ỦY BAN NHÂN DÂN TỈNH LÂM ĐỒNG

TRƯỜNG CAO ĐẲNG NGHỀ ĐÀ LẠT

GIÁO TRÌNH

MÔN HỌC/ MÔ ĐUN: XÂY DỰNG WEBSITE SỬ DỤNG FRAMEWORK (PHP CODEIGNITER FRAMEWORK)

NGÀNH/ NGHỀ: THIẾT KẾ TRANG WEB

TRÌNH ĐỘ: CAO ĐẲNG

Ban hành kèm theo Quyết định số: 1157/QĐ-CĐNĐL ngày 11 tháng 12 năm 2019 của Hiệu trưởng Trường Cao đẳng Nghề Đà Lạt

(LƯU HÀNH NỘI BỘ)

Lâm Đồng, năm 2019

TUYÊN BỐ BẢN QUYỀN

Giáo trình được lưu hành nội bộ Trường Cao đẳng Nghề Đà Lạt.

LỜI GIỚI THIỆU

Vài nét về xuất xứ giáo trình:

Giáo trình này được viết theo căn cứ Thông tư số 03/2017/TT-BLĐTBXH ngày 01 tháng 03 năm 2017 của Bộ Lao động – Thương binh và Xã hội về việc Quy định về quy trình xây dựng, thẩm định và ban hành chương trình; tổ chức biên soạn, lựa chọn, thẩm định giáo trình đào tạo trình độ trung cấp, trình độ cao đẳng.

Quá trình biên soạn:

Giáo trình này được biên soạn có sự tham gia tích cực của các giáo viên có kinh nghiệm, cùng với những ý kiến đóng góp quý báu của các chuyên gia về lĩnh vực công nghệ thông tin.

Mối quan hệ của tài liệu với chương trình, mô đun/môn học:

Căn cứ vào chương trình đào tạo nghề Thiết kế trang web, giáo trình giúp cung cấp cho người học những kỹ năng sử dụng nền tảng CodeIgniter để phát triển ứng dụng web bằng ngôn ngữ PHP.

Để học được mô đun này người học cần có kiến thức cơ bản về PHP và HTML.

Cấu trúc chung của giáo trình này bao gồm 5 bài:

CodeIgniter là một nền tảng ứng dụng web nguồn mở được viết bằng ngôn ngữ PHP bởi Rick Ellis (CEO của EllisLab, Inc). Phiên bản đầu tiên được phát hành ngày 28.02.2006, phiên bản hiện tại: 1.7.2 (phát hành ngày 11.09.2009). Ý tưởng xây dựng CodeIgniter được dựa trên Ruby on Rails, một nền tảng ứng dụng web được viết bằng ngôn ngữ Ruby. Hiện tại, CodeIgniter đang được phát triển bởi ExpressionEngine Development Team thuộc EllisLab, Inc.

Lời cảm ơn

Giáo trình được biên soạn trên cơ sở các văn bản quy định của Nhà nước và tham khảo nhiều tài liệu liên quan có giá trị. Song chắc hẳn quá trình biên soạn không tránh khỏi những thiếu sót nhất định. Ban biên soạn mong muốn và thực sự cảm ơn những ý kiến nhận xét, đánh giá của các chuyên gia, các thầy cô đóng góp cho việc chỉnh sửa để giáo trình ngày một hoàn thiện hơn.

Lâm Đồng, ngày 10 tháng 12 năm 2019

Tham gia biên soạn

1. Phạm Đình Nam
2. Ngô Thiên Hoàng
3. Nguyễn Quỳnh Nguyên
4. Phan Ngọc Bảo

Mục lục

1. Giới thiệu

1.1. Tổng quan

CodeIgniter là một nền tảng ứng dụng web nguồn mở được viết bằng ngôn ngữ PHP bởi Rick Ellis (CEO của EllisLab, Inc). Phiên bản đầu tiên được phát hành ngày 28.02.2006, phiên bản hiện tại: 1.7.2 (phát hành ngày 11.09.2009). Ý tưởng xây dựng CodeIgniter được dựa trên Ruby on Rails, một nền tảng ứng dụng web được viết bằng ngôn ngữ Ruby. Hiện tại, CodeIgniter đang được phát triển bởi ExpressionEngine Development Team thuộc EllisLab, Inc.

