Bài 5: Load Library Session Trong Codeigniter

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

1. Thế nào là session

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

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

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

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

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

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

2. Load library session

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

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

***Cấu trúc code*:**

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

    public function index(){
    }
}
```

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

3. Hàm Khởi tạo Session

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

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

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

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

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

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

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

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

4. Hàm hủy session

Hủy session có 2 cách tất cả.

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

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

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

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

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

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

5. Thực hành Session

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

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

    public function index(){
```

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

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

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

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

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

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

Kết quả thu được sẽ tương tự như sau:

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

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

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

6. Các vấn đề mở rộng trong session

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

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

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

Sử dụng nó cũng tương tư như sử dụng session:

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

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

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

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

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

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

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

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

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

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

Và định nghĩa table name cho nó là ci_sessions.

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

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