử dụng nó mà. Do chúng ta gọi một lần 2 helper nên tôi sử dụng cách 2 để tiến hành gọi helper, chúng ta sẽ gọi các phương thức trong form helper ngay trong phần view, trước khi bắt đầu ví dụ này tôi sẽ liệt kê nhưng 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 tyle 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

tôi muốn các bạn nên sử dụng chính là tạo ra một cái mảng và chứa các thông số cho tiện. ví dụ tôi tạo ra một form input với name là kaito, value là freetuts.net thì trong CI nó chỉ chúng ta viết như sau,giả sử tôi muốn size là 25, id với style là gì gì đó thì chẳng lẽ phải viết dài dòng sao, thay vì viết dài dòng thì tôi sẽ hứng tất cả thông số trong một cái mảng, sau đó truyền $tenbien vào trong form_input là xem như xong. Các bạn có thể xem đoạn code phía dưới và sẽ dễ dàng hình dung ra được cách khai báo các phương thức trong form helper, và bản thân tôi thì thấy nó phức tạp quá, tôi thích sự đơn giản

và chung thủy nên cứ html mà làm tới thôi , viết kiểu này cũng hay và oách nên tùy lúc sử dụng thôi nhé, lạm dụng là vào viện đấy.

```
echo form_input('kaito', 'freetuts.net')
```

Full Code:

```
<?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>
```

Chạy code trên và viewsource để cảm nhận được sự huyền diệu của form helper.

Các Helper thông dụng

Tôi sẽ liệt kê một số helper mà lập trình viên hay sử dụng trong quá trình làm việc với CI, và khuyến cáo là các bạn chỉ nên tìm hiểu cho biết chứ đừng lạm dụng nó nhé, bản thân CI nó đã khá là chậm nên hãy tìm cách tối ưu nó chứ đừng làm nó chạy chậm thêm bằng việc nhồi nhét các helper.

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

Hãy sử dụng helper đúng cách và đúng lúc, hãy thông minh khi làm việc với bất cừ Framework nào, trong loạt bài tới tôi sẽ hướng dẫn các bạn cách tự xây dựng một helper trong CI.

Kết:

Mọi thứ nên bắt đầu từ những điều đơn giản và đừng làm nó phức tạp hơn, đó chính là những gì mà tôi muốn nhắn nhủ với các bạn thông qua bài viết này, tôi không cấm các bạn sử dụng helper , nhưng hãy sử dụng nó một cách hợp lý và hiệu quả, bài viết này tôi không diễn đạt code quá nhiều vì cách sử dụng form helper nó rất là đơn giản, user guide của nó viết thì thấy rối vậy thôi. Tôi tìm hiểu và đúc kết ra được vài cách và cách trên là cách tối ưu nhất mà các lập trình viên hay sử dụng.

Bài 9: Tìm Hiểu Library Form Validation

Trong bài này bạn sẽ được học:

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

Lưu ý:

Tôi sử dụng Codeigniter version 2.1.4.Tên folder của tôi là citest cho bài viết này.

Kết thúc bài học trước, chúng ta đã cùng nhau tìm hiểu xong khái niệm cơ bản về form helper , ở bài này sẽ đi sâu vào vấn đề thực tế hơn trong việc kiểm soát dữ liệu người dùng nhập vào textbox, tạo ra những tập luật để kiểm duyệt dữ liệu, ví dụ dữ liệu này bắt buộc phải là con số, email này phải đúng định dạng…vv giống như các bài trước để có thể thao tác với bất kỳ library nào trong CI, bắt buộc các bạn phải load libs của nó ra.

Load library form validation

Cho tới bài viết này, chắc hẳn các bạn đều đã biết rõ cách load một library cũng như cú pháp gọi tên từng phương thức để sử dụng, cho nên tôi sẽ không nhắc lại nửa, nếu chẳng may các bạn quên thì có thể xem lại bài Session.

```
$this->load->library('form_validation')
$this->form_validation->ten_phuong_thuc()
```

Để có thể nắm bắt vấn đề tốt hơn khi thao tác với **form validation** tôi sẽ liệt kê một số phương thức hay dùng. thứ nhất là phương thức dùng để tạo ra những cái luật đó là **set_rules**, tại đây chúng ta phải truyền vào 3 giá trị, tên của textbox, tên khi xuất ra thông báo lỗi, và giá trị quan trọng nhất chính là tập luật mà các bạn muốn đưa ra, ngoài ra có thể đưa ra nhiều tập luật bằng cách sử dụng dấu |.

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

1. **required:** Yêu cầu phải nhập liệu không được để trống
2. **matches:** Yêu cầu password phải trùng khớp nhau
3. **min_length:** Giới hạn bao nhiêu ký tự khi nhập vào
4. **max_length:** Giới hạn bao nhiêu ký tự khi nhập vào
5. **numeric:** Yêu cầu trong textbox phải nhập liệu là con số
6. **valid_email:** Email nhập liệu bắt buộc phải đúng định dạng
7. **xss_clean:** Xóa XSS của input, bảo mật

Ngoài ra trong user guide của CI còn rất nhiều tập luật khác, các bạn tự tìm hiểu nhé, tôi chỉ liệt kê một số tập luật thường dùng trong thực tế mà thôi, ngoài ra còn một phương thức rất là quan trọng nếu thiếu phương thức này thì xem như các tập luật phía trên bị vô hiệu hóa, vì phương thức **run()** có nhiệm vụ kiểm tra các tập luật trên có hợp lệ hay không, nếu không hợp lệ nó sẽ trả về kết quả là FALSE, tức là nó sẽ kiểm tra phương thức set_rules nếu một trong tất cả số set_rules mà bạn khai báo bị lỗi thì nó sẽ trả về FALSE đồng thời trả về một cái view nào đó, đồng thời xuất câu thông báo lỗi ra bằng phương

thức **validation_errors()**, phần lý thuyết khô cằn đã trôi qua, để các bạn dễ hình dung hơn chúng ta sẽ bắt đầu với một ví dụ nhỏ.

Kiểm tra nhập liệu với form validation

Tôi sẽ dùng lại controller & view của bài form helper, và trong ví dụ này chúng ta sẽ kiêm 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($user)."<br />";
    echo form_label("Password: ").form_password($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>
```

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 tập luật mà chúng ta đưa ra như sau, do ví dụ này chỉ có một action nên việc đầu tiên tôi 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 giá trị xuất ra thông báo lỗi sẽ là Full Name, tôi muốn người dùng bắt buộc phải nhập tên vào textbox, vì thế tôi sẽ dùng tập luật required, và tôi không muốn cho họ nhập vào một cái tên quá dài , tôi ép họ phải nhập không được ít hơn 6 ký tự , tôi có phương thức min_length[6], nhớ nếu sử dụng hơn một tập luật thì phải dùng dấu | khai báo tập luật thứ hai, và để kiểm tra xem email có đúng định dạng không tôi sẽ dùng phương thức valid_email, sau khi khai báo tập luật xong chúng ta tiến hành dùng phương thức run() kiểm tra xem các set_rules có hợp lệ hay không, nếu một trong 3 cái yêu cầu trên bị lỗi thì ngay lập tức nó sẽ trả về là FALSE, và tất nhiên chúng ta sẽ đi load lại cái view của form, ngay tại cái view chúng ta sẽ tiến hành xuất ra thông báo lỗi bằng 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
<?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();
```

Toi chạy link *localhost/citest/index.php/form* để kiểm tra, nếu tôi không nhập bất cứ thứ gì vào textbox mà tôi 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 như ngay textbox fullname tôi nhập ít hơn 6 ký tự và email tôi điền loạn xà ngầu 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 thật là dễ dàng phải không các bạn, còn một vấn đề đặt ra ở đây là, nếu tôi muốn việt hóa các câu thông báo lỗi khi xuất ra thì có được không, hoàn toàn có thể nhé.

