Laravel 大型项目系列教程（一）

一、课程概述

1.课程介绍

本教程将使用 Laravel 完成一个多用户的博客系统，大概会包含如下内容：

- 路由管理。
- 用户管理，如用户注册、修改信息、锁定用户等。
- 文章管理，如发表文章、修改文章等。
- 标签管理，文章会有一到多个标签。
- 数据库管理，如迁移、填充数据等。
- Web 表单验证。
- Blade 模版引擎。
- 分页处理。
- 安全处理。
- 单元测试。
- 部署到应用服务器 Apache。

尽量保证每节教程完整并能运行，会在教程的最后附上这节教程的代码下载地址。

Tip：教程中必要的知识点都会有一个超链接

二、环境要求

- PHP 5.4+

- MySQL 5.1+

- Composer（[中国镜像](http://pkg.phpcomposer.com/)）

三、Let's go!

1.新建一个 Laravel 项目

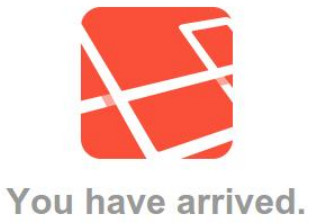
使用如下命令创建一个名为 blog 的 Laravel 项目：

```
$ composer create-project laravel/laravel blog --prefer-dist
```

创建完成之后进入到 blog 目录，修改 app/config/app.php 中的 timezone 为 RPC、locale 为 zh，然后在 blog 目录下启动它自带的开发服务器：

```
$ php artisan serve
Laravel development server started on http://localhost:8000
```

打开浏览器输入 localhost:8000，如果页面如下图就说明项目搭建完成了：



2.安装插件

在 composer.json 中增加：

```
"require-dev": {
    "way/generators": "~2.0"
},
```

运行 composer update 安装，完成后在 app/config/app.php 的 providers 中增加：

```
'Way\Generators\GeneratorsServiceProvider'
```

运行 php artisan 是不是多了 generate 选项，它可以快速地帮我们创建想要的组件。

3.建立数据库

把 app/config/database.php 中 connections 下的 mysql 改成你自己的配置：

```
'mysql' => array(
    'driver'    => 'mysql',
    'host'      => 'localhost',
    'database'  => 'blog',
    'username'  => 'root',
    'password'  => '',
    'charset'   => 'utf8',
    'collation' => 'utf8_unicode_ci',
    'prefix'    => '',
),
```

需要在 MySQL 中先创建一个名为 blog 的数据库

配置完成之后，创建 users 表的数据库迁移文件：

```
$ php artisan migrate:make create_users_table --create=users
```

我们会发现在 app\database\migrations 下多了一个*_create_users_table.php 文件，在这个文件中修改：

```
Schema::create('users', function(Blueprint $table){
    $table->increments('id');
    $table->string('email');
    $table->string('password');
    $table->string('nickname');
    $table->boolean('is_admin')->default(0);
    $table->boolean('block')->default(0);
    $table->timestamps();
});
```

之后进行数据库迁移：

```
$ php artisan migrate
```

你会惊讶地发现在数据库中多了两张表 users 和 migrations，users 表就是我们定义的表，migrations 表记录了迁移的信息。

4.创建 User 模型

数据库迁移完成之后我们将使用 Eloquent ORM ,这是 Laravel 让人着迷的重要原因之一。我们会发现在 app\models 下已经有一个 User.php 文件了，对其修改：

```
use Illuminate\Auth\UserInterface;
use Illuminate\Auth\UserTrait;

class User extends Eloquent implements UserInterface {
    use UserTrait;

    protected $table = 'users';
    protected $hidden = array('password', 'remember_token');
    protected $guard = array('email', 'password');
}
```

5.填充数据

有了 User 模型后，我们就可以向数据库填充数据了，在 app/database/seeds 下创建一个名为 UsersSeeder.php 的文件，增加如下：

```
class UsersSeeder extends Seeder {
    public function run()
      {
        User::create([
            'email'    => 'admin@shiyanlou.com',
            'password' => Hash::make(''),
            'nickname' => 'admin',
            'is_admin' => 1,
        ]);
    }
}
```

