CAKEPHP FRAMEWORK

1.1. Giới thiệu

CakePHP là một nền tảng phát triển ứng dụng nhanh, mã nguồn mở miễn phí sử dụng ngôn ngữ lập trình PHP. Cấu trúc của nó được tạo ra để lập trình viên tạo các ứng dụng web. Mục tiêu chủ yếu của chúng tôi là tạo ra một nền tảng có cấu trúc, cho phép bạn làm việc trên cấu trúc đó một cách nhanh chóng mà không mất đi sự uyển chuyển.

CakePHP loại bỏ sự nhàm chán trong phát triển ứng dụng web. Chúng tôi cung cấp cho bạn các công cụ bạn cần để viết thứ bạn cần: đó là logic đặc thù của ứng dụng. Thay vì phải làm đi làm lại một thứ khi bạn bắt đầu tạo mới dự án (project), bạn chỉ cần tạo một bản copy của CakePHP và tập trung vào việc chính của dự án.

CakePHP có một đội ngũ phát triển và cộng đồng năng động, điều này mang lại giá trị to lớn cho các dự án. Ngoài việc giúp bạn khỏi phải làm đi làm lại một việc nào đó, sử dụng CakePHP đồng nghĩa với phần cốt lõi của ứng dụng của bạn đã được kiểm chứng và cải tiến không ngừng.

1.2. Tính năng của CakePHP

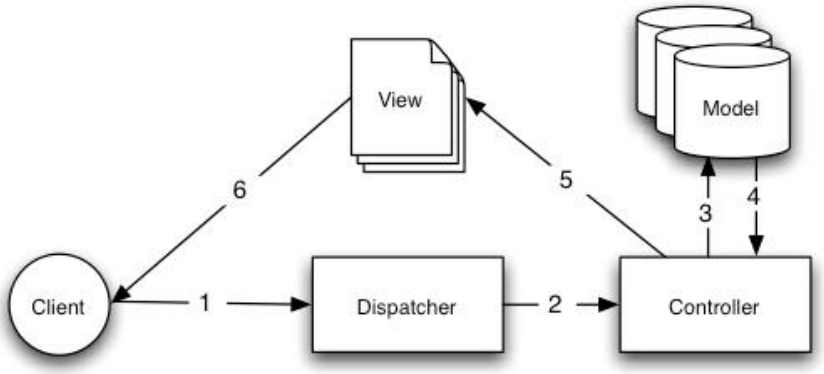
- Cộng đồng năng động, thân thiện
- Việc cấp phép uyển chuyển
- Tương thích với PHP 4 và PHP 5
- Tích hợp sẵn CRUD (Khi làm việc với cơ sở dữ liệu)
- Hỗ trợ làm việc với cơ sở dữ liệu (scaffolding)
- Tự động sinh mã code
- Kiến trúc MVC
- Cho phép tạo ra các URL rõ ràng, dễ hiểu.
- Cung cấp khả năng bắt lỗi
- Cho phép tạo ra các bản mẫu (templating) nhanh chóng và uyển chuyển
- Các tiện ích giao diện cho AJAX, JavaScript, HTML Forms,...
- Có các công cụ xử lý Email, Cookie, Security, Session, yêu cầu (Request Handling)
- Danh sách điều khiển truy cập (ACL) uyển chuyển
- Cung cấp sẵn tiện ích xử lý dữ liệu (Data Sanitization)
- Cung cấp khả năng lưu tạm (Caching) uyển chuyển
- Cung cấp khả năng nội địa hóa (localization)

- Có khả năng chạy từ bất kỳ thư mục web nào mà không cần phải cấu hình Apache hoặc chỉ cấu hình lại rất ít

1.3. Mô hình MVC trong CakePHP

CakePHP tuân theo mô hình thiết kế phần mềm chuẩn (design pattern) MVC. MVC chia chương trình thành 3 phần riêng lẻ:

1. Model: Mô tả dữ liệu của ứng dụng
2. View: Hiển thị dữ liệu của model
3. Controller: Xử lý và điều hướng các yêu cầu của client



Hình 1. Mô hình MVC trong CakePHP

1.4. Cài đặt

1.4.1. Cài đặt

Dowload bộ cakePHP version 1.2 tại:

http://github.com/cakephp/cakephp/archives/1.2

1.4.2. Cấu hình

Để có thể kết nối với database , ta phải cấu hình lại file database.php trong thư mục app/config như sau :

```
var $default = array(
  'driver' => 'mysql',
  'persistent' => 'false',
  'host' => 'localhost',
  'port' => '',
  'login' => 'account',
  'password' => 'password',
  'database' => 'tên database,
  'schema' => '',
  'prefix' => '',
  'encoding' => ''
);
```

Như vậy cakePHP đã được cấu hình thành công và có thể kết nối với cơ sở dữ liệu

1.5. Cấu trúc thư mục của CakePHP

Sau khi bạn download và giải nén CakePHP, bạn cần lưu ý đến 3 thư mục chính sau:

- Thư mục *app* là nơi mà bạn làm việc: Nơi đây lưu các file của ứng dụng do bạn viết.
- Thư mục *cake* là nơi bạn thay đổi các file core của CakePHP. Bạn phải thật cẩn thận khi thay đổi các file trong này, và chúng tôi không thể giúp gì cho bạn nếu bạn thay đổi các file core.
- Thư mục *vendors* là nơi bạn cài các thư viện PHP của nhà cung cấp thứ 3 mà bạn cần để sử dụng với ứng dụng CakePHP của bạn.

1.5.1. Thư mục App

Thư mục app của CakePHP là nơi bạn chính bạn phát triển ứng dụng. Bây giờ chúng ta sẽ xem qua các thư mục phía trong của thư mục app.

config	Nơi lưu giữ một số file cấu hình của CakePHP. File lưu thông tin kết nối đến cơ sở dữ liệu, file bootstrap (dùng để nạp các hàm không nằm trong một lớp nào cả), file cấu hình CakePHP nên được lưu ở đây.
controllers	Chứa các controller và các component của ứng dụng.
locale	Lưu các file văn bản được sử dụng cho mục đích quốc tế hóa ứng dụng.
models	Chứa các file model, behavior và datasource của ứng dụng.
plugins	Chứa các plugin.
tmp	Lưu các file tạm của CakePHP. Dữ liệu thực CakePHP lưu phụ thuộc vào bạn cấu hình CakePHP như thế nào, tuy nhiên thư mục này thường được sử dụng để lưu các mô tả về model, logs, và đôi khi là thông tin session. Bạn phải chắc chắn rằng thư mục này tồn tại và có thể ghi được nếu không thì hiệu năng của ứng dụng sẽ bị ảnh hưởng nghiêm trọng. Ở chế độ sửa lỗi (debug), CakePHP sẽ cảnh báo cho bạn nếu thư mục này không tồn tại hoặc không ghi được.
vendors	Các lớp hoặc các thư viện của nhà cung cấp thứ 3 nên được để trong thư mục này. Làm như vậy sẽ giúp cho các file này dễ dàng được truy cập bằng cách sử dụng hàm App::Import('vendor', 'name'). Thoạt nhìn qua thì thư mục này có vẻ là thừa vì có 1 thư mục vendors khác nằm ở ngoài. Chúng ta sẽ tìm hiểu sự khác nhau của các thư mục này khi chúng ta thảo luận về quản lý thiết lập nhiều ứng dụng và hệ thống phức tạp..
views	Các file hiển thị được để ở đây: các file element, trang thông báo

	lỗi, các file helper, các file layout và các file view.
webroot	Trong môi trường thật (CakePHP có thể chạy ở môi trường development, test, production), thư mục này nên đóng vai trò là thư mục gốc của ứng dụng. Các thư mục ở đây thường là các thư mục cho css, ảnh và javascript.

