CakePHP Framework: Phương pháp xây dựng một helper

Giả sử, bạn có hàm tạo chuỗi ngẫu nhiên, bạn muốn dùng hàm này ở bất cứ nơi đâu trong view, bạn có thể viết hàm này trong một controller nào đó sau đó dùng requestAction để gọi hàm ra.

Tuy nhiên đây không phải là cách tốt, đó là chưa kể tới việc dùng nhiều requestAction sẽ làm cho ứng dụng bị chậm đi. Chúng ta sẽ giải quyết vấn đề này cách xây dựng một helper!

Có thể hiểu nôm na rằng helper trong CakePHP là một tập hợp các thư viện hữu ích để dùng trong view. CakePHP đã xây dựng sẵn rất nhiều helper: form, html, ajax, number, session, rss, xml, time….

Muốn dùng helper nào thì trong Controller ta phải khai báo thông qua biến

$helpers

Ví dụ:

```
1 <?php
2 var $helpers = array('Html','Form','Javascript','Ajax');
3 ?>
```

Các bạn có thể tìm hiểu về các helper có sẵn của cake bằng cách vào:

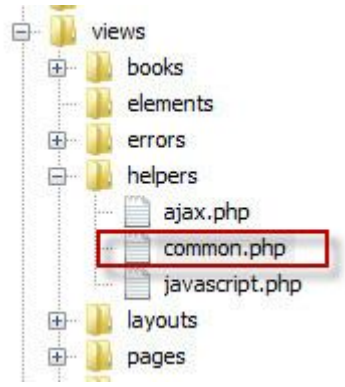
cake/libs/view/helpers

Và bây giờ, chúng ta bắt đầu đi viết một helper cho riêng mình!

Giả sử helper tôi muốn viết có tên là : **Common** => tên file tương ứng là

common.php

Đặt trong **app/views/helpers/**



Chú ý rằng **AppHelper** là lớp cơ sở cho mọi **Helper** .Do do đó, khi tạo ra một Helper mới, bạn có thể e**xtends từ AppHelper** hoặc từ một Helper nào đó có sẵn của CakePHP.

Tên lớp helper = tên helper + "Helper"

Bắt đầu với lớp **Helper với tên Common** , đặt trong thư mục (app/views/helpers/common.php)

Với **hàm tạo 1 chuỗi ngẫu nhiên**.

```
01 <?php
02 class CommonHelper extends HtmlHelper {
03
04  function create_random_string($num) {
05     //Tao du lieu cho hinh ngau nhien
        $chars = array( 'a', 'A', 'b', 'B', 'c', 'C', 'd', 'D', 'e', 'E', 'f', 'F', 'g', 'G', 'h', 'H',
06 'i', 'I', 'j', 'J',  'k', 'K', 'l', 'L', 'm', 'M', 'n', 'N', 'o', 'p', 'P', 'q', 'Q', 'r', 'R', 's', 'S',
    't', 'T',  'u', 'U', 'v', 'V', 'w', 'W', 'x', 'X', 'y', 'Y', 'z', 'Z', '1', '2', '3', '4', '5', '6',
```

```
    '7', '8', '9');
07     $max_chars = count($chars) - 1;
08     for($i = 0; $i < $num; $i++) {
          $code = ( $i == 0 ) ? $chars[rand(0, $max_chars)] : $code .
09
    $chars[rand(0, $max_chars)];
10     }
11     return $code;
12   }
13 }
14 ?>
```

Cách sử dụng lớp **Herpler Common** vừa mới tạo :

- Tôi tạo 1 Controller tên **Testcommons** (app/controllers/ testcommons _controller.php) sử dụng lớp Helper Common vừa tạo

```
1 <?php
2 class TestcommonsController extends AppController {
3     var $helpers = array('Common');
4
```

```
5     function test_helper(){
6         $this->render("test_helper"); // load  file view test_helper.ctp
7     }
8 }
9 ?>
```

- Sử dụng ngoài view : tạo file **test_helper.ctp** (app/views/testcommons/test_helper.ctp)

```
1 <?php
2     echo $this->Common->create_random_string(10);
3 ?>
```

Chạy thử : http://localhost/cakephp/testcommons/test_helper

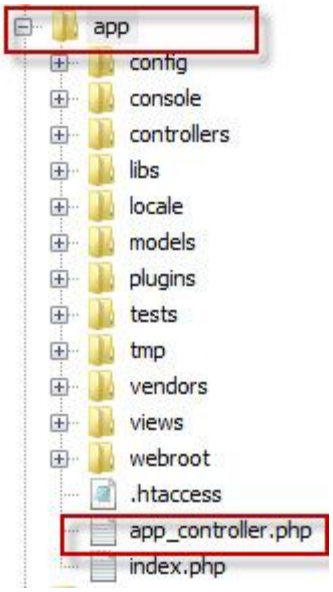
Chú ý :

Biến **$helpers** được khai báo trong Controller nào thì chỉ dùng được trong View của controller đó.

Nếu tôi khai báo trong controller **NewsController** thì sang trang Product, dùng echo $common->create_random_string(6); sẽ bị báo lỗi ngay ! Như vậy không áp dụng được tính chất "dùng mọi lúc, mọi nơi" .

Nhưng không sao, ta có thể giải quyết vấn đề này bằng cách:

- Tạo file **app_controller.php** đặt trong thư mục app, nội dung file này như sau:




```
1 <?php
```

```
2 class AppController extends Controller {

3    var $helpers = array('Html', 'Form','Javascript','Ajax','Common');

4 }

5 ?>
```

Mọi thứ đặt trong AppController sẽ có tác dụng trên toàn bộ các Controller khác, do đó ta chỉ cần khai báo

```
1 var $helpers = array('Html', 'Form','Javascript','Ajax','Common');
```