然后在 DatabaseSeeder.php 中增加：

```
$this->call('UserTableSeeder');
```

之后就真正地向数据库填充数据：

```
$ php artisan db:seed
```

你可以查看数据库，会发现 users 表中多了一条记录。

详情可以查看 Laravel 中数据库的迁移和填充

6.创建视图模版

我们将使用 Laravel 中的 Blade 模版引擎，使用下面命令创建三个视图：

```
php artisan generate:view _layouts.default
php artisan generate:view _layouts.nav
php artisan generate:view _layouts.footer
php artisan generate:view index
```

之后你可以在 app/views 下发现多了一个 index.blade.php 和一个_layouts 文件夹，在_layouts 文件夹下有三个文件 default.blade.php、footer.blade.php 和 nav.blade.php。我们将使用 AmazeUI 框架来做为前端框架，修改 default.blade.php：

```
<!DOCTYPE html>
<html>
<head lang="zh">
    <meta charset="UTF-8"/>
    <title>ShiYanLou Blog</title>
    <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
    <meta name="viewport"
          content="width=device-width, initial-scale=1.0,
minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">
    <meta name="format-detection" content="telephone=no"/>
    <meta name="renderer" content="webkit"/>
    <meta http-equiv="Cache-Control" content="no-siteapp"/>
    <link rel="alternate icon" type="image/x-icon"
href="{{ URL::asset('i/favicon.ico') }}"/>
    <link rel="stylesheet"
href="//cdn.amazeui.org/amazeui/2.1.0/css/amazeui.min.css"/>
    {{ HTML::style('css/custom.css') }}
</head>
<body>
<header class="am-topbar am-topbar-fixed-top">
```

```
  <div class="am-container">
    <h1 class="am-topbar-brand">
      <a href="/">ShiYanLou Blog</a>
    </h1>
    @include('_layouts.nav')
  </div>
</header>

@yield('main')

@include('_layouts.footer')

<script src="//cdn.bootcss.com/jquery/2.1.3/jquery.min.js"></script>
<script src="//cdn.amazeui.org/amazeui/2.1.0/js/amazeui.min.js"></script>
</body>
</html>
```

URL::asset('i/favicon.ico')会生成 http://localhost:8000/i/favicon.ico，HTML::style('css/custom.css')会生成<link media="all" type="text/css" rel="stylesheet" href="http://localhost:8000/css/custom.css">，其中的 i 和 css 文件夹是放在 public 目录下的，public 目录是项目的资源文件夹。@include('_layouts.nav')会包含 app/views/_layouts/nav.blade.php 文件，@yield('main')是用于模版继承的。

修改 nav.blade.php：

```
<button class="am-topbar-btn am-topbar-toggle am-btn am-btn-sm am-btn-secondary am-show-sm-only"
        data-am-collapse="{target: '#collapse-head'}"><span class="am-sr-only">nav switch</span>
        <span class="am-icon-bars"></span></button>
<div class="am-collapse am-topbar-collapse" id="collapse-head">
  <div class="am-topbar-right">
    <a href="#" class="am-btn am-btn-primary am-topbar-btn am-btn-sm topbar-link-btn"><span class="am-icon-user"></span> Login</a>
```

```
  </div>
</div>
```

修改 footer.blade.php：

```
<footer class="footer">
  <p>© 2015 By <a href="http://www.shiyanlou.com" target="_blank">ww
w.shiyanlou.com</a></p></footer>
```

修改 index.blade.php：

```
@extends('_layouts.default')

@section('main')
<div class="am-g am-g-fixed blog-g-fixed">
  <div class="am-u-sm-12">
      <h1>Welcome to ShiYanLou! </h1>
  </div>
</div>
@stop
```

@extends('_layouts.default')会继承 app/views/_layouts/default.blade.php 文件，@yield('main')对应@section('main')并填充为其中的内容。

在 public 目录下新建两个文件夹 i 和 css，在 i 文件夹里放置一个名为 favicon.ico 的图标，在 css 文件夹下新建一个名为 custom.css 的文件，修改如下：