1.6. Các quy ước của CakePHP

1.6.1. Các quy ước về file và tên lớp

Nói chúng, tên file sử dụng dấu gạch dưới, trong khi tên lớp sử dụng CamelCased (ví dụ như SportCar). Vì vậy nếu bạn có lớp **MyNiftyClass**, khi đó trong CakePHP tên của file đó nên là **`my_nifty_class.php`**. Dưới đây là các ví dụ về cách đặt tên file cho mỗi dạng lớp khác nhau mà bạn thường sử dụng trong ứng dụng CakePHP:

- Định nghĩa lớp controller **KissesAndHugsController** nằm trong file tên là **`kisses_and_hugs_controller.php`** (Lưu ý `_controller` trong tên file)
- Định nghĩa lớp component **MyHandyComponent** nằm trong file tên là **`my_handy.php`**
- Định nghĩa lớp model **OptionValue** nằm trong file **`option_value.php`**
- Định nghĩa lớp behavior **EspeciallyFunkableBehavior** nằm trong file **`especially_funkable.php`**
- Định nghĩa lớp view **SuperSimpleView** nằm trong fle **`super_simple.php`**
- Định nghĩa lớp helper **BestEverHelper** nằm trong file **`best_ever.php`**

Mỗi file được đặt trong 1 thư mục quy ước dưới thư mục app.

1.6.2. Quy ước về tên model và tên bảng cơ sở dữ liệu

Model classnames là số ít và 'CamelCased' Person, BigPerson, hay ReallyBigPerson là những ví dụ về tên Model thông thường.

Tên bảng cơ sở dữ liệu tương ứng cho cakePHP model là số nhiều và được gạch dưới. Các bảng sẽ được đề cập tới bởi các model tương ứng là people,big_people và really_big_people tương ứng.

Bạn có thể sữ dụng thư viện tiện ích "Inflector" để kiểm tra từ số nhiều hay số ít.

Tên trường có hai hay nhiều từ sẽ được nối bằng gạch nối dưới như : first_name.

Những khóa ngoại trong các mối quan hệ hasMany,belongsTo hoặc hasOne được chấp nhận mặc định ở dạng tên(số ít) của bảng liên quan theo sau _id. Vì vậy , nếu một Baker hasMany với Cake thì bảng cakes sẽ ràng buộc với bảng bakes theo thông qua một khóa ngoại là baker_id. Đối với một bảng có tên tạo từ nhiều từ như category_types thì khóa ngoại có sẽ là category_type_id.

Kết nối các bảng, được dùng trong kết hợp hasAndBelongsToMany giữa các model sẽ được đặt tên sau theo bảng model sẽ kết nối theo thứ tự alphabetical.

Tất cả các bảng mà các model của CakePHP tương tác tới đều cần một khóa chính duy nhất để nhận định mỗi hàng. Nếu bạn muốn mô hình hóa một bảng mà không có một khóa chính nào thì một trường khóa chính đơn sẽ được thêm vào bảng.

CakePHP không hổ trợ tạo những khóa chính. Nếu bạn muốn thao tác trực tiếp trên bảng dữ liệu, hãy sữ dụng câu truy vấn trực tiếp hoặc thêm khóa chính để nó hoạt động như một mô hình bình thường.

```
CREATE TABLE posts_tags (
      id INT(10) NOT NULL AUTO_INCREMENT,
      post_id INT(10) NOT NULL,
      tag_id INT(10) NOT NULL,
      PRIMARY KEY(id));
)
```

Thay vì sử dụng một mã khóa tự động tăng là khóa chính, bạn cũng có thể sử dụng char (36). Cake sau đó sẽ sử dụng 36 ký tự UUID (String:: uuid) bất cứ khi nào bạn lưu một bản ghi mới bằng cách sử dụng phương thực Model::save.

1.6.3. Quy ước về Controller

Tên lớp Controller ở dạng số nhiều và "CamelCased" và kết thúc bằng Controller, ví dụ: `PeopleController` và `LatestArticlesController`.

Phương thức mặc định khi bạn gọi một controller mà không chỉ định chính xác là gọi action nào là *index()*. Ví dụ: có một yêu cầu tới http://www.example.com/apples/ thì cakePHP sẽ tự động gọi phương thức *index()* của lớp `ApplesController`. Và nếu gọi tới http://www.example.com/apples/view/ thì phương thức view trong controller đó sẽ được thực hiện.

1.6.4. Quy ước về View

File trong view được đặt tên sau hàm chức năng, sau dấu gạch ngang. Hàm getReady() của lớp PeopleController sẽ tìm thấy trong /app/views/people/get_ready.ctp

Mẫu cơ bản là /app/views/controller/underscored_function_name.ctp.

Bằng cách đặt tên các thành phần ứng dụng sử dụng theo quy ước của CakePHP bạn có được chức năng mà không có sự rắc rối và dễ cấu hình

Một số ví dụ sử dụng theo quy ước

- Database table: "people"
- Model class: "Person", tại thư mục /app/models/person.php
- Controller class: "PeopleController", ở thư mục /app/controllers/people_controller.php
- View template, tại thư mục /app/views/peoples/index.ctp

Sử dụng những quy ước , CakePHP biết rằng yêu cầu từ http://example.com/people/ có phương án gọi hàm index() của Peoplecontroller, nơi mà model Person tự động có sẵn (và tự động gắn với bảng people trong cơ sở dữ liệu), và dùng file index.ctp để hiển thị ra trình duyệt của người sử dụng.

1.7. Các thành phần chính của CakePHP

1.7.1. Model

Models đại diện cho dữ liệu và được dùng trong các ứng dụng CakePHP để truy cập dữ liệu. Một Model thường đại diện cho một bảng dữ liệu nhưng có thể được dùng để truy cập bất cứ những gì được lưu trữ dữ liệu như các tập tin, bản ghi LDAP, các sự kiện iCal, hoặc các hàng trong một tập tin CSV.

Một Model có thể được ràng buộc với các Models khác. Ví dụ như : một Recipe có thể được ràng buộc với Author của Recipe bằng Ingredient trong Recipe.

Phần này sẽ giải thích những điểm đặc trưng của Models có thể được tự động hóa, làm thế nào để có thể ghi đè lên những điểm đó, và những thức, thuộc tính nào mà một Model có thể có. Nó sẽ trình bày những cách khác nhau để kết hợp cơ sở dữ liệu của bạn, Nó sẽ thể hiện làm thế nào để tìm kiếm, lưu trữ và xóa dữ liệu. Cuối cùng là xem xét Datasources.

1.7.1.1. Hiểu biết về Models

Một Model đại diện cho mô hình dữ liệu của bạn. Trong lập trình hướng đối tượng một mô hình dữ liệu là một đối tượng đại diện cho một một thứ, như một chiếc xe, một người hay một ngôi nhà. Ví dụ : một blog có thể có nhiều bài viết (Blog Post) và mỗi bài viết có thể có nhiều ý kiến (Comment). Blog, Post, Comment là những ví dụ cho Models, mỗi cái sẽ được ràng buộc với các cái khác.