Việt hóa language form validation

Nói việt hóa cho nó sang chứ thật ra là sửa code trong phần system của nó, rất là đơn giản, ngay tại folder system các bạn tìm đến folder language mở nó ra thấy ngay file form_validation_lang.php, mở file này ra bên trong nó

có đoạn code như sau, để ý cái **%s** nó chính là cái tên bạn 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.";
```

Tôi thử việt hóa phần bắt buộc nhập liệu tức là phương thức required đấy các bạn, nhớ là không được bỏ dấu nhé các bạn, viết không dấu, còn về phần viết có dấu tự các bạn tìm hiểu thêm :), tôi f5 lại trình duyệt và thấy kết quả trả về như sau.

```
$lang['required']        =  " %s yeu cau khong duoc bo trong.";
```

Full Name yeu cau khong duoc bo trong.

Password yeu cau khong duoc bo trong.

Email yeu cau khong duoc bo trong

Xem như tôi vừa hoàn thành việt hóa câu thông báo lỗi xuất ra từ form validation, các câu lỗi còn lại các bạn tự việt hóa theo văn phong của mình, chúc các bạn thành công.

Kết:

Bài viết này chỉ tập trung hướng dẫn các bạn các thao tác căn bản nhất khi làm việc với form validation, ở các bài viết khác tôi sẽ hướng dẫn các bạn một số phương thức nâng cao hơn, cũng như tạo ra một form login check lỗi nhập liệu và check lỗi trùng tên truy cập, thao tác với **database & session**, chào tạm biệt và quyết thắng.

Bài 10: Tìm hiểu library upload

Trong bài này bạn sẽ được học:

1. Giới thiệu về Library upload
2. Thao tác & làm việc với nó

Lưu ý:

Tôi sử dụng Codeigniter version 2.1.4.

Tên folder của tôi là citest cho bài viết này.

Chào mừng các bạn đã quay trở lại freetuts.net, đã lâu rồi tôi không viết tuts về CI căn bản, vì một số lý do khách quan cũng mong các bạn rộng lòng thông cảm, ở các bài trước chúng ta đã cùng nhau tìm hiểu qua các thao tác kết nối model, phân trang và sử dụng helper form & url. Tiếp tục loạt bài trong serie CI căn bản, chúng ta sẽ cùng nhau tìm hiểu một thư viện tiếp theo đó là **File Upload**, đây là môt trong những thư viện mà tôi nghĩ rằng nó cũng khá là phổ biến và quan trọng, vì trong thực tế bất kể chúng ta xây dựng một trang web này cũng phải upload hình ảnh lên.

Giới thiệu library upload:

Bất kể khi bạn làm việc với bất kì framework nào thì đều cũng phải trải qua quá trình upload file và với Codeigniter cũng không ngoại lệ, các sáng lập viên đã xây dựng cho chúng ta một library khá là hay với thao tác hỗ trợ chúng ta tối đa trong việc upload hình ảnh, vậy với **Library upload** chúng ta sẽ làm được những gì với nó.

1. Hỗ trợ định dạng file được chỉ định khi upload (Ví dụ: Cho phép upload với 3 định dạng hình ảnh).

2. Tùy chỉnh độ rộng & cao tấm hình đc phép upload một cách dễ dàng.
3. Giới hạn dung lượng file hình ảnh được upload.
4. Hỗ trợ chúng ta resize ảnh theo nhu cầu.
5. 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 , tôi thấy rằng đã là quá good rồi phải không nào, vừa bảo mật, vừa resize ảnh, đóng dấu bản quyền luôn còn dễ dàng cho phép độ rộng và cao của tấm ảnh được upload lên, theo tôi như thế là quá đủ cho thao tác upload hình ảnh rồi. Để không làm mất thời gian của các bạn, chúng ta sẽ tìm tiến hành tìm hiểu cách làm viêc, cũng như cấu hình library diễn ra như thế nào.

Cấu hình library upload:

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

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

Và sau đó chúng ta sẽ có một số thông tin cấu hình như thế này.

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

Với 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 chúng ta có thể test xem mình upload có thành công hay không, sẽ có câu lệnh debug như sau, với câu lệnh này chúng ta sẽ dễ dàng lấy ra được tên của tấm hình, file tạm của nó...vv.

```
$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 phải xuất ra câu thông báo lỗi, để người dùng có thể hiểu rằng , hình họ vừa upload bị lỗi thao tác như là định dạng không đúng, kích thước hình vượt quá yêu cầu cho phép, thì lúc bấy giờ chúng ta cần phai sự dụng thêm một câu lệnh nửa đó là.

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

Như vậy thì chúng ta sẽ chỉ cần nhớ vỏn vẹn 3 lệnh và một số thông tin cấu hình ở phía trên thì chúng ta hoàn toàn dễ dàng upload hình ảnh một cách thoải mái nhất. Và để cho các ban dễ hình dung hơn thì chắc chắn tôi sẽ phải demo một ví dụ nhỏ rồi.

Thực hành upload file từ máy tính lên web server:

Đến với bài này thì chắc chắn các bạn cũng đã biết cách tạo ra một controller rồi, nên tôi sẽ không nhắc lại vấn đề này nữa để tránh mất thời gian hơn tôi đã chuẩn bị sẵn cho chúng ta một controller upload với các tham số như sau, do trong ví dụ này tôi sử dụng helper form & url nên tôi sẽ tiến hành load nó ra ngay tại constructor. Tạo thêm một folder uploads ngang cấp với folder application, tôi cũng tạo ra action index và load cái form ra.

```
class Upload extends CI_Controller {

    public function __construct(){

        parent::__construct();

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

    }

    public function index(){

        $this->load->view("upload_view");

    }
```

Tôi tạo trong folder view một file có tên là upload_view, và bắt đầu thao tác với form helper 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** cũng giống như , *multipart*/form-*data* trong html bình thường thôi nhé các bạn., tôi tạo $upload với name là img, ở form_submit cũng có name là ok, dùng để check thao tác của người dúng với form, và với hàm base_url thì tôi sẽ chỉ ra đường dẫn tuyệt đối như sau, controller là upload & action sẽ là doupload, thực hiện xong 2 thao tác này 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, tôi sẽ load cái library ra phía dưới các thông tin, kèm theo đó sẽ là $config chính là các cấu hình mình đã khai báo ở phía trên, tất nhiên chỉ như vậy thôi là chưa đủ đâu nhé, để có thể upload file thì cần phải có lệnh để thực thi thao tác. Truyền vào tên file đã tạo ở phần view, hoàn thành xong bước này thì chúng ta có thể upload rồi đấy, upload xong thì cần phải show ra toàn bộ thông tin của tấm ảnh, tôi đặt một tên biến bất cừ cho phương thức show info tấm hình, tôi sẽ kiểm tra dữ liệu trong cặp thẻ pre. Còn ngược lại nếu như mà quá trình upload thất bại thì chúng ta sẽ xuất ra câu thông báo lỗi, khi đó sẽ phải tạo ra một cái biến để đổ câu thông báo lỗi sang view, tôi tạo biến

`$data['errors'] = $this->upload->display_errors();`,

vậy ngay vị trí của controller index chúng ta phải khai báo biến $errors bằng rỗng để nó ẩn câu thông báo lỗi chỉ khi nào có lỗi thì mới show ra thôi. Nếu errors khác rỗng thì chúng ta mới xuất thông báo lỗi ra. Ra trình duyệt và gõ *localhost/citest/index.php/upload* để test nào.

```
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 check lỗi, thì tôi sẽ chọn thử một file .doc nào đó tiến hành nhấn nút submit xem kết quả trả về ntn.

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 viêc cấu hình định dạng file cho phép upload thì các bạn đã phần nào hạn chế đươc việc hacker upload file shell hoặc các mã độc khác lên website của mình.

Full Code Upload

1. 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);
            }
        }
    }
}
```