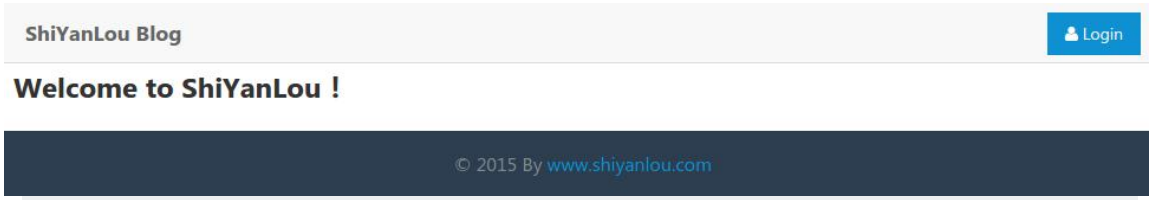
```
.footer p {
   color: #7f8c8d;
   margin: 0;
   padding: 15px 0;
   text-align: center;
   background: #2d3e50;}
.topbar-link-btn {
      color: #fff !important;}
```

7.修改路由访问首页

视图已经有了，这时候需要把路由跟视图进行关联，修改 app/routes.php 如下：

```
Route::get('/', function(){
    return View::make('index');
});
```

不出意外，这时候访问 localhost:8000 会出现下图这样：



终于见到了亲手编写的第一个页面，是不是有点小激动啊？

8.创建登录视图

在 nav.blade.php 中修改登录超链接的地址：

```
<a href="{{ URL::to('login') }}" class="am-btn am-btn-primary am-topbar-btn am-btn-sm topbar-link-btn"><span class="am-icon-user"></span> Login</a>
```

URL::to('login')会生成 http://localhost:8000/login 这个地址。

创建 login.blade.php：

```
$ php artisan generate:view login
```

修改 login.blade.php：

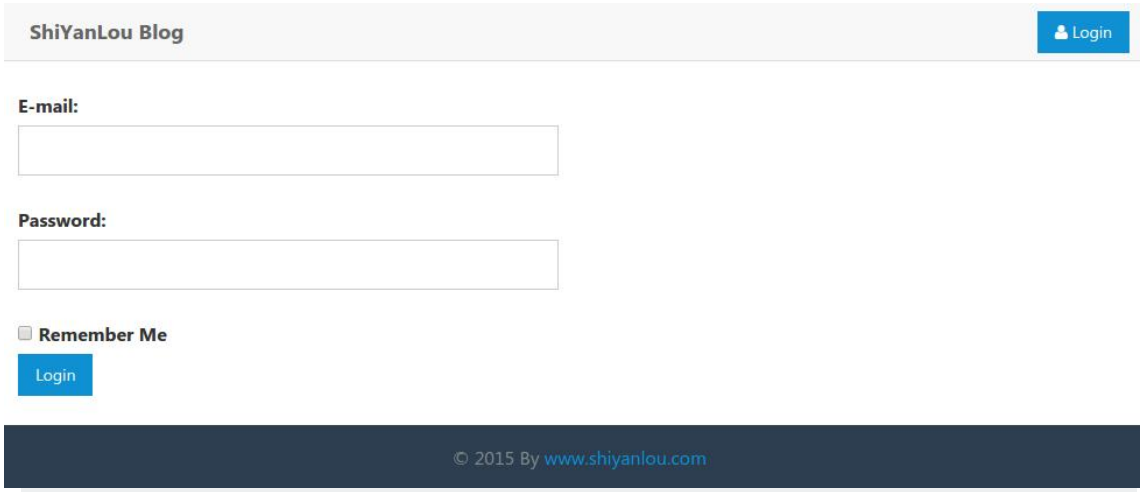
```
@extends('_layouts.default')

@section('main')
  <div class="am-g am-g-fixed">
    <div class="am-u-lg-6 am-u-md-8">
      <br/>
      @if (Session::has('message'))
        <div class="am-alert am-alert-danger" data-am-alert>
```

```
            <p>{{ Session::get('message') }}</p>
        </div>
    @endif
    @if ($errors->has())
        <div class="am-alert am-alert-danger" data-am-alert>
            <p>{{ $errors->first() }}</p>
        </div>
    @endif
    {{ Form::open(array('url' => 'login', 'class' => 'am-form')) }}
        {{ Form::label('email', 'E-mail:') }}
        {{ Form::email('email', Input::old('email')) }}
        <br/>
        {{ Form::label('password', 'Password:') }}
        {{ Form::password('password') }}
        <br/>
        <label for="remember_me">
            <input id="remember_me" name="remember_me"
type="checkbox" value="1">
            Remember Me
        </label>
        <br/>
        <div class="am-cf">
            {{ Form::submit('Login', array('class' => 'am-btn
am-btn-primary am-btn-sm am-fl')) }}
        </div>
    {{ Form::close() }}
    <br/>
  </div>
 </div>
@stop
```