Đây là một ví dụ đơn giản về việc định nghĩa một Model trong CakePHP:

```
<?php
class Ingredient extends AppModel
{
            var $name = 'Ingredient';
}
?>
```

Chỉ cần với khai báo đơn giản như trên, Model Ingredient được đặt cho tất cả những chức năng bạn cần để tạo câu truy vấn để lưu và xóa dữ liệu. Chúng là những phương thức kỳ diệu từ lớp Model của CakePHP bởi tính kỳ diệu của tính thừa kế. Model Ingredient mở rộng ứng dụng Model, AppModel, được mở rộng từ lớp Model nội tại của CakePHP. Đó là cốt lõi của lớp Model để đặt các chức năng cho Model Ingredient của bạn.

Tạo tập tin PHP cho Model của bạn trong /app/models/ hoặc trong thư mục con của /app/models/ . CakePHP sẽ tìm kiếm nó ở một nơi bất kỳ trong thư mục. Theo quy ước nó cần có cùng tên với lớp. Ví dụ : ingredient.php

Với Models đã được định nghĩa, nó có thể được truy cập từ bên trong Controller của bạn. CakePHP tự động làm cho Model cho phép truy cập khi nó có tên phù hợp

với Controller. Ví dụ : Một Controler được đặt tên là IngredientsController sẽ tự động khởi tạo model Ingredient và gắn nó vào controller bằng $this->Ingredient.

```
<?php
class IngredientsController extends AppController
{
    function index()
    {
        //grab all ingredients and pass it to the view:
        $ingredients = $this->Ingredient->find('all');
        $this->set('ingredients', $ingredients);
    }
}
?>
```

Trong ví dụ sau đây, Recipe có một sự ràng buộc với Ingredient model

```
<?php
class RecipesController extends AppController {
    function index() {
        $ingredients = $this->Recipe->Ingredient->find('all');
        $this->set('ingredients', $ingredients);
    }
}
?>
```

Nếu các Model thực sự không có sự ràng buộc lẫn nhau thì bạn có thể sữ dụng phương thức Contrller::loadModel() để có được Model.

```
<?php
class RecipesController extends AppController {
    function index() {
       $recipes = $this->Recipe->find('all');
       $this->loadModel('Car');
       $cars = $this->Car->find('all');
       $this->set(compact('recipes', 'cars'));
    }
}
?>
```

1.7.1.2. Tạo bảng cơ sở dữ liệu

Trong khi CakePHP có thể có datasources mà không phải là cơ sở dữ liệu điều khiển , hầu hết thời gian chúng có . CakePHP được thiết kế cho thuyết bất khả tri và sẽ làm việc với MySQL, MSSQL, Oracle, PostgreSQL và những hệ quản trị khác. Bạn có thể tạo các bảng cơ sở dữ liệu như bình thường. Khi bạn tạo các lớp Model thì chúng sẽ tự động định nghĩa các bảng mà bạn đã tạo.

Tên bảng được quy ước viết bằng chữ thường và các bảng tạo bởi nhiều từ thì được nối với nhau bằng dấu gạch dưới. Ví dụ, một Model tên là Ingredient sẽ dành cho bảng có tên là ingredients. Một model tên là EventRegistration sẽ dành cho bảng có tên là event_registrations. CakePHP sẽ kiểm tra để xác định dạng dữ liệu của từng trường trong bảng và sữ dụng thông tin này để tự động hóa các tính năng khác nhau như trong việc xuất các trường dữ liệu lên View.

Tên các trường của bảng cơ sở dữ liệu cũng được quy ước viết bằng chữ thường và nối với nhau bằng dấu gạch dưới.

Trong phần còn lại của phần này bạn sẽ được thấy làm thế nào để CakePHP định nghĩa cơ sở dữ liệu.

Mỗi hệ quản trị cơ sở dữ liệu định nghĩa loại dữ liệu theo những cách khác nhau. Từ lớp mã nguồn cho đến hệ thống dữ liệu, CakePHP vẽ ra trên một giao diện thống nhất, không có vấn đề mà hệ thống cơ sở dữ liệu bạn cần để chạy trên nó.

CakePHP Type	Field Properties
primary_key	NOT NULL auto_increment
string	varchar(255)
text	text
integer	int(11)
float	float
datetime	datetime
timestamp	datetime
time	time
date	date
binary	blob
boolean	tinyint(1)

Hình 2. Kiểu dữ liệu MySQL tương ứng với CakePHP

CakePHP Type	Field Properties
primary_key	number NOT NULL
string	varchar2(255)
text	varchar2
integer	numeric
float	float
datetime	date (Y-m-d H:i:s)
timestamp	date (Y-m-d H:i:s)
time	date (H:i:s)
date	date (Y-m-d)
binary	bytea
boolean	boolean
number	numeric
inet	inet

Hình 3. Kiểu dữ liệu Oracle tương ứng với CakePHP

1.7.1.3. Các phương thức thường dùng trong Model

- Find - **find($type, $params);**

$type là trường hợp dễ bị lỗi. Chỉ một kí tự viết hoa (ví dụ : 'All')sẽ không cho kết quả như mong đợi.

$param được dùng để truyền các tham số tìm kiếm như :

```
   array(
      'conditions'  =>  array('Model.field'  =>  $thisValue),  //array  of
conditions
      'recursive' => 1, //int
      'fields' => array('Model.field1', 'DISTINCT Model.field2'), //array
of field names
      'order' => array('Model.created', 'Model.field3 DESC'), //string or
array defining order
      'group' => array('Model.field'), //fields to GROUP BY
      'limit' => n, //int
      'page' => n, //int
      'offset'=>n, //int
      'callbacks'  =>  true  //other  possible  values  are  false,  'before',
'after'
   )
```

- **field** – `field(string $name,array $conditions=null, string $order=null)`

Trả về giá trị của một trường duy nhất được chỉ định bằng biến $name từ bản ghi đầu tiên thỏa mản $conditions với thứ tự sắp xếp bởi biến $order

```
   $model->id = 22;
   echo $model->field('name'); // echo the name for row id 22

   echo  $model->field('name',  array('created <' => date('Y-m-d H:i:s')),
'created DESC'); // echo the name of the last created instance
```

- **read()**

Read() là phương thức được dùng để thiết lập mô hình dữ liệu hiện hành. Nhưng nó còn được dùng trong trường hợp khác, là để lấy một bản ghi ra từ cơ sỡ dữ liệu.

```
   function beforeDelete($cascade) {
      ...
      $rating  =  $this->read('rating');  //  gets  the  rating  of  the  record
being deleted.
      $name  =  $this->read('name',  $id2);  //  gets  the  name  of  a  second
record.
      $rating  =  $this->read('rating');  //  gets  the  rating  of  the  second
record.
      $this->id = $id3; //
      $this->read(); // reads a third record
      $record = $this->data // stores the third record in $record
      ...
   }
```

- **Lưu trữ dữ liệu**

CakePHP lưu trữ dữ liệu rất đơn giản . Dữ liệu sẵn sàng để được lưu sẽ truyền qua cho phương thức save() của lớp Model sữ dụng dạng cơ bản sau:

```
Array
(
    [ModelName] => Array
        (
            [fieldname1] => 'value'
            [fieldname2] => 'value'
        )
)
```

Dưới đây là một ví dụ đơn giản để điều khiển sử dụng một model CakePHP để lưu dữ liệu vào một bảng cơ sở dữ liệu:

```
function edit($id) {
    //Has any form data been POSTed?
    if(!empty($this->data)) {
        //If the form data can be validated and saved...
        if($this->Recipe->save($this->data)) {
            //Set a session flash message and redirect.
            $this->Session->setFlash("Recipe Saved!");
            $this->redirect('/recipes');
        }
    }

    //If no form data, find the recipe to be edited
    //and hand it to the view.
    $this->set('recipe', $this->Recipe->findById($id));
}
```

Sau đây là một số phương thức lưu dữ liệu trong model

```
set($one, $two = null)
save(array $data=null, boolean $validate = true, array $fieldList =
array())
save(array $data = null, array $params = array())
create(array $data = array())
saveField(string $fieldName, string $fieldValue, $validate = false)
updateAll(array $fields, array $conditions)
saveAll(array $data = null, array $options = array())
```

- **delete – Xóa dữ liệu**

```
delete(int $id = null, boolean $cascade = true);
```

Xóa bản ghi được chỉ định bởi biến $id. Mặc định thì xóa cả những bản ghi có liên quan tới bản ghi được chỉ định. Ví dụ khi xóa một User thì các thông tin liên quan tới User đó cũng được xóa.