2. 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 $errors;
        echo form_open_multipart(base_url()."index.php/upload/doupload");
        echo form_label("Avartar: ").form_upload($upload)."<br />";
        echo form_label(" ").form_submit("ok", "Upload");
        echo form_close();
    ?>
</body>
</html>
```

Kết thúc bài học:

Xem như chúng ta vừa tìm hiểu xong library upload trong CI, các bạn thấy thao tác của nó cực là basic và dễ sử dụng đúng không nào, ở bài tiếp theo chúng ta sẽ cùng nhau đi sâu vào việc kiểm soát việc resize hình ảnh, watermark cho hình, làm một lúc 3 thao tác là upload ảnh, resize ảnh, watermark. Còn bây

giờ việc của các bạn là xoắn áo lên và gõ lại những gì mà tôi vừa trình bày ở trên, chúc các bạn học tốt và đừng quên ủng hộ freetuts bằng cách like & share nếu cảm thấy nội dung này hay và thật sự bổ ích, cảm on các bạn đã đọc hết bài viết này của tôi.

Bài 11: Tìm Hiểu Library Image

Trong bài này bạn sẽ được học:

1. Giới thiệu về Library Image
2. Thao tác & làm việc với nó

Lưu ý:

Tôi sử dụng Codeigniter version 2.1.4.

Tên folder của tôi là citest cho bài viết này.

Kết thúc bài trước, chúng ta đã hoàn thành khá xuất sắc phần upload hình ảnh trong CI, dựa vào library upload đó chúng ta hoàn toàn 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, file đó có hợp lệ hay không, dung lượng tấm hỉnh tối thiểu là bao nhiêu mb. Như vậy là chưa đủ, khi các bạn bắt tay vào xây dựng hoàn chỉnh một ứng dụng website sẽ phát sinh ra một số vấn đề nửa, ví dụ như làm thế nào để tạo những cái hình nhỏ hơn, thay thế cho hình đại diện trong từng sản phẩm, bời vì không thể nào nén tấm hình lớn lại, đều đó làm cho tốc độ load rất là chậm, thế nên chúng ta cần phải tạo ra các hình nhỏ mà mọi người hay gọi nó là hình **thumbnail**, để làm được điều tôi vừa nói thì chúng ta sẽ cùng nhau tìm hiểu khái niệm **library image** trong codeigniter.

Giới thiệu library Image:

Với library image thì chúng ta hoàn toàn có thể làm được những gì nào?

1/ Resize Image : Từ hình mặc định được upload lên sẽ cắt nhỏ thành hình thumbnail.

2/ Image Crop: Làm việc với thao tác này tức là mình sẽ cắt hình.

3/ Image Rotate: Làm việc với thao tác là xoay hình ảnh.

4/ Image Watermark: Chèn logo hoặc đoan text bất kì vào tấm hình.

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

Cấu hình library image:

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

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

Và phải cần có 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;
```

Nếu tôi giải thích hết toàn bộ thuộc tính ở phía trên thì sẽ mất rất nhiều time của các bạn, nên tôi chỉ tập trung nói về các thuộc tính quan trọng mà các bạn cần phải hiểu rõ về nó trong quá trình làm việc với library image. thuộc tính `$config['image_library']` chính là một dạng library image đặc biệt và chúng ta có rất nhiều library giống như thế,ví dụ **GD, GD2, ImageMagick, NetPBM**, tuy nhiên thông dụng và phổ biến nhất vẫn là GD2, và bản thân library này cũng cấu hình mặc định là GD2 rồi đấy, ngoài ra chúng ta còn phải chỉ ra đường dẫn tấm hình ban đầu mà chúng ta đã upload lên, bởi vì chúng ta cần phải tạo ra hình nhỏ với thuộc tính `$config['source_image']` , còn có thiết lập cho chất lượng tấm hình đó có thay đổi hay không.

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 nho nhỏ, giúp các bạn hình dung vấn đề tốt & hiệu quả hơn là cứ đi tìm hiểu lý thuyết khô cằn.

Thực hành resize với library image:

Tôi sẽ sử dụng lại controller upload mà tôi 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", $data);
    }
    public function doupload(){
        if($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);
            }
        }
    }
}
```

Khi đã upload thành công tức là ở ngay vị trí `$this->upload->do_upload("img")` thì chúng ta sẽ có một số thao tác cũng nhưng muốn resize hình ảnh, chúng ta sẽ cấu hình thông số ngay tại vị trí này, đầu tiên sẽ phải load library imge ra, sau đó tiến hành cấu hình thông tin cho nó. do chúng ta cần phải có tên gốc của tấm hình mới có thể resize được , tôi dùng phương thức `$check  =  ['file_name']` nối chuỗi phía sau đường dẫn tấm hình, sau đó tôi 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 phải nạp tất cả thông tin này bằng cú pháp `$this->image_lib->initialize($config)` , sau đó chúng cần phải gọi lệnh thực thi thao tác resize với cú pháp `$this->image_lib->resize()`, với hàm này nó 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 kì tấm hình nào đó và vào folder uploads xem điều gì sẽ xảy ra nhé. Ví dụ ở đâu tôi upload tấm hình lighhou.jpg, và lúc kiểm tra tôi thấy ngoài hình gốc là lighthou thì còn xuất hiện thêm file lighthou_thumb.jpg, nó sẽ dựa vào độ rộng hoặc độ cao của tấm hình và xem xét cân đối khi thực hiện thao tác resize.

Lý do tại sao chúng ta phải sử dụng hình thumbnail, xin thưa nếu một trang web mà chỉ sử dụng tấm hình gốc ban đầu sẽ làm cho website của bạn load rất chậm, dung lượng của hình gốc là rất nặng, và chúng ta có hàng chục tấm hình cho sản phẩm, mà sản phẩm nào cũng phải load hình ảnh gốc đại diện thì rõ ràng website của các bạn sẽ load rất là chậm. Đó là lý do chúng ta buộc phải sử dụng hình ảnh thumbnail để giảm thiểu thiệt hại băng thông cho website.

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", $data);
    }
    public function doupload(){
        if($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);
            }
        }
    }
}
```

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 $errors;
        }
        echo form_open_multipart(base_url()."index.php/upload/doupload");
        echo form_label("Avartar: ").form_upload($upload)."<br />";
        echo form_label(" ").form_submit("ok", "Upload");
        echo form_close();
```

```
    ?>
</body>
</html>
```

Kết thúc bài học:

Ở bài tiếp theo chúng ta sẽ làm quen với một thao tác trong library image , chính là **watermark**, hy vọng qua bài viết này sẽ giúp các bạn dễ dàng hơn trong việc tối ưu việc upload hình ảnh, đây là một trong những bước khá là quan trọng trong quá trình xây dựng bất kì website nào đó.

Bài 12: Đóng dấu watermark image

Kết thúc bài trước chúng ta đã tìm hiểu xong thao tác resize hình ảnh, và trong bài hôm nay chúng ta sẽ cùng tìm hiểu một khái niệm trong libs image đó chính là **watermark**,và mấu chốt tôi vẫn muốn nhấn mạnh nhất đó là chính là làm như thế nào để có thể thực hiện một lúc 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 là bởi 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 lai địa chỉ website của các bạn. Làm điều đó chỉ để bảo vệ quyền sỡ hữu trí tuệ của các bạn mà thôi.

Cấu hình watermark image:

Với watermark do CI cung cấp thì chúng ta có 2 giải pháp, đó là dóng dấu bằng text hoặc chèn logo lên hình. Trong bài này thì chúng ta sẽ tìm hiểu chèn một lúc một trong hai kiểu đóng dấu, tôi sẽ đi từ dễ đến khó vì thế kiểu đóng dấu hình bằng text sẽ được tìm hiểu đầu tiên.

Đây là những thông tin cần thiết cho việc cấu hình để dóng dấu theo dạng text, tôi sẽ giải thích một số giá trị quan trọng như là kiểu text, màu sắc của text, vị trị text sẽ hiển thị trên tấm hình, kiểu font cho text, và nếu bạn muốn chọn font khác cho text thì vào folder system đến folder font cho font bạn lựa chọn, sau đó vào sửa code là ok ngay thôi, vị trí sẽ nằm ở cuối tấm hình và text sẽ được canh giữa. và quan trọng là giá trị padding nên là 0, vì nó là khoảng cách của tấm hình đến các khung xung quanh, sau khi khai báo hoàn tất chúng ta tiến hình thực thi thao tác watermark bằng

lệnh $this->image_lib->watermark(), xem như chúng ta vừa khai báo xong phần watermark theo dạng text, do tôi đang chỉ muốn test phần watermark nên phần code resize tôi làm ở bài trước tạm thời comment khóa nó lại.