在 routes.php 中增加：

```
Route::get('login', function(){
    return View::make('login');
});
```

这时候访问 localhost:8000/login 或者点击导航条的 Login 按钮会出现下图这样：



9.实现登录

创建用户登录后主页：

```
$ php artisan generate:view home
```

修改 home.blade.php：

```
@extends('_layouts.default')

@section('main')
<div class="am-g am-g-fixed blog-g-fixed">
  <div class="am-u-sm-12">
      <h1>Hello {{{ Auth::user()->nickname }}}</h1>
  </div>
</div>
@stop
```

上面的{{{ }}}可以对字符串做转义处理，一定程度上避免 XSS 攻击。

修改 nav.blade.php：

```
<div class="am-collapse am-topbar-collapse" id="collapse-head">
```

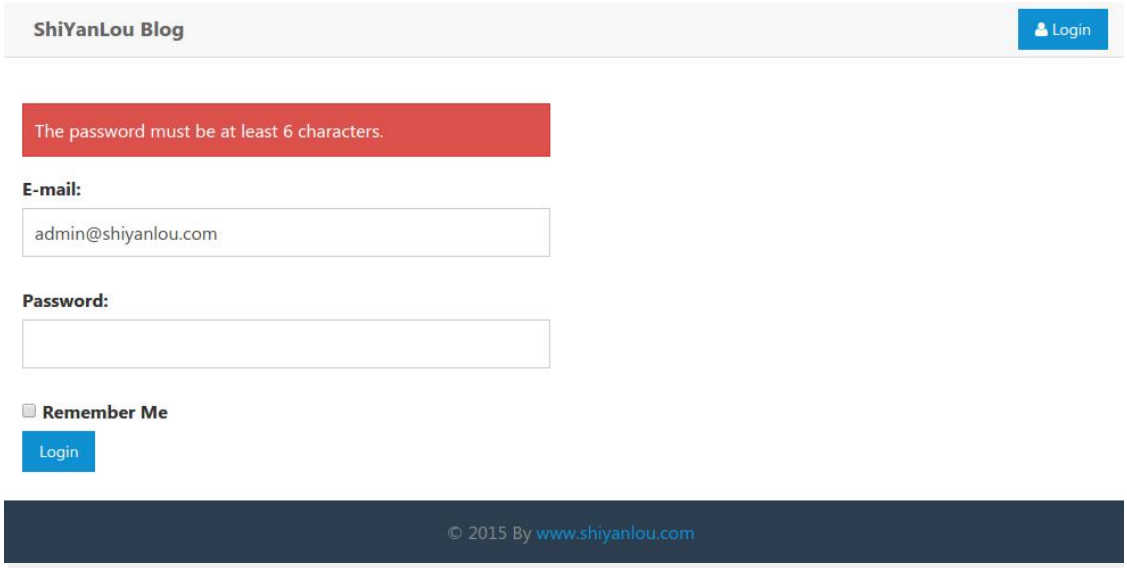
```
  @if (Auth::check())
    <ul class="am-nav am-nav-pills am-topbar-nav am-topbar-right">
      <li class="am-dropdown" data-am-dropdown>
        <a class="am-dropdown-toggle" data-am-dropdown-toggle href="javascrip
t:;">
          <span class="am-icon-users"></span> {{{ Auth::user()->nickname }}} <span
 class="am-icon-caret-down"></span>
        </a>
        <ul class="am-dropdown-content">
          <li><a href="{{ URL::to('logout') }}"><span class="am-icon-power-off"></spa
n> Exit</a></li>
        </ul>
      </li>
    </ul>
  @else
    <div class="am-topbar-right">
      <a href="{{ URL::to('login') }}" class="am-btn am-btn-primary am-topbar-btn
 am-btn-sm topbar-link-btn"><span class="am-icon-user"></span> Login</a>
    </div>
  @endif
</div>
```