- **remove**

```
remove(int $id = null, boolean $cascade = true);
```

sử dụng như hàm delete()

- **deleteAll**

```
deleteAll(mixed $conditions, $cascade = true, $callbacks = false)
```

Giống như với delete() và remove() , nhưng deleteAll() xóa tất cả các bản ghi được chỉ định bởi một điều kiện ràng buộc nào đó.

1.7.1.4. Các loại quan hệ trong Model

CakePHP có 4 kiểu quan hệ như sau:

Relationship	Association Type	Example
one to one	hasOne	A user has one profile.
one to many	hasMany	A user can have multiple recipes.
many to one	belongsTo	Many recipes belong to a user.
many to many	hasAndBelongsToMany	Recipes have, and belong to many tags.

Hình 4. Các loại quan hệ của Model

Các liên kết được định nghĩa bằng cách tạo ra một lớp biến được đặt tên sau liên kết mà bạn đang định nghĩa. Lớp biến đó đôi khi có thể là một chuỗi , nhưng có thể là một mảng đa chiều để xác định cụ thể mối liên kết.

- **hasOne**

Hãy thiết lập một model User với một mối quan hệ hasOne với model Profile.

Trước tiên, các bảng cơ sỡ dữ liệu cua bạn phải được đặt khóa chính xác.Đối với một mối quan hệ hasOne để làm việc , một trong những bảng có chứa khóa ngoại đến một bảng khác. Trong trường hợp này bảng Profile sẽ chứa một trường được gọi là user_id. Mô hình cơ bản là:

Relation	Schema
Apple hasOne Banana	bananas.apple_id
User hasOne Profile	profiles.user_id
Doctor hasOne Mentor	mentors.doctor_id

Hình 5. Quan hệ hasOne

- **belongsTo**

Bây giờ chúng ta có dữ liệu Profile truy cập từ model User, hãy định nghĩa kết hợp belongsTo trong model Profile. Kết hợp belongsTo là sự bổ sung tự nhiên cho kết hợp hasOne và hasMany: Nó cho phép chúng ta xem dữ liệu từ một phương diện khác.

Khi đặt khóa cho bảng dữ liệu đối với mối liên kết belongsTo phải theo quy ước sau:

Relation	Schema
Banana belongsTo Apple	bananas.apple_id
Profile belongsTo User	profiles.user_id
Mentor belongsTo Doctor	mentors.doctor_id

Hình 6. Quan hệ belongsTo

- **hasMany**

Bước tiếp theo là định nghĩa một kết hợp 'User hasMany Comment' . Một kết hợp hasMany sẽ cho phép chúng ta lấy comment của user khi chúng ta lấy một bản ghi của user đó.

Khi đặt khóa cho bảng dữ liệu đối vơi một liên kết hasMany phải theo quy ước sau:

Relation	Schema
User hasMany Comment	Comment.user_id
Cake hasMany Virtue	Virtue.cake_id
Product hasMany Option	Option.product_id

Hình 7. Quan hệ hasMany

3.7.6.5 hasAndBelongsToMany

Bây giờ là dạng kết hợp cuối cùng : hasAndBelongsToMany, hoặc HABTM. Kết hợp này được dùng khi bạn có hai model cần kết nối với nhau liên tục nhiều lần va nhiều cách khác nhau.

Sự khác biệt chính giữa hasMany và HABTM là một liên kết giữa các model trong HABTM không phải độc quyền. Trong khi đó liên kết giữa các model trong hasMany thì độc quyền. Nếu User hasMany với Comments thì một comment chỉ có thể liên kết tới một user xác định.

Khi đặt khóa cho bảng dữ liệu đối vơi một liên kết hasMany phải theo quy ước sau:

Relation	Schema (HABTM table in bold)
Recipe HABTM Tag	**recipes_tags**.id, **recipes_tags**.recipe_id, **recipes_tags**.tag_id
Cake HABTM Fan	**cakes_fans**.id, **cakes_fans**.cake_id, **cakes_fans**.fan_id
Foo HABTM Bar	**bars_foos**.id, **bars_foos**.foo_id, **bars_foos**.bar_id

Hình 8. Quan hệ hasAndBelongsToMany

1.7.2. View

View là nơi thể hiện dữ liệu đã được xử lý của chúng ta. Một view được xem như một trang template. .

Lớp View của CakePHP là nơi bạn trình bày cho người dùng của bạn. Những gì biểu diến sẽ được đưa vào các tập tin (X)HTML và chuyển đến trình duyệt.

CakePHP xem các tập tin được viết bằng PHP và có phần mở rộng mặc định **.ctp** (CakePHP Template). Những tập tin này chứa tất cả logic thông dụng cần để lấy dữ liệu từ controller trong một định dạng mà đã sản sàng cho các đối tượng bạn đang phục vụ.

Tập tin của View được lưu trong /app/views trong một thư mục sau khi controller sử dụng các tập tin, và được đặt tên theo các hành động tương ứng với nó.

Ví dụ: file view của Products controller là hành động "view()", bình thường sẽ được tìm thấy ở /app/views/products/view.ctp.

Lớp view trong CakePHP có thể được tạo thành từ một số bộ phận khác:

- **layouts**: xem các file chứa các mã thông dụng và được tìm trong nhiều gói giao diện ứng dụng của bạn. Hầu hết các trình trình bày được trả bên trong một bản mẫu.
- **elements**: dùng để thiết hổ trợ các view giống nhau khỏi mất thời gian code lại nhiều lần.Elements thường được trả bên trong các view.
- **helpers**: Một lớp Helper là một lớp tiện ích được dùng để xử lý các logic trong view. Cũng giống như component của controller, các view có thể dùng chung một hoặc nhiều lớp helper. Một trong những lớp helper phổ biến là AjaxHelper, lớp này giúp cho việc viết ajax trong view trở nên dễ dàng hơn.

1.7.2.1. Layout

Một Layout chứa mã trình bày bao quanh một view. Bất cứ điều gì bạn muốn xem trong tất cả các view đều được đặt trong một layout.

Layout được đặt trong /aap/views/layouts. Layout mặc định của CakePHP có thể được ghi đè lên bằng cách tạo ra một layout mặc định mới tại /app/views/layouts/default.ctp. khi một layout mặc định mới được được tạo ra. Controller –chuyển đến layout mặc định được chỉ định.

Khi tạo một layout , bạn cần phải báo cho CakePHP nơi bạn đặt code cho view của bạn. Để làm vậy , chắc chắn nơi đặt layout của bạn cho $content_for_layout(nội dung của layout)và $title_for_layout(tên của layout).