1.2. Những điểm nổi bật

Được thiết kế theo mô hình Model-View-Controller: Mô hình MVC giúp tách thành phần hiển thị giao diện (presentation) và xử lý (business logic) của một phần mềm thành những thành phần độc lập, từ đó giúp cho việc thiết kế, xử lý và bảo trì mã nguồn dễ d{ng, đồng thời tăng khả năng mở rộng của phần mềm. CodeIgniter vận dụng mô hình này trong thiết kế, giúp tách biệt các tập tin giao diện với các tập tin xử lý dữ liệu, nâng cao khả năng quản lý và dễ bảo trì.

Nhỏ gọn: Gói c{i đặt chỉ 404KB (không bao gồm phần User Guide). So với các PHP framework kh|c như CakePHP (1.3MB), Symfony (5.08MB) hay Zend Framework (5.66MB)…kích thước của CodeIgniter giúp giảm thiểu đ|ng kể không gian lưu trữ.

Tốc độ nhanh: CodeIgniter được đ|nh gi| l{ PHP framework có tốc độ nhanh nhất hiện nay. Bằng cơ chế lưu nội dung vào bộ đệm (cache), kiểm tra bộ đệm trước khi tiến hành thực hiện yêu cầu, CodeIgniter giảm số lần truy cập và xử lý dữ liệu, từ đó tối ưu hóa tốc độ tải trang.

Miễn phí: CodeIgniter được ph|t h{nh dưới giấy phép Apache/BSD mở rộng, cho phép người dùng tự do thay đổi, phát triển và phân phối mã nguồn.

Hỗ trợ Search Engine Optimization: Cấu trúc URL của CodeIgniter rất thân thiện với các robot tìm kiếm.

Hệ thống thư viện phong phú: CodeIgniter cung cấp c|c thư viện phục vụ cho những tác vụ thường gặp nhất trong lập trình web, chẳng hạn như truy cập cơ sở dữ liệu, gửi email, kiểm tra dữ liệu, quản lý session, xử lý ảnh…đến những chức năng n}ng cao như XML-RPC, mã hóa, bảo mật…

Bảo mật hệ thống: Cơ chế kiểm tra dữ liệu chặt chẽ, ngăn ngừa XSS và SQL Injection của CodeIgniter giúp giảm thiểu các nguy cơ bảo mật cho hệ thống.

1.3. Những điểm hạn chế

Chưa hỗ trợ Object-Relational Mapping: Object Relational Mapping (ORM) là một kỹ thuật lập trình, trong đó các bảng của cơ sở dữ liệu được ánh xạ th{nh c|c đối tượng trong chương trình. Kỹ thuật này giúp cho việc thực hiện c|c thao t|c trong cơ sở dữ liệu (Create Read Update Delate – CRUD) dễ dàng, mã nguồn ngắn gọn hơn. Hiện tại, CodeIgniter vẫn chưa hỗ trợ ORM.

Chưa hỗ trợ AJAX: AJAX (Asynchronous JavaScript and XML) đ~ trở thành một phần không thể thiếu trong bất kỳ ứng dụng Web 2.0 nào. AJAX giúp n}ng cao tính tương t|c giữa người dùng và hệ thống, giúp cho người dùng có cảm gi|c như đang sử dụng ứng dụng desktop vì c|c thao t|c đều diễn ra "tức thời". Hiện tại, CodeIgniter vẫn chưa có thư viện dựng sẵn n{o để hỗ trợ xây dựng ứng dụng AJAX. Lập trình viên phải sử dụng c|c thư viện bên ngo{i, như jQuery, Script.aculo.us, Prototype hay Mootools…

Chưa hỗ trợ một số module thông dụng: So sánh với framework khác, CodeIgniter không có các module thực thi một số tác vụ thường gặp trong quá trình xây dựng ứng dụng web như Chứng thực người dùng (User Authorization), Trình phân tích RSS (RSS Parser) hay Trình xử lý PDF…

Chưa hỗ trợ Event-Driven Programming: Event-Driven Programming (EDP) là một nguyên lý lập trình, trong đó các luồng xử lý của hệ thống sẽ dựa vào các sự kiện, chẳng hạn như click chuột, gõ b{n phím…Đ}y không phải là một khuyết điểm to lớn của CodeIgniter vì hiện tại, chỉ có một số ít framework hỗ trợ EDP, bao gồm Prado, QPHP và Yii.