在 Routes.php 中增加：

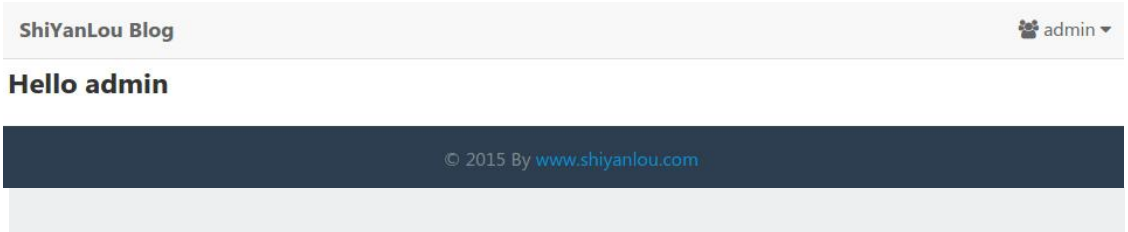
```
Route::post('login', array('before' => 'csrf', function(){
    $rules = array(
        'email'        => 'required|email',
        'password'     => 'required|min:6',
        'remember_me' => 'boolean',
    );
    $validator = Validator::make(Input::all(), $rules);
    if ($validator->passes())
    {
        if (Auth::attempt(array(
            'email'     => Input::get('email'),
            'password' => Input::get('password'),
            'block'     => 0), (boolean) Input::get('remember_me')))
        {
```

```
            return Redirect::intended('home');
        } else {
            return Redirect::to('login')->withInput()->with('message', 'E-mail or password error');
        }
    } else {
        return Redirect::to('login')->withInput()->withErrors($validator);
    }
}));

Route::get('home', array('before' => 'auth', function(){
    return View::make('home');
}));
```

下面就可以尝试用户登录了，如果输入信息有误，会出现错误信息如：



登录成功后会出现下图这样：



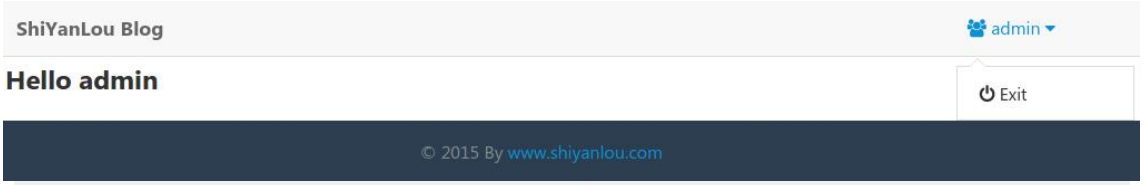
这里我们使用了 Laravel 自带的身份验证 Auth ,你也可以使用更加强大的 Sentry ,Web 表单验证用了 Validator，View 和 Redirect 详细可以查看视图和响应文档，还使用了路由过滤器，`csrf` 过滤器可以使我们轻松地防御 `csrf` 攻击。

10.退出登录

在 `routes.php` 中增加：

```
Route::get('logout', array('before' => 'auth', function(){
    Auth::logout();
    return Redirect::to('/');
}));
```

现在你就可以实现退出功能了，点击 `Exit`：



退出后会跳转到主页。

11.小结

至此简单的用户登录功能就完成了，你除了要完成上述的例子外，还要完成记住我的功能哦！你可以通过下面途径来完成：

- Laravel 官网
- 中文文档 1、中文文档 2
- 实验楼论坛
- Laravel 中文网问答社区
- PHPHub 中文社区
- API 文档
- laravel.io
- LARACASTS

代码下载：

```
$ git clone https://github.com/shiyanlou/laravel-blog-1.git
```