**Lời giới thiệu:**

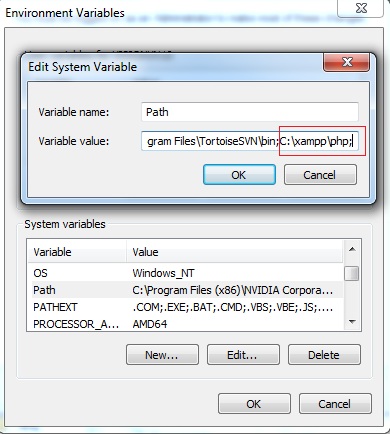
Bài viết này yêu cầu sử dụng FuelPHP 1.1.0 trở lên, nên nếu các bạn dùng Fuel 1.0.1 sẽ không phát huy hết tác dụng những cái mới nhất của FuelPHP cung cấp, nhưng về kiến thức cơ bản thì 1.1.0 vẫn không thay đổi gì nhiều, và bạn cũng có thể dễ dàng thực hiện chương trình này…

**Vấn đề chính:**

1. **Đầu tiên chúng ta sẽ tạo biến môi trường php trong windows (ở đây mình hướng dẫn trên win7)**

Click chuột phải vào My Computer → Propertise → Advanced system settings → Click Button “Enviroment Variables”…

Xem ở phần System variables chọn Path → edit thêm đường dẫn có file php.exe (Ví dụ như trong máy của mình là: C:\xampp\php\)

Xem hình bên dưới để hiểu chi tiết hơn  


1. **Tạo Project với Netbean**

Mở NetBean và tạo ứng dụng php bằng FuelPHP Framework (ở đây mình dùng Netbean 7.4)

Cài plugin FuelPHP cho NetBean:

* + 1. Trên thanh menu chọn Tools → Plugins
    2. Chọn tab Available Plugins → Search → gõ FuelPHP
    3. Đánh dấu PHP FuelPHP Framework → Click button Install
    4. Click Next → check Accept license → Install
    5. Finish → Close

Tại giao diện đầu tiên của NetBean → thanh menu chọn File → New Project (Ctrl+Shift+N)

Chọn ở phần Categories chọn PHP → PHP Application → Next

Project Name: Tên app của bạn (ở đây mình đặt là DemoFirstApp)

Sources Folder: Đường dẫn để chạy app của bạn nếu bạn nào cài xampp thì có đường dẫn tương tự sau “C:\xampp\htdocs\DemoFirstApp”

Chọn Version php (ở đây mình chọn 5.4)

Default Encoding: UTF-8 → Next.

Tới màn hình Run Configuration để mặc định chọn Next

Tới màn hình PHP Framework → FuelPHP PHP Web Framework (ở đây mình chọn v1.7.2)

Finish

1. **Config ứng dụng của bạn**

Tạo 1 database trong phpmyadmin (ở đây mình tạo tên là blog)

Để app chúng ta trỏ tới database mới tạo ta chỉ việc vô “fuel/app/config/development/db.php”

Chỉ việc thay đổi chỗ dbname=tendatabase (ở project này mình đổi thành dbname=blog, username => ‘root’, password => ”)

return array(

'default' => array(

'connection' => array(

'dsn' => 'mysql:host=localhost;dbname=blog',

'username' => 'root', (tùy thuộc use đang sử dụng)

'password' => '',

),

),

);

Tiếp đến kích hoạt gói auth và gói orm bằng cách vô file sau có đường dẫn này: “fuel/app/config/config.php” mở comment ra và thêm auth

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\* Always Load \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

'always\_load' => array(

'packages' => array(

'auth',

'orm',

),

);

Cũng trong file config này luôn chúng ta cần thay đổi chút về biến mảng “whitelisted\_classes” điều này cho phép chúng ta truyền đối tượng validation tới view

'whitelisted\_classes' => array(

'Fuel\\Core\\Response',

'Fuel\\Core\\View',

'Fuel\\Core\\ViewModel',

'Fuel\Core\Validation',

'Closure',

),

Thiết lập nhóm người dùng:

Gói auth được tích hợp sẵn trong FuelPHP, và mặc định chúng ta dùng bộ “SimpleAuth”, khi bạn có nhiều kinh nghiệm trong việc dùng FuelPHP thì bạn có thể tự tạo và tùy chỉnh bộ Auth sao cho thích hợp với những dự án của chúng ta .