1.4. Cài đặt

Yêu cầu hệ thống: CodeIgniter có thể hoạt động trên nhiều hệ điều hành và server, yêu cầu có c{i đặt PHP phiên bản 4.x hoặc cao hơn; hệ quản trị cơ sở dữ liệu: MySQL (4.1+), MySQLi, Mircrosoft SQL Server, Postgres, Oracle, SQLite, và ODBC.

Hướng dẫn c{i đặt:

1. Download bộ nguồn CodeIgniter tại website http://codeigniter.com/, phiên bản hiện tại: 1.7.2.
2. Mở tập tin `application/config/config.php` bằng một chương trình soạn thảo, thay đổi giá trị `$config['base_url']`. Đ}y l{ đường dẫn tuyệt đối đến thư mục CodeIgniter trên server.
3. Nếu ứng dụng có tương t|c với cơ sở dữ liệu, thiết lập các giá trị cần thiết trong tập tin `application/config/database.php`.

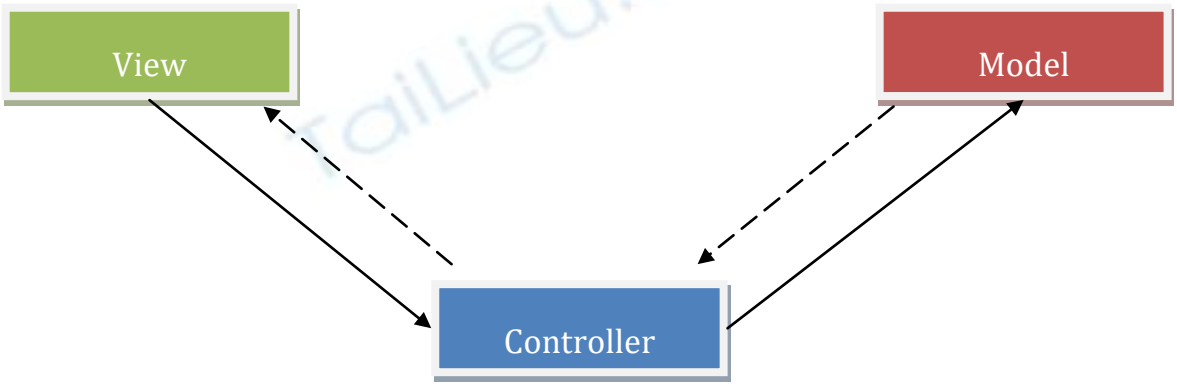
4. Upload tất cả thư mục và tập tin của CodeIgniter lên server.

Để nâng cao tính bảo mật hệ thống, người dùng có thể đổi tên thư mục system của CodeIgniter. Sau khi đổi tên, người dùng phải thay đổi giá trị biến `$system_folder` trong tập tin `index.php`.

2. Tìm hiểu

2.1. Mô hình Model-View-Controller

Model-View-Control (MVC) là một kiến trúc phần mềm, hiện đang được xem là một mẫu thiết kế trong công nghệ phần mềm. Mô hình MVC tách biệt phần xử lý dữ liệu ra khỏi phần giao diện, cho phép phát triển, kiểm tra và bảo trì các thành phần một cách độc lập.



Hình 1: *Minh họa mô hình MVC*

Theo đó:

- **Model** thể hiện các cấu trúc dữ liệu. Các lớp thuộc thành phần Model thường thực hiện các tác vụ như truy vấn, thêm, xóa, cập nhật dữ liệu. Khi dữ liệu trong Model thay đổi, thành phần View sẽ được cập nhật lại.
- **View** là thành phần thể hiện dữ liệu trong Model thành các giao diện tương t|c với người sử dụng. Một Model có thể có nhiều View tùy thuộc vào các mục đích khác nhau.
- **Controller** đóng vai trò trung gian giữa Model v{ View. Thông tin người dùng từ View được gửi cho Controller xử lý, sau đó Controller tương t|c với Model để lấy dữ liệu được yêu cầu, sau cùng Controller trả dữ liệu này về cho View.

