Laravel 大型项目系列教程（六）

优化、单元测试以及部署

一、前言

本节教程将讲解错误处理、配置文件的使用、单元测试以及部署到 Apache 服务器。

二、Let's go

1.错误处理

如果用户访问的 URL 不存在或者服务器存在错误时，我们不希望返货一个错误的页面，而想返回一个友好提示的页面，在 Laravel 中可以很轻松地实现，Laravel 有很简单的错误和日志处理，当服务器端存在错误时，`app/start/global.php` 里默认有一个处理所有异常的异常处理程序：

```
App::error(function(Exception $exception){
    Log::error($exception);});
```

它会把异常信息写到日志中，日志文件默认是 `app/storage/logs/laravel.log`。

如果要显示一个友好的错误提示页面，我们可以创建一个视图：

```
$ php artisan generate:view error
```

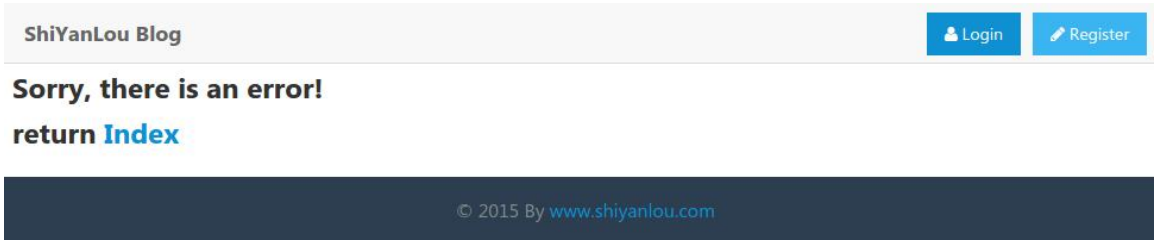
修改 `error.blade.php`：

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


    Sorry, there is an error!
        return Index


@stop
```

在 `App::error(function(Exception $exception)`中增加：

```
return Response::view('error', array(), 500);
```

现在当访问出现错误时，就会出现错误提示页面：



2.404 处理

当访问的 URL 不存在时，我们也可以返回一个友好的提示页面，先创建一个视图：

```
$ php artisan generate:view notFound
```

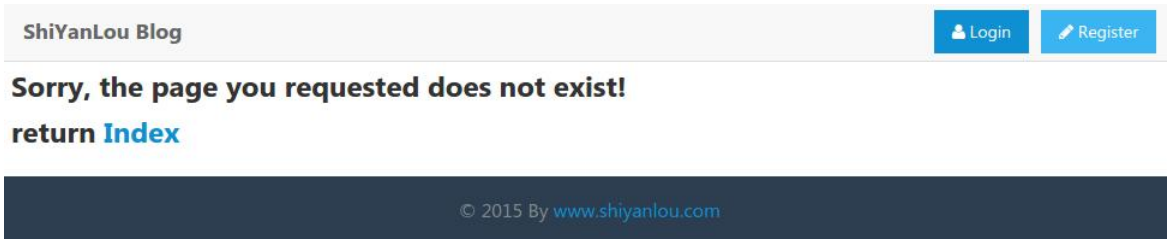
修改 `notFound.blade.php`：

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


    Sorry, the page you requested does not exist!
        return Index


@stop
```

在 `app/start/global.php` 中增加：

```
App::missing(function($exception){
    return Response::view('notFound', array(), 404);
});
```

现在当你访问的 URL 不存在时就会出现下面这样的页面：



3.配置文件

有时候我们可能需要一些事先就设定好的值，程序执行的时候只需要引用这个值，例如分页显示时每页显示的数量，我们可以使用配置文件，在 Laravel 中使用配置文件也很方便，我们可以在 app/config 下新建一个名为 custom.php，在其中添加：

```
return array(
    'page_size' => 10,
);
```

现在你就可以在程序中使用了，把 paginate(10)改成 paginate(Config::get('custom.page_size')就行，其中 custom 对应 app/config 下的文件名，page_size 对应相应配置文件中的键名，配置文件也可以根据你是开发环境还是生产环境进行不同的配置，详细的可以查看官方文档。

4.单元测试

在网站上线前，我们通常需要进行单元测试，Laravel 提供了很方便的单元测试模块。我这里仅实现一个例子 我们可以先在 app/tests 下创建一个名为 MyTest.php 的文件，在里面定义一个名为 MyTest 的类，切记要继承 TestCase 类，然后就可以写测试代码了：

```
class MyTest extends TestCase {

    public function testIndex()
    {
        $this->call('GET', '/');
        $this->assertResponseOk();
        $this->assertViewHas('articles');
        $this->assertViewHas('tags');
    }

    public function testNotFound()
    {
        $this->call('GET', 'test');
        $this->assertResponseStatus(404);
    }
}
```

测试代码写完之后，我们需要安装一个 phpunit 组件，在 composer.json 的 require-dev 中添加：

```
"phpunit/phpunit": "3.7.*"
```

然后 composer update 安装，完成后执行 vendor/bin/phpunit，稍等一会就会出现测试结果，在我们测试的时候如果想要做一些初始化操作，例如数据库迁移和填充等，可以定义在 setUp 方法中，切记要先执行 parent::setUp，测试完成之后如果想要恢复现场，可以在 tearDown 方法中进行，如果在测试的时候想要使用特定的配置文件，我们可以在 app/config/testing 目录下创建，测试时它会自动覆盖原来的配置。

5.部署至 Apache

测试通过后，我们可以把网站部署到应用服务器了，在生产环境中，我们应该把 `app/config/app.php` 中的 `debug` 设为 `false`。这里讲解怎么部署到 Apache 服务器上。首先声明我这里的 `LAMP` 环境是通过 `tasksel` 安装的 ,我们先安装 `mod_rewrite` 模块：

```
$ sudo a2enmod rewrite
```

然后把`/var/www` 目录的权限设为 777，这个目录是存放网站的目录：

```
$ sudo chmod -R 777 /var/www/
```

然后把我们开发的项目文件夹复制到这个文件夹中，我这里是 `blog` 文件夹：

```
$ cd /var/www/
$ cp -r ~/laravel-project/blog/ .
```

上面的开发项目路径要跟你自己的一样，之后我们需要把 `app/storage` 目录的权限改为 777，因为 `storage` 文件夹中会存放日志等，涉及到写操作：

```
$ cd blog/app/$ chmod -R 777 storage/
```

下面配置服务器：

```
$ sudo vim /etc/apache2/sites-enabled/000-default.conf
```

把 `DocumentRoot /var/www/html` 改成 `DocumentRoot /var/www/blog/public`，再修改 `apache2.conf`：

```
$ sudo vim /etc/apache2/apache2.conf
```

把

```
AllowOverride all
```

加到

```
Options Indexes FollowSymLinksAllowOverride NoneRequire all granted
```

之后，现在启动 Apache 服务器：

```
$ sudo service apache2 start
```

在浏览器中访问 `localhost` 或者 `127.0.0.1` 就可以看到我们的网站了 ,至此部署就完成了。

6.小结

本节教程讲了错误处理优化、配置文件的使用、单元测试以及怎么部署到 Apache 服务器，你可以买一个域名和一个服务器，最好买 VPS 云服务器，虚拟空间非常有局限性，然后把你自己写的网站部署到服务器让大家一起访问。

最后的代码下载：

```
$ git clone https://github.com/shiyanlou/laravel-blog-6.git
```

本文详细出处：http://www.shiyanlou.com/courses/123