Dưới đây là một ví dụ về layout mặc định:

```
   <!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 $title_for_layout?></title>
   <link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
   <!-- Include external files and scripts here (See HTML helper for more
info.) -->
   <?php echo $scripts_for_layout ?>
   </head>
   <body>
   <!-- If you'd like some sort of menu to
   show up on all of your views, include it here -->
   <div id="header">
   <div id="menu">...</div>
   </div>
   <!-- Here's where I want my views to be displayed -->
   <?php echo $content_for_layout ?>
   <!-- Add a footer to each displayed page -->
   <div id="footer">...</div>
   </body>
   </html>
```

Biến **$scripts_for_layout** chứa bất kỳ tập tin bên ngoài và các bản có trong xây dựng trong HTML helper. Hữu ích cho cả javascript và CSS

Bạn có thể tạo nhiều layout như bạn muốn: chỉ cần đặt chúng trong app/views/layouts và chuyển đổi giữa chúng bên trong hành động điều khiển của bạn bằng cách sử dụng biến $layout , hoặc hàm setLayout () .

Ví dụ, nếu một phần của trang web của tôi bao gồm một không gian quảng cáo banner nhỏ, tôi có thể tạo ra một bố trí mới với không gian quảng cáo nhỏ hơn và chỉ định nó như là cách bố trí cho hành động của tất cả các điều khiển bằng cách sử dụng như:

```
<?php
class UsersController extends AppController {
function viewActive() {
$this->pageTitle = 'View Active Users';
   $this->layout = 'default_small_ad';
}
function viewImage() {
$this->layout = 'image';
//output user image
}
}
?>
```

CakePHP bố trí các tính năng cốt lõi (bên cạnh việc bố trí layout mặc định của CakePHP) bạn có thể sử dụng trong các ứng dụng riêng của bạn: 'ajax' và flash '. Ba layout khác xml, js, và rss tồn tại trong core cho một cách nhanh chóng và dễ dàng để phục vụ lên nội dung không phải là text / html.

1.7.2.2. Các phương thức thường dùng trong View

- ***set()***

```
set(string $var, mixed $value)
```

Views có hàm set() tương tự như set() của đối tượng controller. Nó cho phép bạn thêm các biến vào view vars. Sử dụng set() từ view cho phép thêm vào các biến gửi đến layout và elements sẻ được triệu gọi sau đó. Sử dụng set() là cách chính để gửi dữ liệu từ controller đến view của bạn. Một khí bạn sữ dụng set() biến đó có thể được truy cập trong view.

Trong view bạn có thể làm:

```
$this->set('activeMenuButton', 'posts');
```

Sau đó trong layout biến $activeMenuButton sẽ có sản và có giá trị “posts”

- ***getVar()***

```
getVar(string $var)
```

lấy giá trị của viewVar với tên là $var

- ***getVars()***

lấy tất cả danh sách các biến có sãn trong phạm vi hiện hành. Trả về một mảng các biến.

- ***error()***

```
error(int $code, string $name, string $message)
```

hiển thị một trang báo lỗi cho người sử dụng. sử dụng layout/error.ctp để trả lại trang.

```
$this->error(404, 'Not found', 'This page was not found, sorry');
```

Điều này sẽ làm cho một trang báo lỗi với thông báo được chỉ định. Quan trọng của nó là bản thảo thực hiện không dừng lại được bởi View::error(). Vì vậy bạn phải ngừng thực thi mã của bạn nếu bạn muốn tạm dừng lại.

- ***element()***

```
element(string $elementPath, array $data, bool $loadHelpers)
```

Chuyển đến một element hoặc xem một phần . Xem phần về element cho biết cho biết thêm nhiều thông tin và ví dụ.

- ***uuid()***

```
uuid(string $object, mixed $url)
```

Tạo ra một DOM ID không ngẫu nhiên duy nhất cho một đối tượng, dựa vào loại đối tượng và địa chỉ. Phương pháp này thường dùng bởi các helpers cần tạo ra DOM ID duy nhất cho element chẳng hạn như AjaxHelper.

```
$uuid = $this->uuid('form', array('controller' => 'posts', 'action' => 'index'));
//$uuid contains 'form0425fe3bad'
```

- ***addScript()***

```
addScript(string $name, string $content)
```

Thêm nội dung cho các bộ đệm bản thảo nội bộ. Bộ đệm này được làm sẵn trong layout như là **$scripts_for_layout.** Phương pháp này hữu ích khi tạo các Helpers cần phải thêm javascript hoặc css trực tiếp cho layout. Hãy nhớ rằng bản thảo được thêm từ layout hoặc các element , trong layout sẽ không được thêm vào $scripts_for_layout. Phương pháp này thường được sử dụng từ các Helpers bên trong, giống như Javascript và Html Helpers.

1.7.2.3. Helper

CakePHP có một số Helpers hỗ trợ trong việc tạo ra cách trình bày. Chúng hỗ trợ trong việc tạo ra khuôn mẫu đẹp (bao gồm cả hình thức), hỗ trợ trong định dạng văn bản, thời gian và số, và thậm chí có thể tăng tốc độ chức năng Ajax. Dưới đây là tóm tắt những Helpers.

Helpers của cakePHP	Mô tả
Ajax	Được sử dụng cùng với thư viện Prototype JavaScript để tạo chức năng Ajax trong trình bày. Chứa shortcut cho các phương pháp kéo / thả, hình thức ajax & liên kết, và nhiều hơn nữa.

Cache	Được sử dụng bởi lõi để xem nội dung cache.
Form	Tạo ra các hình thức HTML Form ,các element cùng nơi và xử lý các vấn đề xác nhận.
Html	Html Thuận tiện cho việc tạo các phương pháp đánh dấu cũng như các thành lập. Hình ảnh, liên kết, bảng biểu, các thẻ tiêu đề và nhiều hơn nữa.
Javascript	Javascript được sử dụng để thoát khỏi các giá trị để sử dụng trong JavaScripts, viết ra dữ liệu cho các đối tượng JSON, và khối mã định dạng
Number	số và định dạng tiền tệ.
Paginator	Chia trang dữ liệu và phân loại
Rss	Phương pháp tiện lợi cho xuất dữ liệu XML.
Session	Truy cập để viết ra các giá trị phiên làm việc trong view
Text	Liên kết một cách mạnh mẻ,làm nổi bật, cắt bỏ từ một cách thông minh.
Time	Tìm ra năm tiếp theo ,định dạng đẹp (Hôm nay, 10:30) và chuyển đổi vùng thời gian
Xml	Phương pháp tiện lợi cho việc tạo tiêu đề XML và các element

Với các bạn đã từng làm việc với Cakephp hẳn các bạn không lạ gì việc sử dụng các hepler trong cake như form, html, javascript. Sau đây tôi xin liệt kê một số hàm hay sử dụng nhất trong cakephp, cấu trúc và cách sử dụng chúng.

$javascript->link($url, $inline);

$url (require): đường dẫn tới file javascript. Các file javascript sẽ được đặt trong thư mục 'app/webroot/js'. Mặc định url là đường dẫn tới thư mục app/webroot/js/.

$inline: (boolean) . Nếu giá trị là true, thì thẻ script sẽ được in trong thẻ header , nếu là false thì sẽ in trong $script_for_layout. Giá trị mặc đinh là true.

Ví dụ:

```
<?php echo $javascript->link('script.js', true); ?>
//Dữ liệu trả về html
<script type="text/javascript" src="/test/js/script.js"></script>
```

$html->css(mixed $path, string $rel = null, array $htmlAttributes = array(), boolean $inline = true);

Tạo một link tới file css. Nếu $inline được gán là false, thì thẻ css sẽ được đặt bên trong biến $script_for_layout, còn là true thì đặt trong thẻ header. Mặc định đường dẫn trong của file css là /app/webroot/css/.