Để kích hoạt SimpleAuth đơn giản bạn chỉ việc mở file này lên “fuel/packages/auth/config/simpleauth.php” và thiết lập như sau

return array(

'groups' => array(

-1 => array('name' => 'Banned', 'roles' => array('banned')),

0 => array('name' => 'Guests', 'roles' => array()),

1 => array('name' => 'Users', 'roles' => array('user')),

50 => array('name' => 'Moderators', 'roles' => array('user', 'moderator')),

100 => array('name' => 'Administrators', 'roles' => array('user', 'moderator', 'admin')),

),

);

1. **Tạo người dùng (Users)**

Để xây dựng 1 admin panel chúng ta cần phải tạo 1 bảng users có 1 user là administartor đầu tiên. Chúng ta có thể dùng phpMyAdmin hay Navicat, nhưng tốt nhất bạn để làm những việc trên bạn nên dùng phương thức Oil của FuelPHP cung cấp…

Tại sao lại phải tạo user bằng Oil ?

Vì khi tạo user bằng Oil thì password của user đó được mã hóa trước khi lưu vào table users vì vậy việc tạo user bằng phpMyAdmin hay Nav lúc đầu là không ổn và tốn thời gian

Bắt đầu tạo:

Mở cmd lên (bằng cách nhấn nút win+R rồi gõ cmd enter)

Trỏ tới thư mục project gõ “cd c:\xampp\htdocs\DemoFirstApp\” ->  rồi gõ “**php oil g model users username:varchar[50] password:string email:string group:int last\_login:string login\_hash:string profile\_fields:text**“.

Với đoạn code trên oil sẽ tự động sinh ra model “fuel\app\classes\model\user.php” và 1 file “fuel\app\migration\001\_create\_users.php”

Chỉ còn 1 bước đơn giản tiếp theo nữa thôi oil sẽ tự tạo cho bạn 1 table users trong phpmyadmin gõ tiếp: “**php oil refine migrate**” -> Enter -> Done

Sau đây là đoạn code để gõ trong cmd nha.

$ oil generate model users username:varchar[50] password:string group:int email:string last\_login:int login\_hash:string profile\_fields:text

Creating model: C:\xampp\htdocs\DemoFirstApp\fuel\app\classes\model\user.php

Creating migration: C:\xampp\htdocs\DemoFirstApp\fuel\app\migrations/001\_create\_users.php

$ oil refine migrate

Performed migrations for app:default:

001\_create\_users

Thế là ta đã có 1 table trong phpmyadmin giờ chỉ việc tạo 1 users Administrator trong table user, trong gói Auth có 1 phương thức tạo user cực ngon mà không cần phải viết code tạo user bằng cách dùng Oil Console

$ oil console

Fuel 1.7.2 - PHP 5.6.3 (cli) (Nov 12 2014 17:12:19) [WINNT]

>>> Auth::create\_user('admin', '123456789', 'info@fsharefilm.com', 100);

1

-- Ctrl + C để thoát

Giải thích một chút đoạn code trên nhé: trong gói Auth chúng ta có hàm create\_user và truyền các biến username, password, emal, group, như các bạn thấy trong file config simpleauh.php thì 100 là thuộc group administrator có quyền cao nhất. Sau đó server sẽ phản hồi trả về cho chúng ta giá trị 1 là đã tạo thành công user đó có id là 1

1. **Generation Code (Chỗ này hơi bị đã)**

Việc tự động sinh code là 1 cách tuyệt với và hiệu quả cho việc bắt đầu code

Nếu những ai mới làm quen về Framework php thì các bạn nên tìm hiểu  cách tạo dựng 1 controller, model, view, làm những chức năng cơ bản xem, thêm, xóa, sửa như thế nào trước bằng thủ công sau đó bạn có thể dụng bộ sinh code tự động của fuelPHP này lúc đó bạn sẽ cảm thấy đỡ nhàm chán hơn trong việc code đi code lại những chức năng cơ bản đó

Với bộ sinh code tự động cho nền admin quản trị thì fuelPHP có tích hợp sẵn css Bootstrap, template admin hoàn toàn tương thích với tất cả màn hình thiết bị, tự động sinh giao diện rất chuẩn thân thiện người dùng.