```
$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();
```

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['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 rồi đấy, còn chưa thấy tức là các bạn khai báo sai ở chỗ nào đó,xem kỹ code hoặc copy code của tôi bỏ vào chạy thử. Tiếp theo chúng ta sẽ tìm hiểu và cấu hình chèn logo cho watermark, xem userguide của CI và tôi có một số thông tin cấu hình như sau. 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, chúng ta chỉ thay đổi thông tin khai báo là có thể dễ dàng chèn logo vào hình ảnh.

```
$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['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);
            }
        }
    }
```

Sau khi upload xong thì kiểm tra xem logo đã chèn vào hình hay chưa, tôi nghĩ là các bạn sẽ hoàn toàn làm được, vì thao tác này khá giống với thao tác add text vào hình, việc sử dụng watermark trong CI thật sự là quá đơn giản phải không các bạn ? nhưng điều tôi muốn nói với các bạn rằng, làm sau thực hiện được cùng lúc 3 việc là upload hình , resize hình, watermark và CI có làm được hay không, câu trả lời là có, trong quá trình tìm hiểu về phần **library images**, tôi có trao đổi với vài người bạn và họ cho tôi vài giải pháp trong việc xử lý vấn đề này, và tôi cũng đã chọn cho mình cách tối ưu nhất để đơn giản hóa 3 thao tác thực hiện trong một phiên upload duy nhất.

Vấn đề mở rộng & tối ưu trong library image:

Sau khi tiến hành resize xong thì chúng ta phải clear các thông tin mà chúng ta đã cấu hình ở phía trên, thì library image có cung cấp cho chúng ta câu lệnh `$this->image_lib->clear()`, sau khi clear xong thì các bạn phải nhớ là vì `$config` chúng ta sẽ sử dụng lại với phần watermark nên buộc chúng ta phải

hủy sự hoạt động của nó bằng hàm unset($config) , 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 tôi muốn các bạn lưu ý nhất chính 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 nào làm môt lúc 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);
        }
    }
}
```

Kết thúc bài học:

Hiện tại tôi nghĩ đây là cách đơn giản và nhanh nhất nếu các bạn muốn sử dụng một lúc 2 thao tác chỉ với một lần upload thì tôi khuyên nên dùng cách này, còn nếu bạn có giải pháp khác thì hãy chia sẽ với tôi bằng cách comment ở phía dưới bài viết này. Chào tạm biệt và chúc các bạn học tốt.

Bài 13: Tìm Hiểu Helper Date

Trong bài này bạn sẽ được học:

1. Giới thiệu về helper date.
2. Các hàm thông dụng.

Lưu ý:

Tôi sử dụng Codeigniter version 2.1.4.

Tên folder của tôi là citest cho bài viết này.

Hôm nay chúng ta sẽ tiếp tục tìm hiểu sâu hơn về các **helper** mà CI đã cung cấp, nếu bạn nào chưa hiểu rõ khái niệm về helper thì xin mời xem lại bài helper này , do bài viết này chỉ tập trung vào các hàm thông dụng và trong thực tế thì hầu như là ít lập trình viên nào sử dụng **helper date**, nên tôi sẽ không nói quá nhiều về cách thức hoạt động của nó.

Giới thiệu về helper date:

Khi các bạn bắt tay xây dựng hoàn chỉnh một ứng dụng website thì ít nhất chúng ta phải xử lý một số chức năng có liên quan đến thời gian & ngày,tháng,năm ,

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, nói nôm na helper date cung cấp cho chúng ta đầy đủ các hàm để xử lý các vấn đề về thời gian.Chúng ta sẽ đi qua từng hàm để biết cách sử dụng chúng.

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');
```

Sau khi hoàn tất quy trình này, chúng ta hoàn toàn có thể sử dụng toàn bộ hàm của helper date một cách thoải mái.

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 ở chỗ là mỗi kiểu hiển thị date thì phải bắt đầu bằng prefix `%`. Cái này rất hữu ích nếu chúng ta muốn hiển thị thông tin theo ngày, tháng, năm.

Ví dụ:

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

Mấy tham số trên nếu các bạn đã biết php thuần thì tôi cũng không cần phải giải thích làm gì, vì nó thuộc về basic rồi nhé.

days_in_month()

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

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

Kết quả sẽ trả về con số 31, không tin thì mở lịch ra xem nhé :D

timezone_menu()

Hàm này giúp chúng ta có được list 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) Enitwetok, 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, Ascention 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) Adu 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, Colomba, 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

standard_date()

Hàm này giúp chúng ta tạo ra các mốc thơi gian theo tiêu chuẩn nào đó với cú pháp sau.

```
$format  =  'DATE_RFC822';
$time  =  time();
echo standard_date($format, $time);
```

Tham số đầu tiên chứa kiểu chuẩn thời gian, tham số thứ 2 chứa thời gian theo chuẩn unix.

Các định dạng hỗ trợ:

- DATE_ATOM : 2005-08-15T16:13:03+0000
- DATE_COOKIE : Sun, 14 Aug 2005 16:13:03 UTC
- DATE_ISO8601 : 2005-08-14T16:13:03+00:00
- DATE_RFC822 : 14 Aug 05 16:13:03 UTC
- DATE_RFC850 : Sunday, 14-Aug-05 16:13:03 UTC
- DATE_RFC1036 : Sunday, 14-Aug-05 16:13:03 UTC
- DATE_RFC1123 : Sun, 14 Aug 2005 16:13:03 UTC
- DATE_RFC2822 : Sun, 14 Aug 2005 16:13:03 +0000
- DATE_RSS : Sun, 14 Aug 2005 16:13:03 UTC
- DATE_W3C : 2005-08-14T16:13:03+0000

Kết thúc bài học:

Với helper date chúng ta được cung cấp rất nhiều hàm dùng để xử lý vấn đề thời gian, và tôi chỉ liệt kê một số hàm thông dụng hay dùng mà thôi, nếu các bạn muốn hiểu sâu hơn thì mở user guide của nó và xem thêm, hoặc có thể google tìm theo bài viết hướng dẫn về nó. Ở loạt bài tiếp theo chúng ta sẽ tìm hiểu một số helper thông dụng khác.

Bài 14: Tìm Hiểu Helper Text

Trong bài này bạn sẽ được học:

1. Giới thiệu về helper text.
2. Các hàm thông dụng.

Lưu ý:

Tôi sử dụng Codeigniter version 2.1.4.

Tên folder của tôi là citest cho bài viết này.

Nhìn tiêu đề, hẳn các bạn đã đoán ra hôm nay chúng ta sẽ tìm hiểu về cái gì đúng không nào. Việc xử lý văn bản trong website là một khái niệm rất phổ

biến đối với lập trình viên, **helper text** do CI cung cấp cho chúng ta rất nhiều hàm để làm việc này, do bài viết chỉ cô đọng về lý thuyết sẽ hơi khô khan, nhưng cũng rất mong các bạn kiên trì đọc hết bài nhé. Như vậy ở bài trước chúng ta đã tìm hiểu xong helper date, và hôm nay chúng ta sẽ tiếp tục đi qua các hàm của text để xem cách sử dụng từng hàm ra sao.

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

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

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

Các hàm thông dụng trong helper text

word_limiter()

Hàm này xóa bỏ các chuỗi với số lượng từ được giữ lại , nó có phần đuôi thêm vào đoạn văn bản sau khi cắt chữ sẽ là ba chấm.