Ví dụ:

```
<?php echo $html->css('forms'); ?>
//Dữ liệu trả về html
<link rel="stylesheet" type="text/css" href="/test/css/forms.css" />
//Ta cũng có thể khai báo nhiều file css cùng một lúc
<?php echo $html->css(array('forms','tables','menu')); ?>
//Dữ liệu trả về html
<link rel="stylesheet" type="text/css" href="/test/css/forms.css" />
<link rel="stylesheet" type="text/css" href="/test/css/tables.css" />
<link rel="stylesheet" type="text/css" href="/test/css/menu.css" />
```

$html->image(string $path, array $htmlAttributes = array());

Tạo ra một thẻ định dạng là image. Mặc định nó sẽ chỉ tới thư mục /app/webroot/img/

Ví dụ:

```
<?php echo $html->image('cake_logo.png', array('alt' => 'CakePHP'))?>
//Dữ liệu trả về html
<img src="/img/cake_logo.png" alt="CakePHP" />
```

$html->link(string $title, mixed $url = null, array $htmlAttributes = array(), string $confirmMessage = false,boolean $escapeTitle = true);

Tạo ra một thẻ html link. Các tham số truyền vào là
$title: Nhãn của đường link.
$url: Đường dẫn của link liên kết.
$htmlAttributes: các thuộc tính html của thẻ link.
$confirmMessage: Thông báo khi kích vào link. Mặc định giá trị là false.

Ví dụ:

```
<?php echo $html-
>link('Enter', '/pages/home', array('class'=>'button')); ?>
//Dữ liệu trả về HTML
<a href="/pages/home" class="button">Enter</a>
<?php echo $html-
>link('Delete', array('controller'=>'recipes', 'action'=>'delete', 6), arra
y(), "Are you sure you wish to delete this recipe?");?>
//Dữ liệu trả về html
<a href="/recipes/delete/6" onclick="return confirm('Are you sure you wi
sh to delete this recipe?');">Delete</a>
//Chuỗi query cũng có thể tạo ra từ một thẻ link()
<?php echo $html-
>link('View image', array('controller' => 'images', 'action' => 'view', 1,
'?' => array( 'height' => 400, 'width' => 500))); ?>
//Dữ liệu trả về html
```

```
   <a href="/images/view/1?height=400&width=500">View image</a>
```

$html->url($path);

Ví dụ:

```
   //Giả sử ta đang trong project test
   <?php echo $html->url('cake.jpg'); ?>
   //Dữ liệu trả về html
   '/test/cake.jpg'
   <?php echo $html->url('/file.txt'); ?>
   //Dữ liệu trả về html
   /test/file.txt
```

$form->create(string $model = null, array $options = array());

Tất cả các tham số có thể là tùy chọn. Nếu không có tham số truyền vào, thì mặc định phương thức là POST, và controller và action nào gọi ra view chứa form này thì khi submit thì form sẽ chuyển dữ liệu về controller và action đấy.

Ví dụ:

```
   //ta tạo ra một form trong action có tên là add
   <?php echo $form->create('Recipe'); ?>
   //Dữ liệu trả về html:
   <form id="RecipeAddForm" method="post" action="/recipes/add">
```

Các tham số truyền vào:
$model: tên model dùng để khởi tạo form đấy.
$option: Các thuộc tính khai báo thêm cho form đấy bao gồm
$options['type']: Chỉ định các phương thức truyền dữ liệu. Giá trị hợp lệ bao gồm 'post', 'get', 'file', 'put' và 'delete'.

Ví dụ:

```
   <?php echo $form->create('User', array('type' => 'get')); ?>
   //Dữ liệu trả về html
   <form id="UserAddForm" method="get" action="/users/add">
   <?php echo $form->create('User', array('type' => 'file')); ?>
   //Dữ liệu trả về html
   <form id="UserAddForm" enctype="multipart/form-
data" method="post" action="/users/add">
```

options['url']: Chỉ định đường dẫn khi form truyền dữ liệu. Giá trị url có thể truyền vào một chuỗi hay một mảng.

Ví dụ:

```
   <?php echo $form->create(null, array('url' => '/recipes/add')); ?>
   // hoặc
   <?php echo $form-
>create(null, array('url' => array('controller' => 'recipes', 'action' => '
add'))); ?>
   //Dữ liệu trả về html:
   <form method="post" action="/recipes/add">
```

```
   <?php echo $form-
>create(null, array('url' => 'http://www.google.com/search', 'type' => 'get
')); ?>
   //Dữ liệu trả về html:
   <form method="get" action="http://www.google.com/search">
```

options['action']: Cho phép chỉ định một action bất kỳ trong controller hiện tại. Giả sử bạn muốn truyền dữ liệu tới form login trong controller hiện tại, bạn có thể khai báo như sau:

```
   <?php echo $form->create('User', array('action' => 'login')); ?>
   //Dữ liệu trả về html:
   <form id="UserLoginForm" method="post" action="/users/login">
   </form>
```

$form->end('submitName');

Tạo ra một thẻ form đóng trong trường hợp đã sử dụng $form->create.

Ví dụ:

```
   <?php echo $form->create(); ?>
   <!-- Form elements go here -->
   <?php echo $form->end(); ?>
   hoặc:
   <?php echo $form->end('Finish'); ?>
   //Dữ liệu trả về html
   <div class="submit">
   <input type="submit" value="Finish" />
   </div>
   </form>
```

1.7.3. Controller

1.7.3.1. Giới thiệu

Một controller (điều khiển) được sử dụng để quản lí về mặt logic của một phần trong ứng dụng của bạn. Hầu hết, các controller được sử dụng để quản lí một model (mô hình) đơn giản. Ví dụ, nếu bạn xây dựng một trang web cho một hiệu bánh trực tuyến, bạn cần phải có RecipesController và IngredientsController để quản lý công thức và thành phần. Trong CakePHP, những controller được đặt tên như là của model nhưng ở dạng số nhiều.

Model Recipe được điều khiển bởi RecipesController, model Product được điều khiển bởi ProductsController v.v…

Những lớp controller trong ứng dụng của bạn được kế thừa từ lớp CakePHP AppController. Nó được kế thừa từ lớp nhân Controller, một phần của thư viện CakePHP. Lớp AppController được định nghĩa trong thư mục app/app_controller.php và nó chứa nhiều phương thức được chia sẻ giữa tất cả các controller trong ứng dụng của bạn.

Controller có thể thêm vào nhiều phương thức mà chúng thường chỉ dẫn đến các action (hành động). Action là một phương thức điều khiển sử dụng để hiển thị views (xem). Một action đơn giản chỉ là một phương thức của controller.

CakePHP là người vận chuyển để gọi các action khi chúng nằm trong URL yêu cầu đến action của controller (tham chiếu đến "Route Configuration" để biết cách thức gắn các tham số tên action của controller trong URL).