Mô hình MVC thường được sử dụng trong các ứng dụng web, vì thành phần View (mã HTML/XHTML) được sinh ra từ các ngôn ngữ lập trình web. Thành

phần Controller sẽ nhận các dữ liệu GET/POST, xử lý những dữ liệu n{y, sau đó chuyển sang Model xử lý.

Model sẽ trả dữ liệu về phía Controller, sau đó Controller sinh m~ HTML/XHTML để thể hiện trên View.

2.2. Cấu trúc CodeIgniter

Hình 2: *Cấu trúc CodeIgniter*

Tập tin `index.php` được xem như controller đầu vào, tiếp nhận các yêu cầu từ phía client và chuyển các yêu cầu này cho hệ thống xử lý.

Thư mục `system` bao gồm phần lõi của CodeIgniter. Chúng bao gồm c|c thư viện xây dựng sẵn, các tập tin ngôn ngữ, ghi chú về hệ thống. Trong số đó, c|c thư mục sau khá quan trọng:

- Thư mục `application`: Dành cho lập trình viên, các tập tin được lập trình cho ứng dụng sẽ lưu trong thư mục này.
- Thư mục `cache`: Bộ đệm của hệ thống, chứa c|c trang đ~ được xử lý trước đó.
- Thư mục `helpers`: Chứa các hàm hỗ trợ cho lập trình viên khi viết ứng dụng.

  Thư mục `libraries`: Chứa c|c thư viện dựng sẵn của CodeIgniter.

Đối với lập trình viên, các tập tin của ứng dụng sẽ được lưu trong thư mục `system/application`. Trong đó:

- Thư mục `config`: Chứa các tập tin cấu hình hệ thống
- Thư mục `controllers`: chứa các lớp controller
- Thư mục `errors`: chứa các tập tin lỗi
- Thư mục `helpers`: chứa các hàm tiện ích do người dùng định nghĩa
- Thư mục `hooks`: chứa các tập tin để mở rộng mã nguồn CodeIgniter
- Thư mục `language`: chứa các tập tin ngôn ngữ
- Thư mục `libraries`: chứa c|c thư viện cho người dùng dùng định nghĩa
- Thư mục `models`: chứa các lớp model
- Thư mục `views`: chứa các lớp view

Ta cũng có thể đổi tên của thư mục `application` th{nh tên tùy ý. Sau khi đổi tên, cần thiết lập tên mới cho biến `$application_folder` trong tập tin `index.php`. Ngoài ra, ta cũng có thể tạo nhiều ứng dụng trong cùng một bộ c{i đặt CodeIgniter bằng cách tạo ra c|c thư mục con bên trong `system/application`. C|c thư mục này có cùng cấu trúc giống như thư mục `application` gốc.

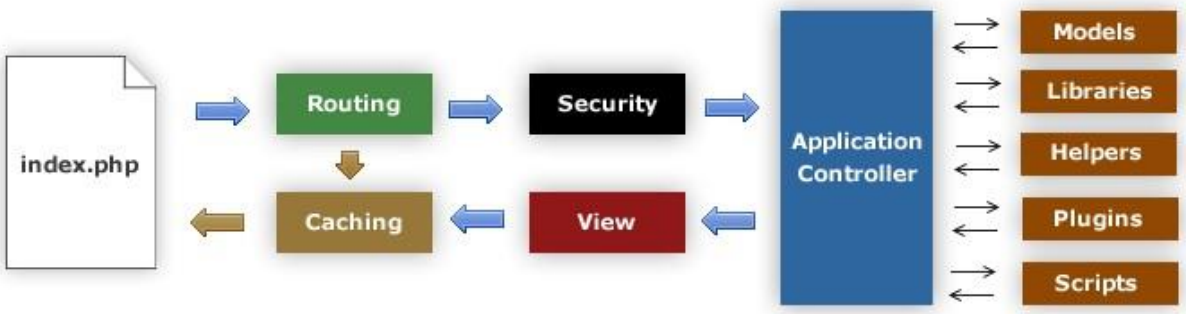
Chẳng hạn, ta có hai ứng dụng, `foo` và `bar`. Khi đó cấu trúc thư mục sẽ như sau:

```
system/application/foo/
system/application/foo/config/
system/application/foo/controllers/
system/application/foo/errors/
system/application/foo/libraries/
system/application/foo/models/
system/application/foo/views/
system/application/bar/
system/application/bar/config/
system/application/bar/controllers/
system/application/bar/errors/
system/application/bar/libraries/
system/application/bar/models/
system/application/bar/views/
```