Bây giờ chúng ta sẽ thử tạo phần quản lý bài post trong admin bẳng dòng lệnh dưới đây:

c:\xampp\htdocs\DemoFirstApp>php oil g admin posts title:string slug:string summary:text body:text user\_id:int

Creating controller: C:\xampp\htdocs\DemoFirstApp\fuel\app\classes/controller/base.php

Creating controller: C:\xampp\htdocs\DemoFirstApp\fuel\app\classes/controller/admin.php

Creating views: C:\xampp\htdocs\DemoFirstApp\fuel\app\views/admin/template.php

Creating views: C:\xampp\htdocs\DemoFirstApp\fuel\app\views/admin/dashboard.php

Creating views: C:\xampp\htdocs\DemoFirstApp\fuel\app\views/admin/login.php

Creating migration: C:\xampp\htdocs\DemoFirstApp\fuel\app\migrations/002\_create\_posts.php

Creating model: C:\xampp\htdocs\DemoFirstApp\fuel\app\classes/model/post.php

Creating controller: C:\xampp\htdocs\DemoFirstApp\fuel\app\classes\controller\admin\posts.php

Creating view: C:\xampp\htdocs\DemoFirstApp\fuel\app\views\admin\posts\index.php

Creating view: C:\xampp\htdocs\DemoFirstApp\fuel\app\views\admin\posts\view.php

Creating view: C:\xampp\htdocs\DemoFirstApp\fuel\app\views\admin\posts\create.php

Creating view: C:\xampp\htdocs\DemoFirstApp\fuel\app\views\admin\posts\edit.php

Creating view: C:\xampp\htdocs\DemoFirstApp\fuel\app\views\admin\posts\\_form.php

Creating view: C:\xampp\htdocs\DemoFirstApp\fuel\app\views/template.php

Và các bạn đã thấy dòng lệnh trên đã tự động tạo cho mình những file cần thiết để tạo dựng 1 admin panel là:

* + fuel\app\classes/controller/base.php
  + fuel\app\classes/controller/admin.php
  + fuel\app\views/admin/template.php
  + fuel\app\views/admin/dashboard.php
  + fuel\app\views/admin/login.php

Sau đó là việc tạo các file cần thiết để tạo dựng thành 1 ứng dụng quản lý posts trong admin

* + fuel\app\classes\model\post.php
  + fuel\app\classes\controller\admin\posts.php
  + fuel\app\views\admin\posts\index.php
  + fuel\app\views\admin\posts\create.php
  + fuel\app\views\admin\posts\edit.php
  + fuel\app\views\admin\posts\\_form.php

Ngoài ra còn sinh ra 1 file migrate post.php để khi chúng ta dùng lệnh “**php oil refine migrate**”  chúng ta sẽ có được 1 table posts và thực hiện ngay lệnh này nha

c:\xampp\htdocs\DemoFirstApp>php oil refine migrate

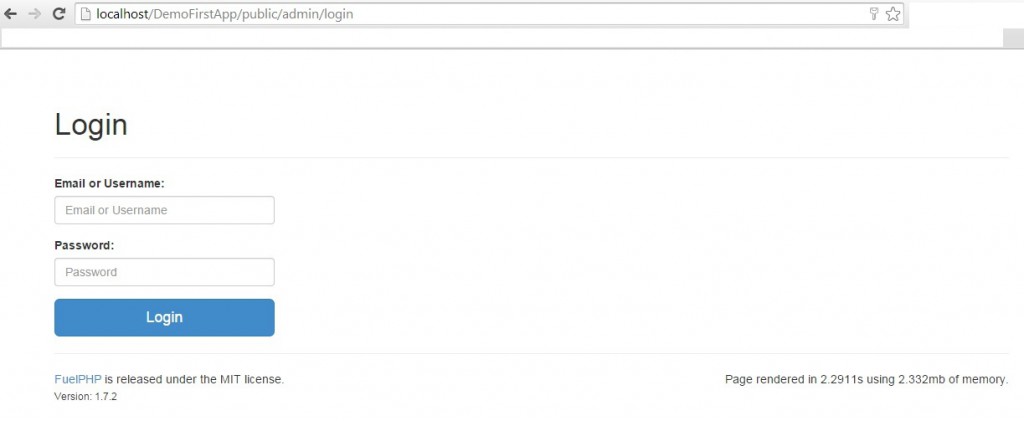
Performed migrations for app:default:

002\_create\_posts

Cuối cùng là oil sinh ra 1 file “fuel\app\views/template.php” và đó là template cho giao diện frontend

Tất cả đều tự động, rất nhanh, chính xác, và chuẩn không sợ sai sót nhỏ nào khi dùng lệnh để gen code, và nếu chúng ta muốn custom lại chỗ nào thì thì chỉ việc vào những file đó chỉnh sửa lại theo ý mình là chúng ta đã có được những gì hoàn hảo nhất

Đây là form login bắt đăng nhập khi chúng ta gõ đường dẫn sau: “localhost/DemoFirstApp/public/admin/posts”