```
$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, tức là cắt chuỗi từ câu thứ 4.
```

character_limiter()

Hàm này dùng để xóa bỏ chuỗi với số lượng từ được quy định.Nó sẽ duy trì tính toàn vẹn của các từ (có nghĩa là từ có nghĩa hay là từ cách nhau bởi dấu cách)nên số ký tự có thể là nhiều hơn hoặc ít hơn một chút so với con số mà ta chỉ định.

```
$string  =  "Day la vi du ve character_limiter";
$string  =  character_limiter($string, 9);
echo $string;
// Ket  Qua: Day la vi, nó tính luôn khoảng cách nên ko thể cắt tối chỗ du đc.
```

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 chứa mảng các ký tự sẽ bị ẩn đi, tham số thứ ba, có thể định nghĩa từ thay thế, nếu không nó sẽ hiển thị mặc định là `####`, ở đây tôi 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.

1. Tham số thứ nhất là phần văn bản gốc.
2. Tham số thứ hai là đoạn văn bản muốn tô màu.
3. Tham số thứ ba & tư sẽ chứa phần đóng mở html định dạng màu sắc cho văn bản.

```
$string  =  "Freetuts.net la website chia se kinh nghiem lap trinh";
 echo  $string   =   highlight_phrase($string,  "Freetuts.net",  "<span  style  =  'color:red;'>",
"</span>");
//Ket qua : Freetuts.net sẽ được tô màu đỏ.
```

Kết thúc bài học:

Helper text chỉ đơn giản là nhiêu đó hàm thôi, còn vài hàm nhưng tôi nghĩ các bạn sẽ chẳng bao giờ sử dụng tới nó, nên tôi sẽ không giới thiệu cách sử dụng các hàm đó, để cắt một chuỗi trong đoạn văn bản thì chỉ nên dùng php thuần xử lý, bản thân framework đã rất chậm, nên tìm cách tối ưu nó, chứ không phải làm nó chậm hơn bằng việc load nhiều helper. Bài viết này mục đích tôi chỉ giới thiệu helper text cũng như cách sử dụng hàm thông dụng, không khuyến cáo các bạn sử dụng nó trong quá trình làm website.

Bài 15: Tìm Hiểu Helper Language

Trong bài này bạn sẽ được học:

1. Giới thiệu về helper language.
2. Các hàm thông dụng.

Lưu ý:

Tôi sử dụng Codeigniter version 2.1.4.Tên folder của tôi là citest cho bài viết này.

Hôm nay chúng ta sẽ cùng tìm hiểu một helper cũng khá là phổ biến trong quá trình xây dựng website , việc tùy chỉnh ngôn ngữ trên website khá là cần thiết, nếu các bạn đang làm việc với php thuần sẽ khá là đau đầu khi nghĩ đến việc xử lý vấn đề song ngữ. Nhưng với CI thì mọi chuyện trở nên rất là dễ dàng, bộ helper cung cấp cho chúng ta đầy đủ các hàm thông dụng.

Giới thiệu helper language:

Trong bài viết này , chúng ta chỉ tìm hiểu ở khái niệm cơ bản nhất về **helper language** giống như cách khai báo helper và sử dụng nó như thế nào chứ không đi sâu vào vấn đề thao tác với CSDL, vậy thì các bạn hiểu thế nào là website song ngữ tức là nó có nhiều hơn một ngôn ngữ, ngoài tiếng việt thì nó có thêm ngôn ngữ tiếng anh, tiếng pháp chẳng hạn. Chúng ta sẽ tìm hiểu helper này qua một ví dụ cụ thể như sau.

Cấu hình & sử dụng helper language:

Đầu tiên chúng ta sẽ 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 xong folder trên, chúng ta sẽ tạo tiếp file tên là *vi_lang.php* dùng để chứa các thông số ngôn ngữ , chú ý là phải có tiền tố *lang* ở đằng sau để CI có thể hiểu được nhé.
Tạo xong đầy đủ folder & file trên, 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";
```

Giải thích cho đoạn code trên , ta chỉ cần 2 tham số *$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 tôi sẽ tạo một controller tên là *demolang* , ngay tại action *index*tôi 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')." : Hasegawa kaito<br />";
echo $this->lang->line('email')." : hoaiminhit1990@gmail.com<br />";
echo $this->lang->line('address')." : freetuts.net<br />";
echo $this->lang->line('phone')." : 0934567890";
?>
```

```
Họ tên : Hasegawa kaito
Hòm thư : hoaiminhit1990@gmail.com
Địa chỉ : freetuts.net
Điện thoại : 0934567890
```

Ta gọi trực tiếp key khai báo ở file *vi_lang* theo cú pháp **$this->lang->line('tham_so')**.

Kết thúc bài học:

Các bạn có thể kết hợp sử dụng nó với các thao tác xử lý form cũng được, và bài viết sẽ dừng lại ở mức độ basic và nếu có thời gian chúng ta sẽ tiếp tục đi sâu vào helper này hơn. xử lý với model, load js để hiển thị ngôn ngữ theo ý đồ của bạn.

Bài 16: Kỹ thuật master layout

Đây là một vấn đề mở rộng mà CI không đề cập trong user guide, đó là khái niệm về **master layout**, vậy tại sao chúng ta lại phải tìm hiểu và sử dụng nó. 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..vvv và trong từng action như vậy, chúng ta phải đổ nội dung của layout ra. 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, chẳng lẽ cứ mỗi action là phải copy lại đoạn html để sử dụng lại, điều này không hợp lý tí nào cả. Vì vậy đó là lý do chúng ta phải có một giải pháp nào đó cho việc xử lý vấn đề layout trong **codeigniter framework**.

Cấu hình master layout

Bài viết này có liên quan đến view, vì thế để có thể tiếp thu bài này tốt hơn thì các bạn cần phải có kiến thức tương đương về **view**, Đầu tiên chúng ta phải tạo ra một cái 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, nói chung chung mà không demo thì các bạn sẽ không thể nào hình dung được vấn đề, tôi sẽ ví dụ thông qua 2 controller là `admin & home`, hiển thị 2 layout hoàn toàn khác nhau.

Thực hành master layout -xây dựng layout admin:

Như tôi đã trình bày ở phía trên, chúng ta cần phải có một khung layout cố định , tức là một file chứa phần `header & footer, tôi 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 đấy các bạn, kèm theo đó tôi sè tạo thêm 2 file `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 tôi tiến hành tạo ra layout basic 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, vấn đề đặt ra ở đây là làm sao chúng ta có thể load view chồng view được?

```
<!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">2014 © by freetuts.net</div>
```

```
</body>
</html>
```

Chuyện đó nói sau, giờ chúng ta phải tạo ra controller admin và action index và tôi load cái view ra luôn, vậy tôi có cú pháp sau, nội dung trong view là đoạn text được bao bọc trong thẻ h1.

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`, nó sẽ phải xuất hiện ở ngay vị trí content trong file `main.php`, để có thể làm được điều đó thì chúng ta phải làm sao truyền một cái view sang file main. Để có thể **load view lồng view**, thì chúng ta phải tạo ra một trường nào đó để chúng ta truyền tham số sang file main. Và ngay tại đó các bạn phải load tiếp một cái view nửa. Chúng ta sẽ tạo ra một `$data['subview'] = 'View của action'` sau đó truyền tham số vào file main bằng cú pháp sau.