Để biết tập tin index.php sẽ chạy ứng dụng nào, ta khai báo lại giá trị của biến `$application_folder`.

```
$application_folder = "application/foo";
```

2.3. Dòng chảy dữ liệu trong CodeIgniter



Hình 3: *Sơ đồ thể hiện dòng chảy dữ liệu trong CodeIgniter*

1. Tập tin `index.php` đóng vai trò l{m controller đầu vào, thiết lập các tài nguyên cần thiết cho hệ thống.

2. *Routing:* Qu| trình điều hướng giúp x|c định các yêu cầu v{ hướng xử lý đối với chúng.

3. *Caching:* Nếu dữ liệu được yêu cầu đ~ được lưu trong bộ đệm, CodeIgniter sẽ trả dữ liệu trong bộ đệm về phía client. Quá trình xử lý kết thúc.

4. *Security:* Dữ liệu trước khi được chuyển đến các Controller sẽ được lọc để phòng chống XXS hoặc SQL Injection.

5. *Application Controller:* Controller xử lý dữ liệu nhận được bằng cách gọi đến các Models, Libraries, Helpers, Plugins…có liên quan.

6. *View:* Dữ liệu được chuyển qua View để hiển thị cho người dùng. Nếu chức năng caching được bật, dữ liệu sẽ được lưu trong cache cho những lần yêu cầu tiếp theo.

Trong thực tế, đôi khi ta cần xen v{o c|c bước trong sơ đồ trên. Chẳng hạn, trước khi một controller được gọi, ta có thể muốn lưu log xuống tập tin trong hệ thống. CodeIgniter sử dụng cơ chế hook cho phép ta thực hiện việc n{y. Để kích hoạt hook, ta cần thay đổi giá trị của biến `$config['enable_hooks']` trong tập tin `application/config/config.php`:

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

C|c hook được khai báo trong tập tin application/config/hooks.php, có mẫu sau:

```
$hook['pre_controller'] = array(
                                'class'    => 'MyClass',
                                'function' => 'Myfunction',
                                'filename' => 'Myclass.php',
                                'filepath' => 'hooks',
                                'params'   => array('beer', 'wine',
'snacks')                                    );
```

Trong đó:

- `class`: tên lớp sẽ được gọi. Nếu chỉ sử dụng một hàm thủ tục, ta có thể bỏ trống giá trị này.
- `function`: tên h{m/phương thức sẽ được gọi.
- `filename`: tên tập tin chứa lớp/h{m đó.
- `filepath`: đường dẫn đến tập tin hook. Nếu tập tin nằm trong thư mục `application/hooks`, ta chỉ cần đặt giá trị này bằng `hooks`. Nếu tập tin nằm trong thư mục `application/hooks/utilities`, ta đặt giá trị này bằng `hooks/utilities`. Lưu ý, không có dấu / ở cuối.
- `params`: các tham số được truyền vào cho hàm.