Quay lại với ví dụ hiệu bánh online của chúng ta, RecipesController có thể chứa các phương thức (hay action) view(), share(), và search(). Controller sẽ tìm thấy chúng trong /app/controllers/recipes_controller.php anh có nội dung:

```
<?php

# /app/controllers/recipes_controller.php

class RecipesController extends AppController {
    function view($id)     {
        //action logic goes here..
    }

    function share($customer_id, $recipe_id) {
        //action logic goes here..
    }

    function search($query) {
        //action logic goes here..
    }
}

?>
```

1.7.3.2. App Controller

Như đã giới thiệu, lớp AppController là lớp cha của tất cả các controller trong ứng dụng của bạn. Chính lớp AppController cũng được kết thừa từ lớp Controller chứa trong thư viện nhân của CakePhp. AppController được định nghĩa trong /app/app_controller.php như sau:

```
<?php
class AppController extends Controller {
}
?>
```

Các thuộc tính và phương thức được tạo trong AppController sẽ có giá trị trên tất cả các controller trong ứng dụng của bạn. Đó là ý tưởng để tạo ra code (mã chương trình), cái mà được sử dụng chung trong tất cả các controller. Components (sẽ đề cặp đến sau) là mã chương trình tốt nhất được sử dụng ở nhiều controller (không nhất thiết phải là tất cả).

Thông thường các luật kế thừa từ các đối tượng được áp dụng, CakePHP cũng tạo ra một dãy phụ để làm việc khi chúng mang theo những thuộc tính đặc biệt, giống như danh sách các component hoặc helper được controller sử dụng. Trong trường hợp này, mảng giá trị AppController được nối với các mảng của lớp con. CakePHP nối những biến trong AppController với các mảng trong controller của ứng dụng

```
$components
$helpers
$uses
```

Nhớ rằng để thêm các helper Html và Form cần định nghĩa $helpers trong AppController.

Cũng lưu ý rằng để gọi lại AppController trong controller con để đạt được kết quả tốt nhất:

```
function beforeFilter(){
   parent::beforeFilter();
}
```

1.7.3.3. Page Controller

Nhân CakePHP có chứa controller mặc định được gọi là Pages Controller (cake/libs/controller/pages_controller.php). Trong chính chủ bạn có thể thấy sau khi cài đặt CakePHP được phát ra nhờ sử dụng controller này. Nó thường sử dụng để tạo ra những trang tĩnh. Ví dụ nếu bạn tạo một tập tin view /views/pages/about_us.ctp, khi đó các bạn có thể truy cập http://l2qs.ttct/pages/about_us. Khi các bạn « bake » một ứng dụng sử dụng console (điều khiển dòng lệnh) của CakePHP thì pages controller sẽ được sao chép tự động vào thư mục app/controllers/ và nếu cần bạn có thể sửa đổi nó. Hoặc là bạn có thể sao chép từ nhân của ứng dụng.

1.7.3.4. Những thuộc tính Controller (attributes)

Xin giới thiệu một số thuộc tính cơ bản của lớp Controller, các bạn có thể tham khảo đầy đủ ở http://api.cakephp.org/class/controller.

- ***$name***

Người dùng PHP4 nên bắt đầu định nghĩa controller bằng cách sử dụng thuộc tính $name. Thuộc tính $name để đặt tên cho controller. Thông thường, tên chỉ là số nhiều của tên model được sử dụng.

```
<?php

#   $name controller attribute usage example

class RecipesController extends AppController {
   var $name = 'Recipes';
}

?>
```

- ***$components, $helpers và $uses***

Tiếp đến, một trong những thuộc tính được CakePHP sử dụng nhiều đó là helpers, components, và models. Bạn sẽ sử dụng chúng chung với controller hiện tại. Sử dụng những thuộc tính này là làm cho các lớp MVC đưa ra bởi $components và $uses có hiệu lực trong lớp controller và các biến trong lớp (ví dụ: $this->ModelName) và $helpers cho lớp view như một biến tham chiếu đối tượng ($helpername)

Mặc định, các controller truy xuất đến model chính của chúng. RecipesController sẽ có model Recipe và tồn tại $this->Recipe, và ProcductsController cũng vậy có model Product và $this->Product. Tuy nhiên, khi bạn cho phép controller truy xuất đến các model bổ trợ thì sử dụng biến $uses, tên của controller hiện hành cũng phải thêm vào. Xem ví dụ minh họa ở dưới.

Các helper Html, Form, và Session luôn luôn có. Nhưng nếu bạn chọn định nghĩa mảng $helpers trong AppController, thì phải thêm Html và Form nết bạn muốn nó có hiệu lực trong controller. Chúng ta sẽ bàn kĩ vấn đề này ở phần tiếp theo.

Ví dụ về controller của CakePHP thêm các lớp MVC.

```
<?php
class RecipesController extends AppController {
    var $name = 'Recipes';

    var $uses = array('Recipe', 'User');
    var $helpers = array('Ajax');
    var $components = array('Email');
}
?>
```

Mỗi biến trên được sát nhập với các giá trị được kết thừa, bởi vậy đôi lúc nó không cần thiết ví dụ khai báo lại helper Form hoặc những thứ đã khai báo trong controller App của bạn,

Nếu bạn không muốn sử dụng Model trong contrller của bạn thì gán $uses = array(). Chúng sẽ cho phép bạn sử dụng một controller mà không cần có model đúng tên.

- ***$layout và $page***

Trong các controller của CakePHP có một số thuộc tính để điều khiển cách bố trí (layout) trang web.

Thuộc tính $layout có thể được gán bằng tên của một layout được lưu trữ trong /app/views/layouts. Bạn có thể chỉ rõ một layout bằng cách gán bằng tên layout bỏ đi .ctp. Nếu bạn không định nghĩa, thì mặc định nó sẽ lấy /app/views/layouts/default.ctp vốn có sẵn trong CakePHP.

```
<?php

//   Using $layout to define an alternate layout

class RecipesController extends AppController {
    function quickSave() {
        $this->layout = 'ajax';
    }
}

?>
```

Bạn cũng có thể thay đổi tựa đề (title) của trang web (cái dòng chữ nằm góc phải trên của trình duyệt) bằng cách sử dụng $pageTite. Và đừng quên thêm biến $title_for_layout, ít nhất là nằm giữa thẻ <title> và thẻ </title> ở đầu của đoạn mã HTML.

```
<?php
```

```
//   Using $pageTitle to define the page title

class RecipesController extends AppController {
    function quickSave() {
        $this->pageTitle = 'My search engine optimized title';
    }
}

?>
```

Bạn cũng có thể thiết đặt tự đề của trang bằng cách sử dụng $this->pageTitle (phải thêm $this->part). Bạn nên làm như vậy để việc phân cách trang web một cách logic từ layout và nội dung. Với những web tĩnh bạn nên sử dụng $this->pageTitle trong view nếu bạn muốn có title khác.

Nếu không được thiết đặt thì CakePHP sẽ tự động tạo ra từ tên controller và tên của view.

- ***$params***

Các tham số controller sẽ nằm trong $this->params trong trang controller của CakePHP. Thuộc tính này sẽ cung cấp truy cập thông tin của yêu cầu hiện tại. Hầu hết việc sử dụng $this->params là để truy cập thông tin từ controller thông qua POST và GET.

$this->params['form']: lưu trữ dữ liệu POST từ form được lưu trữ. Thông tin kèm theo cũng được chứa trong $_FILES.
$this->params['admin']: được đặt bằng 1 nếu hoạt động hiện tại thông qua admin routing.
$this->params['bare']: lưu trữ 1 nếu layout hiện tại trống, 0 nếu ngược lại.
$this->params['isAjax']: lưu trữ 1 nếu Ajax được gọi. Biến này được thiết đặt khi RequestHandler Component được sữ dụng trong controller.
$this->params['controller']: lưu trữ tên controller hiện tại trong yêu cầu (request). Ví dụ nếu URL /posts/view/1 được yêu cầu thì $this->params['controller'] là "posts".
$this->params['action']: lưu trữ tên của action hiện tại chứa trong yêu cầu. Ví dụ nếu yêu cầu URL /posts/view/1 was requested thì $this->params['action'] chứa "view".
$this->params['pass']: trả về một mảng (đánh chỉ số bằng số)các tham số sau action.

```
// URL: /posts/view/12/print/narrow

Array
(
    [0] => 12
    [1] => print
    [2] => narrow
)
```

$this->params['url']: lưu trữ URL hiện tại, chứa cặp khóa-giá trị. Ví dụ URL /posts/view/?var1=3&var2=4 được gọi, $this->params['url'] sẽ chứa:

```
[url] => Array
(
    [url] => posts/view
```

```
        [var1] => 3
        [var2] => 4
    )
```

$this->data: dùng để mang dữ liệu từ FormHelper forms đến controller.

```
    // The FormHelper is used to create a form element:
    $form->text('User.first_name');
```

Được dịch ở ra ở view:

```
    <input name="data[User][first_name]" value="" type="text" />
```

Khi submit dữ liệu sẽ được chứa trong this->data

```
    //The submitted first name can be found here:
    $this->data['User']['first_name'];
```

$this->params['prefix']: thiếu đặt tiền tố của routing. Ví dụ giá trị là "admin" với các yêu cầu /admin/posts/someaction.

$this->params['named']: lưu trữ các tham số trong URL ở dạng /key:value/. Ví dụ URL/posts/view/var1:3/var2:4 thì $this->params['named'] là mảng như sau:

```
    [named] => Array
    (
        [var1] => 3
        [var2] => 4
    )
```

- ***persistModel***

Sử dụng để tạo những cạc nhớ (cache) những thể hiện của model của controller sử dụng. Khi đặt là true, tất cả các model có liên quan đến controller sẽ được lưu trữ. Nó tăng thêm hiệu quả trong nhiều trường hợp.

Còn rất nhiều thuộc tính khác.

1.7.3.5. Những phương thức Controller (methods)

- ***Tương tác với View***

set(string $var, mixed $value) : đây là phương thức chính để có thể truyền giá trị biến từ Controller ra View.

```
    <?php
    //First you pass data from the controller:
    $this->set('color', 'pink');
    //Then, in the view, you can utilize the data:
    ?>

    You have selected <?php echo $color; ?> icing for the cake.
```

Phương thức set() cũng cho phép chúng ta truyền mảng ra view. Điều này thuận tiện khi đưa ra một tổ hợp thông tin.

render(string $action, string $layout, string $file) : ở mỗi lần yêu cầu một action của controller, sau khi thực hiện xong các thao tác, thì phương thức này được gọi một cách tự động. Phương thức này thực hiện ở tất cả các view (sử dụng dữ liệu được đưa ra trong hàm set()), thay chỗ các view vào trong các layout và thể hiện ra ở giao diện người dùng.

Theo quy ước, các tập tin xem mặc định sẽ được đưa ra. Nếu action search() của RecipesController được yêu cầu, thì tập tin view /app/views/recipes/search.ctp sẽ được đưa ra.

```
class RecipesController extends AppController {
...
    function search() {
        // Render the view in /views/recipes/search.ctp
        $this->render();
    }
...
}
```

Mặc dù CakePHP tự động gọi phương thức render() (trừ khi thiết đặt $this->autoRender là false) thì sau mỗi trình tự hoạt động, bạn có thể sử dụng nó để chỉ định một tập tin khác bằng cách chỉ ra tên của action trong controller nào nhờ vào sử dụng tham biến $action. Nếu $action bắt đầu bằng '/' nó sẽ lấy một view hoặc element trong thư mục /app/views.

```
// Render the element in /views/elements/ajaxreturn.ctp
$this->render('/elements/ajaxreturn');
```

Bạn cũng có thể chỉ định tập tin view hoặc element nhờ vào tham số $file. Khi sử dụng $file, chú ý cài đặt một vài hằng số của CakePHP.

Tham số $layout cho phép bạn chỉ được layout cho view được đưa ra.

- ***Flow Control***

redirect(string $url, integer $status, boolean $exit): là phương thức thường được sử dụng nhất. Phương thức chứa tham biến đầu tiên là định dạng URL của CakePHP. Phương thức để chuyển hướng view sang một view khác.

```
function placeOrder() {
    //Logic for finalizing order goes here
    if($success) {
        $this->redirect(array('controller' => 'orders', 'action' => 'thanks'));
    } else {
        $this->redirect(array('controller' => 'orders', 'action' => 'confirm'));
    }
}
```

Cũng có thể sử dụng URL bất kì.

```
$this->redirect('/orders/thanks'));
$this->redirect('http://www.example.com');
```

Cũng có thể đưa dữ liệu qua action khác.

```
$this->redirect(array('action' => 'edit', $id));
```

Tham số tứh hai cho phép bạn định nghĩa một mã trạng thái (status) của HTTP để điều hướng. Bạn có thể sử dụng 301 (chuyển mãi mãi), 303 (xem khác), hoặc chuyển hướng một cách tự nhiên.

Phương thức sẽ gặp exit() sau khi chuyển hướng, trừ khi tham số thứ 3 là false.

Bạn có thể điều hướng đến một trang liên kế:

```
$this->redirect($this->referer());
```

```
$this->redirect($this->referer());
```

Nếu sử dụng một tiền tố điều hướng và bạn muốn điều hướng bỏ qua tiền tố thì bạn thiết lập tham số thứ ba là null, false sẽ không hiệu quả. Ví dụ dưới điều hướng về trang admin, đã bỏ tiền tố.

```
$this->redirect(array('controller' => 'orders', 'action' => 'add', 'admin' => null));
```

flash(string $message, string $url, integer $pause): giống như direct() phương thức flash thường sử dụng để chuyển người dùng sang trang mới. Nó khác ở chổ nó sẽ hiện thông báo trước khi chuyển tới môt URL khác.

Tham số đầu tiên là thông điệp được thể hiện, thứ hai là đường dẫn theo chuẩn CakePHP, tham số thứ ba là số giây sẽ dừng trước khi chuyển hướng.

Có thể dùng phương thức setFlash() để thực hiện tạo thông điệp đó.

- ***Callbacks***

Các controller của CakePHP sẽ chặn và gọi trước hoặc sau khi action controller được nêu ra.

beforeFilter() : Hàm này sẽ thực thi trước khi mọi action của controller thực thi. This function is executed before every action in the controller. Đây là nơi thuận lợi để kiểm tra các hoạt động của người dùng.

beforeRender() : được gọi sau khi action được gọi nhưng trước khi view được nêu ra. Cái này không hay dùng như cần nếu bạn gọi render() được khi kết thúc một action.

afterFilter() : được gọi sau khi action controller và sau khi goi view hoàn thành. Đây là phương thức cuối cùng ở controller được gọi.

- ***Các phương thức hữu ích khác***

constructClasses : phương thức sẽ gọi các model được yêu cầu bởi controller. Quá trình gọi được thực hiện bằng chuẩn CakePHP, nhưng phương pháp này là tiện dụng để có khi truy cập vào bộ điều khiển từ góc độ khác nhau. Nếu bạn cần CakePHP trong một kịch bản dòng lệnh hoặc sử dụng từ bên ngoài, constructClasses () có thể có ích.

paginate : Cho phép bạn tạo ra một trang theo mô hình có sẵn, hỗ trợ đánh số trang…

requestAction(string $url, array $options) : hàm này gọi một action của controller từ các vị trí khác nhau và nhận kết quả từ action đó. $url theo định dạng URL của CakePHP. Để có thêm dữ liệu nhận từ action sử dụng mảng $options.