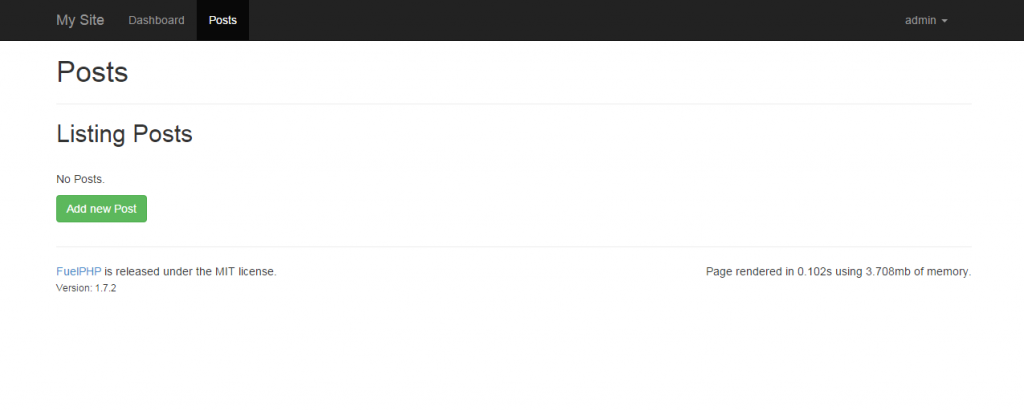
[](http://fsharefilm.com/wp-content/uploads/2014/12/form_login1.jpg)

form\_login\_fuelphp

* + Uh thì đăng nhập:
    1. username: admin
    2. password: 123456789
    3. Thông tin username và password là do bạn thêm user admin đầu tiên ở phía trên nha, còn ở trong bài viết này mình đặt như vậy.

Sau khi đăng nhập xong thì có màn hình Dashboard đầu tiên, muốn tới màn hình quản lý post thì nhấp vô phần menu Posts

* + Đây là màn hình quản lý  Posts

[](http://fsharefilm.com/wp-content/uploads/2014/12/Posts.png)

Posts

* + Tìm hiểu những controller được sinh ra

Chúng ta đã có được 1 controller base (Cotroller\_Base)  và mọi controller khác để phải được dẫn xuất từ đây, code bên trong controller này khá đơn giản

class Controller\_Base extends Controller\_Template

{

public function before(){

parent::before();

// Assign current\_user to the instance so controllers can use it

$this->current\_user = Auth::check() ? (Config::get('auth.driver', 'Simpleauth') == 'Ormauth ? Model\Auth\_User::find\_by\_username(Auth::get\_screen\_name()) : Model\_User::find\_by\_username(Auth::get\_screen\_name())): null;

// Set a global variable so views can use it

View::set\_global('current\_user', $this->current\_user);

}

}

Giải thích một chút về đoạn code trên, do được dẫn xuất từ Controller\_Template, nên tất cả các view sẽ được dùng bởi 1 template 1 cách tự động, khai báo thêm hàm before() trong controller base, tạo 1 biến toàn cục user khi user đó đã xác thực quyền truy cập (login thành công), để cho các view có thể thấy được thông tin user đang đăng nhập.

Một controller khác được tạo ra được gọi là Controller\_Admin được dẫn xuất từ Controller\_Base, thì trong controller này có hàm before() chạy trước sẽ xác thực quyền truy cập nếu thuộc group admin thì sẽ đăng nhập thành công để quản lý, còn nếu không thì sẽ ra thông báo lỗi không truy cập được admin panel và quay trở lại ngoài frontend. Nếu user không đăng nhập hoặc đăng nhập sai sẽ bị ra ngoài trang login

|  |
| --- |
| <?php  **class** Controller\_Admin **extends** Controller\_Base  {  **public** $template = 'admin/template';  **public function** before()  {  **parent**::*before*();  **if** (Request::*active*()->controller !== 'Controller\_Admin' or ! in\_array(Request::*active*()->action, **array**('login', 'logout')))  {  **if** (Auth::*check*())  {  $admin\_group\_id = Config::*get*('auth.driver', 'Simpleauth') == 'Ormauth' ? 6 : 100;  **if** ( ! Auth::*member*($admin\_group\_id))  {  Session::*set\_flash*('error', e('You don\'t have access to the admin panel'));  Response::*redirect*('/');  }  }  **else**  {  Response::*redirect*('admin/login');  }  }  }  **public function** action\_login()  {  // Already logged in  Auth::*check*() and Response::*redirect*('admin');  $val = Validation::*forge*();  **if** (Input::*method*() == 'POST')  {  $val->add('email', 'Email or Username')  ->add\_rule('required');  $val->add('password', 'Password')  ->add\_rule('required');  **if** ($val->run())  {  $auth = Auth::*instance*();  // check the credentials. This assumes that you have the previous table created  **if** (Auth::*check*() or $auth->login(Input::*post*('email'), Input::*post*('password')))  {  // credentials ok, go right in  **if** (Config::*get*('auth.driver', 'Simpleauth') == 'Ormauth')  {  $current\_user = Model\Auth\_User::*find\_by\_username*(Auth::*get\_screen\_name*());  }  **else**  {  $current\_user = Model\_User::*find\_by\_username*(Auth::*get\_screen\_name*());  }  Session::*set\_flash*('success', e('Welcome, '.$current\_user->username));  Response::*redirect*('admin');  }  **else**  {  $this->template->set\_global('login\_error', 'Fail');  }  }  }  $this->template->title = 'Login';  $this->template->content = View::*forge*('admin/login', **array**('val' => $val), **false**);  }  /\*\*  \* The logout action.  \*  \* **@access** public  \* **@return** void  \*/  **public function** action\_logout()  {  Auth::*logout*();  Response::*redirect*('admin');  }  /\*\*  \* The index action.  \*  \* **@access** public  \* **@return** void  \*/  **public function** action\_index()  {  $this->template->title = 'Dashboard';  $this->template->content = View::*forge*('admin/dashboard');  }  }  /\* End of file admin.php \*/ |

1. Tùy chỉnh Form tạo mới posts (Form Create)

Do bộ máy sinh code tự động thì có thể làm cho form của mình không được như ý muốn, ví dụ như form tạo mới 1 post thì ở phần user\_id lại hiện là textbox  để gõ, không phải là select để chọn user nào đang tồn tại hoặc người đang đăng nhập đăng post mới nhất, để giải quyết vấn đề này mình làm như sau:

Và bây giờ chúng ta sẽ bắt đầu vào việc tùy chỉnh phương thức action\_create() trong “file fuel/app/classes/controller/admin/posts.php” làm theo đoạn code phía dưới

|  |
| --- |
| **public function** action\_create()  {  **if** (Input::*method*() == 'POST')  {  $val = Model\_Post::*validate*('create');  **if** ($val->run())  {  $post = Model\_Post::*forge*(**array**(  'title' => Input::*post*('title'),  'slug' => Input::*post*('slug'),  'summary' => Input::*post*('summary'),  'body' => Input::*post*('body'),  'user\_id' => Input::*post*('user\_id'),  ));  **if** ($post and $post->save())  {  Session::*set\_flash*('success', e('Added post #'.$post->id.'.'));  Response::*redirect*('admin/posts');  }  **else**  {  Session::*set\_flash*('error', e('Could not save post.'));  }  }  **else**  {  Session::*set\_flash*('error', $val->error());  }  }  $this->template->title = "Posts";  $this->template->content = View::*forge*('admin\posts/create');  } |

Đoạn code trên có 3 sự thay đổi đó là

* + - Tạo đối tượng view

$view =  View::forge('admin/posts/create');

* + - Set biến toàn cục users có dạng mảng key => value

$view->set\_global('users', Arr::assoc\_to\_keyval(Model\_User::find('all'), 'id', 'username'));

* + - Set lại biến content

$this->template->content = $view;

Tương tự ta làm tiếp hàm action\_edit()

|  |
| --- |
| **public function** action\_edit($id = **null**)  {  $post = Model\_Post::*find*($id);  $val = Model\_Post::*validate*('edit');  **if** ($val->run())  {  $post->title = Input::*post*('title');  $post->slug = Input::*post*('slug');  $post->summary = Input::*post*('summary');  $post->body = Input::*post*('body');  $post->user\_id = Input::*post*('user\_id');  **if** ($post->save())  {  Session::*set\_flash*('success', e('Updated post #' . $id));  Response::*redirect*('admin/posts');  }  **else**  {  Session::*set\_flash*('error', e('Could not update post #' . $id));  }  }  **else**  {  **if** (Input::*method*() == 'POST')  {  $post->title = $val->validated('title');  $post->slug = $val->validated('slug');  $post->summary = $val->validated('summary');  $post->body = $val->validated('body');  $post->user\_id = $val->validated('user\_id');  Session::*set\_flash*('error', $val->error());  }  $this->template->set\_global('post', $post, **false**);  }  $this->template->title = "Posts";  $this->template->content = View::*forge*('admin\posts/edit');  } |

Bây giờ chúng ta sẽ chỉnh sửa lại html trong “fuel/app/view/admin/posts/\_form.php” vì view create và edit đều được render bởi \_form.php

<div class="form-group">

<?php echo Form::label('User id', 'user\_id', array('class'=>'control-label')); ?>

<?php echo Form::select('user\_id', Input::post('user\_id', isset($post) ? $post->user\_id : $current\_user->id), $users, array('class' => 'col-md-4 form-control')); ?>

</div>

* + The FrontEnd (Tạo Blog Frontend bằng tay)

Tạo mới 1 controller “fuel/app/classes/controller/blog.php”

|  |
| --- |
| class Controller\_Blog extends Controller\_Base{  public function action\_index(){  $view = View::forge('blog/index');  $view->posts = Model\_Post::find('all');  $this->template->title = 'My Blog';  $this->template->content = $view;  }  } |

Đồng thời tạo file view “fuel/app/view/blog/index.php” để khi ta vào action\_index() của controller blog sẽ có view load lên

<h2>Recent Posts</h2>

<?php foreach ($posts as $p): ?>

<h3><?php echo Html::anchor('blog/view/'.$p->slug, $p->title); ?></h3>

<p><?php echo $p->summary; ?></p>

<?php endforeach; ?>

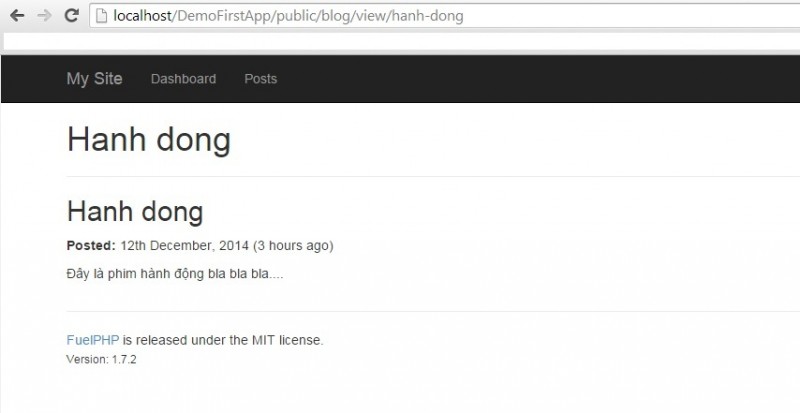
Tiếp đến ta tạo 1 action view trong controller blog: action\_view()

|  |
| --- |
| public function action\_view($slug){  $post = Model\_Post::find\_by\_slug($slug);  $this->template->title = $post->title;  $this->template->content = View::forge('blog/view', array('post'=>$post));  } |

Cuối cùng là ta tạo “fuel/app/view/blog/view.php”

|  |
| --- |
| <h2><?php echo $post->title ?></h2>  <p>  <strong>Posted: </strong>  <?php  echo date('nS F, Y', $post->created\_at)  ?> (  <?php  echo Date::time\_ago($post->created\_at)  ?>)  </p>  <p><?php echo nl2br($post->body) ?></p> |

Hàm find\_by\_slug khi được chạy sẽ cho câu sql “WHERE slug = ‘foo’ ” và sẽ trả về single Model\_Post với dữ liệu đó, bạn có thể thay thế từ slug thành bất kỳ field nào tồn tại trong table post.

[](http://fsharefilm.com/wp-content/uploads/2014/12/view_blog1.jpg)

1. Sử dụng ORM

Việc dùng ORM trong fulePHP vô cùng đơn giản và hiệu quả, dùng để tạo mối quan hệ giữa các bảng với nhau, ví dụ:

* + - 1 bài post thì có 1 user ta sẽ khai báo trong model post như sau
    - protected static $\_belongs\_to = array('user');
    - 1 User thì có nhiều bài post ta khai báo trong model user như sau
    - protected static $\_has\_may = array('post');

Để kiểm tra nhanh coi việc tạo mối quan hệ với các bảng có hoạt động hay không ta dùng oil console kiểm tra nhé

#c:\xampp\htdocs\DemoFirstApp>php oil console

Fuel 1.7.2 - PHP 5.6.3 (cli) (Nov 12 2014 17:12:19) [WINNT]

>>> $post = Model\_Post::find('first');

>>> $post->user->username

admin

>>> Nhấn Ctrl+C để thoát

Ngon rồi đó đã ra admin là người post bài đầu tiên, vậy việc tạo mối quan hệ với các bảng không có gì là khó khăn trong fuelphp để tìm hiểu kỹ hơn về Relating Model bạn có thể vô link này xem thêm phần Fully configured: <http://fuelphp.com/docs/packages/orm/relations/intro.html>

Tiếp đến chúng ta sẽ cùng nhau sửa lại file “fuel/app/views/blog/view.php” một chút mục đích là lấy username của user post bài đó, vì chúng ta đã có đối tượng $post->user

<h2><?php echo $post->title ?></h2>

<p>

<strong>Posted: </strong><?php echo date('nS F, Y', $post->created\_at) ?> (<?php echo Date::time\_ago($post->created\_at)?>)

by <?php echo $post->user->username ?>

</p>

<p><?php echo nl2br($post->body) ?></p>

Vậy là xong phần blog nếu thích làm thêm phần comment cho blog nữa bạn có thể kéo xuống coi để làm tiếp nha.

1. Thêm bình luận (Adding Comments)

Ta sẽ bắt đầu gen code admin cho comments

#c:\xampp\htdocs\DemoFirstApp>php oil g admin comments name:string email:string website:string message:text post\_id:int -s

Creating migration: C:\xampp\htdocs\DemoFirstApp\fuel\app\migrations/003\_create\_comments.php

Creating model: C:\xampp\htdocs\DemoFirstApp\fuel\app\classes/model/comment.php

Creating controller: C:\xampp\htdocs\DemoFirstApp\fuel\app\classes\controller\admin\comments.php

Creating view: C:\xampp\htdocs\DemoFirstApp\fuel\app\views\admin\comments\index.php

Creating view: C:\xampp\htdocs\DemoFirstApp\fuel\app\views\admin\comments\view.php

Creating view: C:\xampp\htdocs\DemoFirstApp\fuel\app\views\admin\comments\create.php

Creating view: C:\xampp\htdocs\DemoFirstApp\fuel\app\views\admin\comments\edit.php

Creating view: C:\xampp\htdocs\DemoFirstApp\fuel\app\views\admin\comments\\_form.php

Cuối cùng đừng quên chạy lệnh này nữa mới hoàn tất

c:\xampp\htdocs\DemoFirstApp>php oil refine migrate

Performed migrations for app:default:

003\_create\_comments

Ta sẽ tạo mối quan hệ với các bảng khi có phần comment

* + - Model\_User:

protected static $\_has\_many = array('posts', 'comments');

protected static $\_belongs\_to = array('user');

* + - Model\_Post:

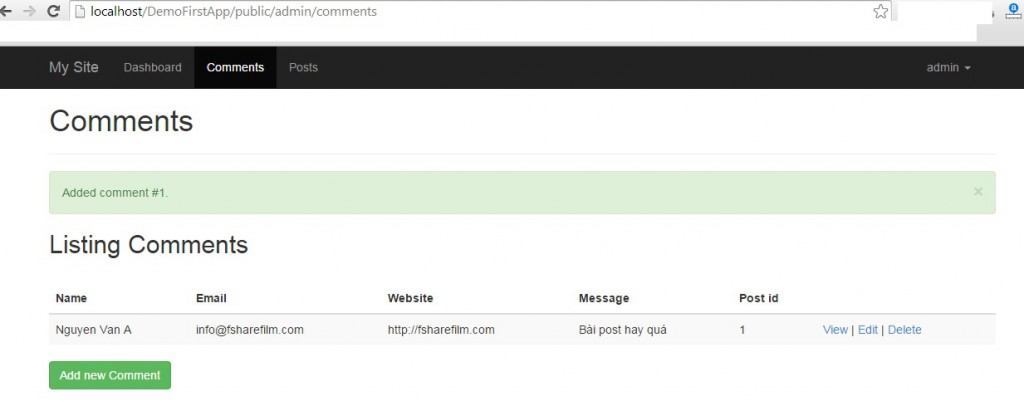
protected static $\_belongs\_to = array('user');

protected static $\_has\_many = array('comments');

* + - Model\_Comment:

protected static $\_belongs\_to = array('post', 'user');

Và chúng ta add 1 comment mới thông qua giao diện admin vô đường dẫn sau: “http://localhost/DemoFirstApp/public/admin/comments/create”

[](http://fsharefilm.com/wp-content/uploads/2014/12/comments.jpg)

comments

Bây giờ chúng ta thử kiểm mối quan hệ với các bảng bằng oil console

c:\xampp\htdocs\DemoFirstApp>php oil console

Fuel 1.7.2 - PHP 5.6.3 (cli) (Nov 12 2014 17:12:19) [WINNT]

>>> Model\_Post::find('first')->comments

array (

1 =>

Model\_Comment::\_\_set\_state(array(

'\_is\_new' => false,

'\_frozen' => false,

'\_sanitization\_enabled' => false,

'\_data' =>

array (

'id' => '1',

'name' => 'Nguyen Van A',

'email' => 'info@fsharefilm.com',

'website' => 'http://fsharefilm.com',

'message' => 'B├ái post hay qu├í',

'post\_id' => '1',

'created\_at' => '1417771920',

'updated\_at' => '1417771920',

),

'\_custom\_data' =>

array (

),

'\_original' =>

array (

'id' => '1',

'name' => 'Nguyen Van A',

'email' => 'info@fsharefilm.com',

'website' => 'http://fsharefilm.com',

'message' => 'B├ái post hay qu├í',

'post\_id' => '1',

'created\_at' => '1417771920',

'updated\_at' => '1417771920',

),

'\_data\_relations' =>

array (

),

'\_original\_relations' =>

array (

),

'\_reset\_relations' =>

array (

),

'\_disabled\_events' =>

array (

),

'\_view' => NULL,

'\_iterable' =>

array (

),

)),

)

>>>

Bạn thấy màn hình console mình đã xuất ra 1 đối tượng, điều đó có nghĩa là thiết lập mối quan hệ với các bảng thành công.

Bây giờ bạn chỉ cần copy toàn bộ nội dung code bên dưới và paste vào “fuel/app/views/blogs/view.php” để hiển thị comment và nhập comment

<h2><?php echo $post->title ?></h2>

<p>

<strong>Posted: </strong><?php echo date('nS F, Y', $post->created\_at) ?> (<?php echo Date::time\_ago($post->created\_at)?>)

by <?php echo $post->user->username ?>

</p>

<p><?php echo nl2br($post->body) ?></p>

<hr />

<h3 id="comments">Comments</h3>

<?php foreach($post->comments as $comment): ?>

<p><?php echo Html::anchor($comment->website, $comment->name) ?> said "<?php echo $comment->message; ?>" </p>

<?php endforeach; ?>

<h3>Write Comment</h3>

<fieldset>

<?php echo Form::open(array("class"=>"form-horizontal", 'action' => 'blog/comment/'.$post->slug));?>

<div class="form-group">

<?php echo Form::label('Name', 'name', array('class'=>'col-sm-2 control-label')); ?>

<div class="col-sm-10"><?php echo Form::input('name', '', array('class' => 'form-control')) ?></div>

</div>

<div class="form-group">

<?php echo Form::label('Website', 'website', array('class'=>'col-sm-2 control-label')); ?>

<div class="col-sm-10"><?php echo Form::input('website', '', array('class' => 'form-control')) ?></div>

</div>

<div class="form-group">

<?php echo Form::label('Email', 'email', array('class'=>'col-sm-2 control-label')); ?>

<div class="col-sm-10"><?php echo Form::input('email', '', array('class' => 'form-control')) ?></div>

</div>

<div class="form-group">

<?php echo Form::label('Comment', 'comment', array('class'=>'col-sm-2 control-label')); ?>

<div class="col-sm-10"><?php echo Form::textarea('message', '', array('class' => 'form-control')) ?></div>

</div>

<div class="form-group">

<div class="col-sm-offset-2 col-sm-10"><?php echo Form::submit('submit', 'Comment', array('class' => 'btn btn-primary')) ?></div>

</div>

<?php echo Form::close(); ?>

</fieldset>

Đây là giao diện sau khi chỉnh xong nha chuẩn giao diện Twitter Bootstrap

Nhưng chưa hoàn tất việc có thể nhập comment chúng ta cần tạo action\_comment($slug) thêm đoạn code sau  vào blog controller:

public function action\_comment($slug){

$post = Model\_Post::find\_by\_slug($slug);

//Lazy validation

if(Input::post('name') AND Input::post('email') AND Input::post('message')){

//Create a new comment

$post->comments[] = new Model\_Comment(array(

'name' => Input::post('name'),

'website' => Input::post('website'),

'email' => Input::post('email'),

'message' => Input::post('message'),

'user\_id' => $this->current\_user->id

));

//Save the post and the comment will too

if($post->save()){

$comment = end($post->comments);

Session::set\_flash('success', 'Add comment #'.$comment->id);

}else{

Session::set\_flash('error', 'Can not comment');

}

Response::redirect('blog/view/'.$slug);

}else{

$this->action\_view($slug);

}

}