Khóa `pre_controller` cho biết thời điểm hook n{y được thực hiện. CodeIgniter hỗ trợ các thời điểm sau:

- `pre_system`: được gọi khi hệ thống vừa khởi động. Ở giai đoạn này chỉ có lớp benchmark v{ c|c hook được kích hoạt. Các tiến trình khác vẫn chưa xảy ra.
- `pre_controller`: được gọi trước khi các controller hoạt động. Các tiến trình routing v{ security đ~ được thực hiện.
- `post_controller_constructor`: được gọi ngay sau khi hàm tạo của controller được thi hành. Các hàm trong controller vẫn chưa được gọi.
- `post_controller`: được gọi ngay sau khi controller thực hiện xử lý yêu cầu.
- `display_override`: thực hiện việc chồng (override) hàm `__display()` để hoàn tất nội dung trang trước khi gửi đến trình duyệt của người dùng. Nội dung trang có thể được lấy bằng cách gọi `$this->CI->output->get_output()`.
- `cache_override`: thực hiện việc chồng hàm `_display_cache()` để lấy trang trong bộ đệm.
- `scaffolding_override`: không sử dụng từ CodeIgniter 1.6.
- `post_system`: được gọi sau khi hệ thống đ~ thực hiện xong một yêu cầu.

2.4. CodeIgniter URL

Theo mặc định, cấu trúc URL của CodeIgniter được thiết kế dựa vào các segment thay cho kiểu query truyền thống. Cách tiếp cận này giúp URL trở nên ngắn gọn, có ý nghĩa, dễ ghi nhớ và thân thiện với các bộ máy tìm kiếm. Một URL trong CodeIgniter có dạng:

```
domain.com/index.php/controller/method/param/…
```

Trong đó:

- Segment `controller` là tên của lớp controller được gọi.
- Segment `method` là tên của phương thức trong lớp controller ở trên.
- Segment `param` l{ c|c đối số của phương thức đó

Như vậy, URI `domain.com/index.php/product/view/1` sẽ mang ý nghĩa: Gọi đến phương thức `view()` thuộc lớp `Product` với đối số truyền vào bằng `1`.

Định tuyến URI (URI Routing)

Trong thực tế, URI `domain.com/index.php/product/view/1` lại không thân thiện với các bộ máy tìm kiếm bằng URI `domain.com/index.php/product/view/appple-ipad-1`. CodeIgniter cho phép lập trình viên có thể thay đổi cơ chế xử lý URL mặc định thông qua chức năng Định hướng URI bằng cách thiết lập các quy luật trong tập tin `application/config/routes.php`. Tập tin này chứa mảng `$route`, với khóa là URI mới và giá trị l{ URI cũ. Các quy luật được đọc theo thứ tự từ trên xuống, quy luật trước có độ ưu tiên cao hơn quy luật sau. CodeIgniter cho phép sử dụng các ký tự đại diện (wildcard) để thay thế. CodeIgniter đ~ quy định sẵn hai ký tự đại diện:

- `:num` tương ứng với các giá trị số.
- `:any` tương ứng với bất kỳ giá trị nào.

Ví dụ:

```
$route['product/view/(:num)'] = "product/view/$1";
```

Ý nghĩa: Gọi đến phương thức `view()` trong lớp `Product` với tham số truyền vào là giá trị kiểu số.

```
$route['product/:any'] = "product/find_product";
```

Ý nghĩa: Tất cả các yêu cầu bắt đầu bằng `product` sẽ gọi đến hàm `find_product()` của lớp `Product`.

Ta cũng có thể sử dụng biểu thức chính quy (Regular Expression) để thiết lập các quy luật. Chẳng hạn:

```
$route['product/([a-z]+)/(\d+)'] = "$1/id_$2";
```

Thêm hậu tố vào URL

Các bộ máy tìm kiếm có xu hướng thân thiện hơn với các tập tin HTML. Bằng

cách thay đổi giá trị biến `$config['url_suffix']` trong tập tin `system/application/config/config.php`, ta có thể thêm hậu tố HTML hay bất cứ hậu tố nào khác vào cho URL. Ví dụ, với `$config['url_suffix'] = '.html'`, URL của ta sẽ như sau:

```
domain.com/index.php/controller/method/param.html
```

Loại bỏ chuỗi index.php trong URL

Theo mặc định, chuỗi `index.php` được thêm v{o URL. Để loại bỏ chuỗi này, ta có thể sử dụng một tập tin `.htaccess` có nội dung như sau:

```
Options +FollowSymLinks All -Indexes
RewriteEngine On
RewriteCond $1 !^(index\.php|resources|robots\.txt)
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php/$1 [L,QSA]
```