```
<?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">2014 © by freetuts.net</div>
</body>
</html>
```

Sau đó mở file `main.php` ra, ngay tại vị trí content tôi `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. tôi style css cho layout dễ nhìn hơn. Để test chúng ta ra trình duyệt và gõ. `localhost/citest/index.php/admin` nếu kết quả trả về như hình, xem như bước đầu thành công rồi nhé.

Ấy, nhưng mà chúng ta không muốn show chữ `admin/index_view` ra như thế này, mà chúng ta muốn show nội dung của index_view cơ mà, để làm được điều này thì không có gì là khó. Ngay tại vị trí mà chúng ta echo, chúng ta tiến hành load ra một cái view với tham số là index_view, vậy chúng ta có cú pháp như sau.

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

Nếu kết quả trả về như hình, xem như chúng ta thành công trong việc xây dựng master layout cơ bản nhất, xem như chúng ta đa giải quyết được bài toán view lồng view một cách đơn giản.

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, F5 lại trình duyệt xem **title page** có đúng như sự kỳ vọng của chúng ta hay không.

```
<?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 giải thích cho vấn đề này, tôi sẽ ví dụ thông qua controller category và action là add, và tôi cũng muốn thông qua ví dụ này để có thể thể hiện việc show dữ liệu thông qua master layout. Tôi có `$info` chứa thông tin tôi cần hiển thị ở `addcate_view`, đầu tiên tôi tiến hành test bằng cách in dữ liệu trong cặp thẻ pre, sau đó tôi dùng vòng lặp foreach xuất dữ liệu thông tin cái mảng vừa khai báo. $k là khóa, $v là giá trị của khóa, cái này quá basic rồi hen.

```
<?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 />";
}
?>
```

Kết Quả:

Vậy xem như chúng ta vừa hoàn thanh xuất sắc trong việc xử lý vấn đề layout trong CI, để các bạn hiểu rỏ hơn thì tôi có thêm một ví dụ nhỏ, xây dựng layout cho trang home, không có gì ghê gớm đâu, chỉ là thay màu header & nội dung trong view là xong.

Thực hành master layout - xây dựng layout home:

Tôi tạo controller home & action index, copy các thông số từ controller admin, thay đổi tên view là xong ngay thôi.

```
<?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);
    }
}
```

```
<!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>
```

Ra trình duyệt và gõ, *localhost/citest/index.php/home*, nếu kết quả trả về như hình xem như các bạn đã thành công, trong việc xây dựng layout home.

Kết thúc bài học:

Thật ra có rất nhiều library ra đời hỗ trợ chúng ta trong việc xử lý layout đối với Codeigniter framework, nhưng để control được hoàn toàn các libs đó, các bạn sẽ phải tốn khá nhiều thời gian, đọc document của nó, chưa kể đâu phải bạn nào cũng giỏi tiếng anh đâu, thật ra thì nó cũng nôm na là view lồng view thôi, kỹ thuật master layout này do tôi tìm hiểu từ một tutorial của nước ngoài, cảm thấy hay nên tôi chia sẽ với các bạn, hy vọng các bạn sẽ có riêng cho mình những giải pháp trong việc xử lý layout. Bài viết này khá là quan trọng, vì

trong các bài viết sau, đều có liên quan đến nó. Mong các bạn dành time ra xem bài cho kỹ.

Bài 17: Xây dựng crud add - update - edit user

Crud là một thuật ngữ không hề xa lạ với dân lập trình, nó là khái niệm viết tắt của `"Create, Read, Update, Delete"`, nói tới đây chắc hẳn các bạn đã biết, hôm nay chúng ta sẽ cùng nhau tìm hiểu về cái gì rồi đúng không. Để có thể tiếp thu bài này tốt hơn thì các bạn cần phải có một số kiến thức như sau, biết cách thao tác với form validation, load model, active record, session và biết cách cấu hình master layout.

Để có thể hiểu rõ hơn và thực tế hơn , chúng ta sẽ cùng nhau tìm hiểu cách viết chức năng `liệt kê danh sách thành viên, thêm, xóa sửa thành viên`. Tất cả chỉ gói gọn với kiến thức CI căn bản chứ không có gì khó khăn cả, ráp kiến thức lại và chúng ta viết hoàn chỉnh một chức năng trong ứng dụng website mà thôi.

Ok, giới thiệu qua loa thế là được rồi, bây giờ chúng ta phân tích xem cần chuẩn bị những gì để bắt tay vào viết chức năng **crud**, đầu tiên cần phải lên database cho chức năng **User**, do chỉ là ví dụ đơn giàn nên table user của tôi chỉ có 5 field thôi nhé.

Xây dựng database cho crud user:

Tôi có chuẩn bị sẵn cấu trúc database 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;
```

Tôi cũng insert sẵn dữ liệu cho table user, import database là quá basic rồi, nên tôi sẽ không hướng dẫn lại nhé. Vào phpmyadmin, vào database bạn đã tạo sẵn, vào sql, paste 2 dòng code sql này vào.

```
insert  into `user`(`id`,`username`,`password`,`email`,`level`)
values
(1,'admin','123','kaito99999@yahoo.com',2),
(2,'kaito','123456','kaito9999@gmail.com',1),
(7,'dlink','132132','xcxzc',1),(8,'lynsick','123','lynsick@yahoo.com',1),
(9,'yongc','cuibap','yongc@yahoo.com',1),(10,'eric','21222','hgffh',1),
(18,'van cuong','123','vancuong@gmail.com',1),
(19,'lksport','123','lksport@yahoo.com',1),
(20,'indochina','123','indochina@gmail.com',2),
(22,'itnameserver','123','itname@yahoo.com',1);
```

Ok, sau khi hoàn tất việc lên database, thì chúng ta tiến hành khai báo cấu hình database trong CI như sau. Vào folder *application/config/database.php*.

```
$db['default']['hostname']  =  'localhost';
$db['default']['username']  =  'root';
$db['default']['password']  =  'vertrigo';
$db['default']['database']  =  'ciexam';
$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 webserver mà các bạn đang sử dung nhé,tôi thì dùng vertrigo vì cảm thấy nó thông dụng và dễ xài, ít gặp các lỗi như các webserver khác.

Hoàn thành công việc cấu hình database xong, tiếp theo chúng ta cho nó autoload luôn,đỡ phải mất công khai báo trong file model nửa. với chức năng chúng ta đang xây dựng, sẽ phải dùng tớisession, form validation, database, nên chúng ta tiến hành cho các library này autoload cho tiện, vậy thì autoload ra sao,

trong folder config, có file autoload.php mở file lên và tìm đến dòng 55 thêm vào như sau.

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

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

Cấu hình master layout:

Vào folder `application/views`, tạo thêm folder user và trong folder tạo thêm 4 file nửa, 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);
    }
}
```

Tôi khởi tạo phương thức `$_data` để có thể truyền biến thoải mái trong toàn bộ action và tôi sẽ truyền biến theo cú pháp `$this->_data`. Tiếp theo chúng ta vào file main.php xứ lý html và load **$subvie**w vào div content, và chúng ta sẽ phải dùng tới hàm **base_url()** nên chúng ta gọi helper url ngay constructor để có thể tái sử dụng lại trong toàn bộ 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">User</a></li>
                <li><a href = "#">Chuyên mục</a></li>
                <li><a href = "#">Bài viết</a></li>
                <li><a href = "#">Comment</a></li>
            </ul>
            <p>
                Xin chào <span>Hasegawa kaito</span>
                <a href = "#">[ Thoát ]</a>
            </p>
        </div><!-- End header -->
        <div id = "content">
            <!--=  =  =  =  =  Begin #content  = =  =  =  =  =  -->
            <?php echo $this->load->view($subview); ?>
        </div><!-- End #content -->
        <div id = "footer">
            copyright &copy; by Hasegawa kaito, power by freetuts.net&reg;
        </div><!-- End #footer -->
    </body>
</html>
```

Tôi 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;}
/* = = = = 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: #FFFFF;
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` tôi cho nội dung đơn giản để test xem cấu hình master layout thành công chưa, chạy link *localhost/citest/index.php/user*

Ok, xem như chúng ta cấu hình thành công master layout rồi đấy công việc tiếp theo làm sẽ là liệt kê list thành viên ra layout.

Viết chức năng list với crud user:

Nhắc tới liệt kê sanh sách thành viên thì chúng ta sẽ nghĩ ngay tới việc phải thao tác với `model & active record`,vậy trong folder models tôi 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);
    }
}
```

Do tôi không muốn liệt kê toàn bộ column trong table user nên tôi chỉ định các comlum được phép hiển thị là `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()` sẽ giúp chúng ta 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, sau đó test bằng cách tạo `$this->_data['info']` trỏ tới phương thức **getList** trong model, sau đó sang index_view in dữ liệu trong cặp thẻ pre.

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

```
Array (
    [0]  =>  Array (
            [id]  =>  1
            [username]  =>  admin
            [email]  =>  kaito99999@yahoo.com
            [level]  =>  2
        )
    …. (các giá trị khác của mảng)
)
```

Chạy link *localhost/citest/index.php/user*, nếu trình duyệt trả về như sau thì là thành công, tiếp theo chúng ta sẽ dùng vòng lặp foreach để dữ liệu vào trong một cái table cho hiển thị đẹp mắt hơn nhé.

```
<table cellpadding = "0" cellspacing = "0" border = "0" width = "100%" id = "list_cate" class = "list">
    <tr>
         <td colspan = "6" align = "center">
             <a href = "<?php echo base_url(); ?>index.php/user/add">Add User</a><br />
         </td>
     </tr>
    <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>
```

Chúng ta có 6 cột tất cả, khởi tạo`$stt = 0` ở phía ngoài vòng lặp sau đó cho nó tăng dần đều `$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);
   }
```

Sau khi hoàn tất xong đoạn code trên, các bạn quay lại trình duyệt và nhấn F5, kết quả trả về như hình xem như là hoàn thành việc liệt kê danh sách. Tiếp theo chúng ta sẽ thao tác với một chức năng khá là đơn giản với bất kỳ ứng dụng nào, đó là chức năng Delete User.

Viết chức năng delete user với crud:

Giống như câu nói dân gian, xây dựng một cái gì đó thì rất là khó khăn, còn phá đi nó thì rất là đơn giản. 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, nhưng lạm dụng uri quá cũng không tốt lắm, vì thật chất nó là phương thức **GET** trong php thuần nên nó không thật sự an toàn. vì thế chúng ta có thể đếm vị trí id bằng cách sau. ví dụ link chúng ta có cấu trúc như sau. localhost/citest/index.php/user/del/id...bỏ index và chúng ta đếm nhé, 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 tôi 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 cái khóa chúng ta đặt tên, còn tham số thứ 2 là câu thông báo lỗi muốn xuất ra.

Khởi tạo xong rồi , ngay tại action index chúng ta sẽ gọi nó ra thôi, gọi ra bằng cú pháp

`$this->_data['mess'] = $this->session->flashdata('flash_mess')`,

ngay 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` ngay tại index_view nhé, F5 trình duyệt click xóa đại 1 user nào đó, nếu kết quả như hình xem như các bạn thành công.

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

Viết chức năng add user với crud:

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(); ?>index.php/user/add" 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">Username Infomations</legend>
        <label>Username:</label><input type = "text" name = "username" size = "28" class = "input"/>
```

```
        <label>Email:</label><input type = "text" name = "email" size = "28" 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" selected>Member</option>
            <option value = "2" >Administrator</option>
        </select></br />
        <label> </label><input type = "submit" name = "ok" value = "Add User" class = "btn" />
    </fieldset>
</form>
```

Hàm `validation_errors()` dùng để xuất ra thông báo lỗi kiểm soát dữ liệu nhập vào, chạy link *localhost/citest/index.php/user/add*. kết quả như hình, xem như bước đầu ok

Sau khi tạo ra thành công một cái form , thì công việc tiếp theo thì 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 tập luật theo ý của các bạn. Tôi sẽ không giải thích lại toàn bộ các tập luật trên, ví nó thuộc về kiến thức cũ nhé, tôi chỉ nói về hai tập luật mới, đó là matches & callback, matches kiểm tra xem password trúng với repass hay không, giá trị truyền vào là tên form của re-pass. callback là hàm check trùng lặp dữ liệu trong database, tí nửa chúng ta sẽ nói nhiều về nó hơn.

Tôi thử kiểm tra, bằng cách không nhập gì vào textbox mà nhấn submit xem có kết quả gì không nha, kết quả trả về là.

```
The Username field is required.
The Password field is required.
The Email field is required.
```

Như vậy, tập luật required đã hoạt động tốt, tiếp theo chúng ta kiểm tra xem email có hợp lệ hay không, username có giới hạn ký tự không, password có giống nhau hay không.

```

The Username field must be at least 4 characters in length.
The Password field does not match the password2 field.
The Email field must contain a valid email address.
```

Ok , xem như các tập luật đều đang được vận hành khá tốt, chúng ta cần phải trải qua một bước quan trong trong form validation.Kiểm tra xem các **set_rules** ở phía 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, tức là thêm một thành viên đấy các bạn.

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 registed,
please try again!");
            return FALSE;
        } else {
            return TRUE;
        }
    }
```

Tương tự như username, thì chúng ta cũng cần check luôn email xem có trùng lặp hay không, do 2 phương thức có phần giống nhau nên tôi copy phần check _user sang **check _email** cho đỡ tốn time nhé.

```
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 registed,
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, `checkUsename & 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 mà thôi. nếu nó tồn tại thì xuất ra thông báo lỗi. Các bạn có thể test với demo ở cuối bài viết.

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 những thông tin mà chúng ta cần thêm vào csdl, chúng ta dùng phương thức `$this->input->post()` .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);
  }
```

Ok, hoàn tất việc thêm một thành viên vào csdl, tiếp theo chúng ta sẽ viết chức năng edit user.

Viết chức năng edit user với crud:

Để edit được, các bạn phải nhấn vào link edit và khi đó thông qua một cái id mà chúng ta muốn sửa nó sẽ thể hiện trên một cái đường dẫn, và nhiệm vụ của chúng ta là phải tiến hành thực thi 2 thao tác. thao tác đầu tiên chúng ta phải lấy ra những record đang có trong csdl ứng với cái id đó để chúng ta đổ vào cái form, và nhiệm vụ thứ 2 là 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.

Ở phần này nó kha là giống với các thao tác mà chúng ta đã làm ở phần add user, cho nên phần này tôi sẽ không giải thích nhiều chỉ tập trung vào sự khác biệt giữa add & edit mà thôi.

Tôi tạo action edit và tôi truyền $id vào, để có thể edit bất kì record nào thì chúng ta phải lấy ra được cái id của record đó, tôi 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 tiên hành lấy dữ liệu từ trong form ra bằng cú pháp

`$this->_data['info'] = $this->Muser->getUserById($id)`

sau đó sang edit_view copy phần form bên add_view vào đổi lại dường dẫn trong form như sau.

```
<?php echo base_url(); ?>index.php/user/edit/<?php echo $info['id'];
```

```
<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"/>
        <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> </label>
        <input type = "submit" name = "ok" value = "Edit User" class = "btn" />
    </fieldset>
</form>
```

Cái chỗ value là basic nên tôi không nhắc lại nha, tiếp theo quay lại file muser để viết phương thức **updateUser** như sau.

```
public function updateUser($data_update, $id){
        $this->db->where("id", $id);
        $this->db->update($this->_table, $data_update);
}
public function edit($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);
    }
```

Y chang action add thôi, 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");
}
```

Vá 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 nửa. Tuy nhiên chúng ta hoàn toàn có thể sử dụng một lúc cả 2 phương thức cho add & edit luôn. 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 lúc này chúng ta sẽ có `$this->db->where("id != ", $id)`, tương tự ở email cũng y chang thế.