2.5. Model

Model là những lớp được xây dựng nhằm thực hiện việc trao đổi thông tin với cơ sở dữ liệu. Một lớp Model có thể thực hiện các tác vụ truy vấn, thêm, xóa, cập nhật dữ liệu. Trong CodeIgniter, việc khai báo các lớp model dành cho một thực thể n{o đó l{ không cần thiết, vì trong controller của thực thể đó, ta có thể gọi đến thư viện database để thực hiện các thao tác với cơ sở dữ liệu. Tuy nhiên, để tiện cho việc quản lý, xây dựng lớp model cho một thực thể được khuyến khích.

Một lớp model chuẩn trong CodeIgniter có cấu trúc như sau:

```
class Example_model extends Model {
    /**
*     Hàm tạo      */
    function __contruct()
    {
        parrent::__contruct();
    }

    /**
*     Mô tả hàm
*     @param kiểu dữ liệu $param mô tả biến
     */
    function exampleFunction($param)
    {
        // Do something here
    }
    …
}
```

Khi khai báo một lớp model, ta cần tuân theo một số quy tắc:

- Tên lớp được viết hoa chữ đầu tiên, phần còn lại viết thường. Ví dụ: `User_model`, `Blog_model`, `Article_model`...
- Tên tập tin được đặt như tên lớp, v{ được đặt trong thư mục `application/models/`. C|c thư mục có thể được lưu lồng v{o nhau để thuận tiện cho việc quản lý. Ví dụ: `application/models/user_model.php`, `application/models/front_end/blog_model.php`, `application/models/back_end/article_model.php`...
- Bắt buộc phải kế thừa từ lớp `Model`. Trong hàm tạo của lớp con, phải gọi đến hàm tạo của lớp cha.

Để sử dụng model trong controller, ta sử dụng đoạn mã sau:

```
$this->load->model('model_name');
```

Trong đó, *model_name* là tên của lớp model. Nếu lớp model được lưu trong một thư mục, ta cần ghi rõ đường dẫn đến thư mục đó. Ví dụ lớp `Blog_model` được lưu trong thư mục `application/models/front_end/blog_model.php`. Để sử dụng lớp này, ta sử dụng đoạn mã sau:

```
$this->load->model('front_end/blog_model');
```

Sau khi load lớp model, ta có thể sử dụng lớp n{y trong chương trình bằng cách gọi

```
$this->model_name->method();
```

Để gán một tên khác cho lớp model khi sử dụng trong chương trình, ta có thể truyền vào tham số thứ hai như sau:

```
$this->load->model('model_name', 'ModelName');
```

Khi đó, để sử dụng c|c phương thức của lớp model, ta gọi:

```
$this->ModelName->method();
```

2.6. View

View là những tập tin HTML được xây dựng nhằm thể hiện dữ liệu trong model thành các giao diện tương t|c với người dùng. View có thể là một trang web hoàn chỉnh, hay chỉ là một phần của trang web (header, footer, sidebar...). Nội dung của tập tin view, ngoài mã HTML còn có thể chứa mã PHP. View không bao giờ được gọi trực tiếp mà phải thông qua controller.

Để sử dụng view trong controller, ta sử dụng đoạn mã sau:

```
$this->load->view('view_name', $data);
```

Trong đó, `view_name` là tên của view, `$data` chứa các dữ liệu sẽ được hiển thị trong view. Cũng giống như model, ta có thể lưu view trong c|c thư mục để tiện cho việc quản lý.