```
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 ngay tại vị trí của check_user chúng ta phải 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) {
            $this->form_validation->set_message("check_user", "Your username has been registed,
please try again!");
            return FALSE;
        } else {
            return TRUE;
        }
}
public function check_email($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  registed,
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);
}
```

Kết thúc bài học:

Hy vọng qua bài viết này các bạn sẽ hình dung được cách xây dựng chức năng **CRUD**, từ đó đúc kết cho mình những kinh nghiệm thực tế hơn trong quá trình tìm hiểu và chinh phục codeigniter framework.

Bài 18: Tìm Hiểu Library Shopping Cart

Chào mừng các bạn đã quay trở lại freetuts.net. Như vậy ở bài trước chúng ta đã tìm hiểu xong cách viết ứng dụng **Crud add, edit, delete**, tuy bài viết chỉ dừng lại ở mức cơ bản nhưng cũng đủ cho các bạn có cái nhìn tổng quan hơn về codeigniter framework. Hôm nay chúng ta sẽ đi đến một library khá là 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.

Cấu hình library shopping cart:

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

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

Để tiện trong quá trình viết tuts tôi sẽ load thư viện shopping cart ngay tại hàm khởi tạo của controller.

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

Vậy là cấu hình thư viện xong, giờ ta tìm hiểu cách sử dụng nó nhé.

Thêm sản phẩm:

Shopping Cart trong Codeigniter lưu dưới dạng mảng và có những key bắt buộc không được sửa như như id, qty, price, name. Ngoài ra nó còn có thêm key `options` chứa các thông tin con như size, color của sản phẩm đó 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(){
        $data = array(
            "id"  =>  "1",
            "name"  =>  "Viet Nam Khong So Trung Quoc",
            "qty"  =>  "1",
            "price"  =>  "100000",
            "option"  =>  array("author"  =>  "freetuts.net"),
        );
        // Them san pham vao gio hang
        if($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. Các bạn hãy chạy code và sẽ thấy kết quả thành công hay thất bại.

Show danh sách sản phẩm:

Trong đoạn code show danh sách sản phẩm.Tôi 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>";
}
```

Nếu kết quả như hình xem như chúng ta bước đầu tiếp cận được thao tác insert sản phẩm.

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 ra, 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'  =>  'Viet Nam Khong So Trung Quoc',
                'price'  =>  '10000',
                'qty'    =>  '1',
                'options'  =>  array('author'  =>   'freetuts.net')
            ),
            array(
                'id'  =>  '2',
                'name'  =>  'Trung Quoc Vi Pham Chu Quyen Viet Nam',
                'price'  =>  '20000',
                'qty'    =>  '1',
                'options'  =>  array('author'  =>   'freetuts.net')
            ),
            array(
                'id'  =>  '3',
                'name'  =>  'Tau Trung Quoc Lien Tuc Dam vao Tau Viet Nam',
                'price'  =>  '30000',
                'qty'    =>  '1',
                'options'  =>  array('author'  =>   'freetuts.net')
            ),

        );
        // 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] => Viet Nam Khong So Trung Quoc
            [qty] => 1
            [price] => 100000
            [option] => Array
                (
                    [author] => freetuts.net
                )
            [subtotal] => 100000
    )
          ... (các phần tử khác trong mảng)
)
```

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 foreach 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;
                $delOne = array("rowid" => $item['rowid'], "qty" => $item['qty']);
            }
        }
        if($this->cart->update($delOne)){
            echo "Xoa san pham thanh cong";
        }else{
            echo "Xoa san pham that bai";
        }
```

```
}
```

Xóa hết sản phẩm:

Hàm xóa hết sản phẩm của Shopping Cart trong codeigniter giống với hàm hủy toàn bộ session. Tôi dùng hàm `destroy`để xóa hết toàn bộ sản phẩm tồn tại trong giỏ hàng.

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

Cập nhật sản phẩm:

Giống như thao tác xóa một sản phẩm thì để có thể cập nhật một sản phẩm thì chúng ta cần phải có được rowid của sản phẩm đó. rowid giúp chúng ta xác định 9 xác 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";
        }
}
```

Các hàm khác:

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

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

Hàm đêm tống 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 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 đươc.

```
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 />";
                }
            }
        }
}
```

Kết thúc bài học:

Thông qua bài viết này hy vọng các bạn có thể phần nào nắm được các thao tác cơ bản trong việc sử dụng library shopping cart thao tác thêm, cập nhật sản phẩm.Tuy nó có rất nhiều nhược điểm nhưng nếu các bạn chỉ dùng để xây dựng các website có quy mô nhỏ & vừa thì nó hoàn toàn đáp ứng được hết nhé .Ở bài tiếp theo chúng ta sẽ tìm hiểu thao tác csdl trong shopping cart nó ra sao. Chào thân ái và quyết thắng.

Bài 19: Rewrite URL trong Codeigniter

Version: Codeigniter 3x

Sau một khoảng thời gian không đụng tới Codeigniter thì hôm nay lại có dịp tiếp xúc với Framework này. Chuyện là mình đang làm một dự án sử dụng Codeigniter nên hôm nay hứng thú làm một tuts và sẽ viết tiếp serie của tác giả Kaito. Trong bài này mình se hướng dẫn các bạn làm thế nào để **viết lại đường dẫn trong Codeigniter** (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 họ luôn ấy.

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

Ok ta bắt đầu nhé.

1. Tìm hiểu Route trong Codeigniter

Nếu bạn đang dùng Codeigniter 3x hay 2x thì đừng quá lo lắng nhé, vì nó cũng tương tự như nhau, chỉ khác ở một điểm là Route trong CI3X sẽ có chức năng callback.

Ok. Mọi thứ liên quan đến **Rewrite URL trong Codeigniter** 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;<br><br>
```

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: `$route['default_controller'] = 'product/index';`
- **404_override**: Nếu bạn không khai báo thì nó sẽ lấy trang mặc định lỗi của CI, còn nếu ban khai báo thì nó sẽ gọi đến controller và action trong đó. Ví dụ: `$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...';
```

Nguyên tắc capturing group trong regular expression

Để xử lý tốt **route trong CI** thì bạn phải biết khái niệm capturing group (xem bài các ký hiệu regex căn bản). Nhưng tôi cũng giải thích sơ qua cho bạn hiểu nhé.

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** nhé các bạn.

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['freetuts-blog'] = 'blog/user';
```

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

```
$route['freetuts-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 `freetuts-blog/`.

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

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

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

Sử dụng Regular Expression trong route

Bạn có thể sử dụng Regular Expression để sử dụng 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)`

Mình không liệt kê hết được, nếu bạn muốn thì hãy học regular expression nhé.

Route Callback trong Codeigniter

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

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;
};
```

Có lẽ tới đây thôi vì lý thuyết hơi dài dòng, ta đi vào làm thực tế nhé.

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 tôi có 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 chính nha các bạn 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

Ok ta sẽ viết hai routes cho đơn giản nhé:

```
$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 bất ngờ.

Rewrite trang category

Trang danh sách sản phẩm mình muốn nhận slug của category để truy vấn database nên nó sẽ có dạng sau:

- domain.com/slug-chuyen-muc
- domain.com/slug-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/slug-cua-tag
- domain.com/tag/slug-cua-tag/page/1

Và đây là routes:

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

Bạn chạy URL trên có thể sẽ thấy không đúng thì tức là tại bạn đặt sai vị trí cho route này, bạn phải đặt nó trên route của cate nhé, bởi vì trong route của cate có `$route['(:any)'] = 'product/category/$1';` nên nó sẽ đúng với bất kì trường hợp nào, chính vì vậy nó sẽ ngưng và ko duyệt xuống dưới nên giải pháp là đặt tag 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 là slug của tin và id của tin, như vậy UR sẽ có dạng:

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

Ok 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 đặ 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 được URL rồi đấy :D.

3. Lời kết

Bài này cũng tương đối đơn giản vì nó mang tính chất tham khảo, nếu làm không được bạn hãy comment để mình có thể giải đáp thắc mắc cho bạn, hoặc bạn có thể đặt câu hỏi trong phần hỏi đáp. Chúc bạn học tốt