Khi sử dụng, chỉ cần khai b|o đường dẫn tương đối đến thư mục chứa view. Ví dụ tập tin `header.tpl.php` được lưu trong thư mục `application/views/front_end`. Để hiển thị view này, ta sử dụng đoạn mã sau:

```
$this->load->view('front_end/header.tpl');
```

CodeIgniter cho phép sử dụng nhiều view trong cùng một phương thức của controller. Dữ liệu trong các view sẽ được kết hợp lại với nhau. Ví dụ:

```
class Example extends Controller {     function showExample()
    {
        // Loading multiple views
        $this->load->view('header.tpl');
        $this->load->view('content.tpl');
        $this->load->view('sidebar.tpl');         $this->load-
>view('footer.tpl');
    }
}
```

Để hiển thị dữ liệu trong model ra view, ta có thể truyền một mảng hoặc đối tượng làm đối số thứ hai khi load view. Ví dụ:

```
$data = array(
    'name'  => 'Maxwell Smart',
    'email' => 'maxwell.smart@control.org',
    'age'   => '33'
);

$this->load->view('employee_detail', $data);
```

Hoặc

```
$objEmployee = new Employee();

$this->load->view('employee_detail', $objEmployee);
```

Khi đó, c|c khóa của mảng hoặc các thuộc tính của đối tượng sẽ được chuyển thành các biến để sử dụng trong tập tin view. Dưới đ}y l{ nội dung mẫu của tập tin `employee_detail.php`:

```
<p>Name: <?php echo $name ?></p>
<p>Email: <?php echo $email ?></p>
```

```
<p>Age: <?php echo $age ?></p>
```

Ta cũng có thể sử dụng các cấu trúc điều khiển (`if`, `else`, `switch`...case...), lặp (`for`, `while`, `do...while...`)...bên trong view để hiển thị dữ liệu. Ví dụ sau sẽ hiển thị danh sách các nhân viên:

```
// Employee Controller
$data['employeeList'] = $this->EmployeeModel->getAllEmployee(); $this-
>load->view('employee_list', $data);
```

Tập tin `employee_list.php`

```
<ol> <?php foreach( $employeeList as
$objEmployee ) :
?>
<li>
    <p>Employee ID: <?php $objEmployee->employee_id ?></p>
    <p>Full-name: <?php $objEmployee->full_name ?></p>
    <p>D.O.B: <?php $objEmployee->dob ?></p>
<p>Email: <?php $objEmployee->email ?></p>
</li>
<?php
endforea
ch;
?>
</ol>
```

CodeIgniter còn cho phép lập trình viên có thể lấy về nội dung của view thay vì xuất trực tiếp trình duyệt, bằng cách truyền `true` l{m đối số thứ ba khi tải view.

```
$string = $this->load->view('myfile', '', true);
```

2.7. Controller

Controller là những lớp đóng vai trò trung gian giữa view và model. Controller nhận các yêu cầu từ phía người dùng, kiểm tra chúng trước khi chuyển qua cho model. Sau khi model xử lý yêu cầu và trả dữ liệu về, controller chuyển sang view để hiển thị dữ liệu cho người dùng. Trong CodeIgniter, các lớp controller được gán vào segment thứ nhất trong URI.

Một lớp controller chuẩn trong CodeIgniter có mẫu:

```
class Example extends Controller {
    /**
*     Hàm tạo      */
    function __contruct()
    {
        parrent::__contruct();
    }

    /**
*     Mô tả hàm
*     @param kiểu dữ liệu $param mô tả biến
     */
    function method($param)
    {
        // Do something here
    }
    …
}
```

Khi khai báo một lớp controller, ta cần tuân theo một số quy tắc:

- Tên lớp được viết hoa chữ đầu tiên, phần còn lại viết thường. Ví dụ: `User`, `Blog`, `Article`…
- Tên tập tin được đặt như tên lớp, v{ được đặt trong thư mục `application/controller`s/. C|c thư mục có thể được lưu lồng v{o nhau để thuận tiện cho việc quản lý. Ví dụ: `application/controllers/user.php`, `application/controllers/front_end/blog.php`, `application/controllers/back_end/article.php`…
- Bắt buộc phải kế thừa từ lớp `Controller`. Trong hàm tạo của lớp con, phải gọi đến hàm tạo của lớp cha.

Segment thứ hai trong URI sẽ gọi đến phương thức tương ứng trong controller. Các giá trị của các segment còn lại trong URI chính là các tham số truyền v{o cho phương thứ này. CodeIgniter quy định một phương thức đặc biệt, `index()`. Phương thức này có thể xem như đầu vào của controller, sẽ được tự động gọi trong trường hợp segment thứ hai của URI bị bỏ trống.

Ví dụ: