在学习完前面两章的内容之后,我相信你绝对已经迷上了Symfony,成为了他的忠实粉丝,无需再费周折,接下来我将带着你去了解Symfony的控制器(`controllers`),现在,我忠实的读者,看看它到底能为你做些什莫呢?.

返回原生的响应(Returning Raw Responses)

Symfony的将自 己定义为一个请求 - 响应(`Request-Response`)框架。当用户发出请求(`request`)到应用程序时，Symfony的创建一个 `Request` 对象来封装所有相关的必不可少的信息。同样，执行任何控制器(`controller`)的任何动作(`action`)的结果 是创建了一个 `Response` 的对象,这个对象是Symfony生成用来返回给用户的HTML内容。

到目前为止，本教程中的操作一般使用 `$this->render()` 方法 返回的渲染响应作为结果。如果你需要它，你也可以创建一个原始 `Response` 对象返回任何文本内容：

```
// src/AppBundle/Controller/DefaultController.php
namespace AppBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Response;

class DefaultController extends Controller
{
    /**
     * @Route("/", name="homepage")
     */
    public function indexAction()
    {
        return new Response('Welcome to Symfony!');
    }
}
```

路由参数(Route Parameters)

在大多数情况下，应用程序的URL包括在他们变量部件(`variable parts`)。如果您正在创建类似于博客那样的应用程序，博客上显示的文章应当包括文章标题以及其他唯一标识符的URL，让应用程序确切的知道要显示那一篇文章。

在Symfony的应用程序中，路由的变量部件(`variable parts`)被包含在大括号中(`/blog/read/{article_title}/`)。每个变量的部分被赋予一个唯一的名字以便以后用于在控制器检索每一个变量的值.

接下来我们将为路由变量(`route variables`)创建一个新的动作(`action)` ，用来操作路由参数,打开 `src/AppBundle/Controller/DefaultController.php` 文件,并且添加一个名为 `helloAction` 的方法,如下所示:

```php
// src/AppBundle/Controller/DefaultController.php
namespace AppBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class DefaultController extends Controller
{
    // ...

    /**
     * @Route("/hello/{name}", name="hello")
     */
    public function helloAction($name)
    {
        return $this->render('default/hello.html.twig', array(
            'name' => $name
        ));
    }
}
```

打开你的浏览器并且访问 `http://localhost:8000/hello/fabien` ,就可以看到执行这个新的动作(`action`)的结果了.如果人品太差,你会看到一个错误页面.或许你可能已经猜到,导致这个错误的原因是因为你尝试渲染一个不存在的模板(`default/hello.html.twig)` ,创建一个新的模板并且添加如下内容:

```twig
{# app/Resources/views/default/hello.html.twig #}
{% extends 'base.html.twig' %}

{% block body %}
    <h1>Hi {{ name }}! Welcome to Symfony!</h1>
{% endblock %}
```

再次浏览 `http://localhost:8000/hello/fabien` 页面,你将会看到你传入的路由参数已经被控制器渲染到这个模板上了.如果你想要更改URL的最后一部分(e.g. `http://localhost:8000/hello/thomas`) 并且重新加载网页内容,页面将会显示不同的信息.如果你去掉了URL的最后一部分内容(e.g. `http://localhost:8000/hello`), symfony将会显示一个错误的页面，因为该路由需要一个名字，但是你没有提供给它。

使用格式(Using Formats)

如今，Web应用程序应该不仅仅只是提供HTML页面了。从XML的RSS源以及Web服务，到JSON的Ajax请求，也有很多不同的格式可供选择。Symfony支持这些格式的是直接得益于一个被称为 `_format` 的特殊变量,该变量用来存储用户请求的格式。

调整 `hello` 的路由,为html添加一个 `_format` `变量` 作为默认值:

```php
// src/AppBundle/Controller/DefaultController.php
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;

// ...

/**
 * @Route("/hello/{name}.{_format}", defaults={"_format"="html"}, name="hello")
 */
public function helloAction($name, $_format)
{
    return $this->render('default/hello.'.$_format.'.twig', array(
        'name' => $name
    ));
}
```

很显然，当你想要支持多种格式的请求时，你必须为每个支持的格式提供一个模板。在这种情况下，你应该创建一个新的 `hello.xml.twi g` 模板：

```
<!-- app/Resources/views/default/hello.xml.twig -->
<hello>
    <name>{{ name }}</name>
</hello>
```

现在,当你访问 `http://localhost:8000/hello/fabien` 时,你会看到普通的HTML页面,因为 `html` 是默认格式,当访问 `http://localhost:8000/hello/fabien.html` 时, 你会再次得到HTML页面， 这是因为这次你已经明确要求为 `html` 格式 .最后,你可以访问 `http://localhost:8000/hello/fabien.xml` ,你会看到在你的浏览器中呈现新的XML模板。

还有一点,对于标准的格式,Symfony的也将自动选择最好的 `Content-Type` 首部去响应.如果你想要限制一个给定的动作给支持的格式,你可以使用 `@Route()` 注释的 `requirements` 选项,如下所示:

```
// src/AppBundle/Controller/DefaultController.php
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;

// ...

/**
 * @Route("/hello/{name}.{_format}",
 *     defaults = {"_format"="html"},
 *     requirements = { "_format" = "html|xml|json" },
 *     name = "hello"
 * )
 */
public function helloAction($name, $_format)
{
    return $this->render('default/hello.'.$_format.'.twig', array(
        'name' => $name
    ));
}
```

这样一来,该 `hello` 动作将匹配像 `/hello/fabien.xml` 或 `/hello/fabien.json` 的网址.如果你尝试得到像 `/hello/fabien.js` 的 网址，它会返回一个404错误页面,因为 `_format` 变量的值不符合其要求。

重定向(Redirecting)

如果您想将用户重定向到另一个页面，使用 `redirectToRoute()` 方法：

```
// src/AppBundle/Controller/DefaultController.php
class DefaultController extends Controller
{
    /**
     * @Route("/", name="homepage")
     */
    public function indexAction()
    {
        return $this->redirectToRoute('hello', array('name' => 'Fabien'));
    }
}
```

`redirectToRoute()` 方法利用参数的路由名称和作为参数的可选数组去生成URL,并且向重定向站点.

显示错误页面(Displaying Error Pages)

每个Web应用程序的执行过程中不可避免地发生错误。在 `404` 错误的情况下，Symfony提供了一个快捷的处理方式，你可以在你的控制器使用如下方法：

```php
// src/AppBundle/Controller/DefaultController.php
// ...

class DefaultController extends Controller
{
    /**
     * @Route("/", name="homepage")
     */
    public function indexAction()
    {
        // ...
        throw $this->createNotFoundException();
    }
}
```

对于 `500` 错误，PHP异常控制器只是抛出一个普通的异常,但Symfony却将它改造成一个适当的 `500` 错误页面：

```php
// src/AppBundle/Controller/DefaultController.php
// ...

class DefaultController extends Controller
{
    /**
     * @Route("/", name="homepage")
     */
    public function indexAction()
    {
        // ...
        throw new \Exception('Something went horribly wrong!');
    }
}
```

从请求中获取信息(Getting Information from the Request)

有时你的控制器需要访问信息依赖于用户请求，例如他们更喜欢的语言,IP地址或URL查询参数。要访问这些信息，需要添加一个新类型的 `Request` 的动作参数。这个新参 数的名称并不重要，但必须是工作之前的请求类型（不要忘了补充 `use` 申明，导入这个 `Request` 类）：

```php
// src/AppBundle/Controller/DefaultController.php
namespace AppBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;

class DefaultController extends Controller
{
    /**
     * @Route("/", name="homepage")
     */
    public function indexAction(Request $request)
    {
        // is it an Ajax request?
        $isAjax = $request->isXmlHttpRequest();

        // what's the preferred language of the user?
        $language = $request->getPreferredLanguage(array('en', 'fr'));

        // get the value of a $_GET parameter
        $pageName = $request->query->get('page');

        // get the value of a $_POST parameter
        $pageName = $request->request->get('page');
    }
}
```

在模板中，你还可以通过由Symfony的自动提供的特殊 `app.request` 变量来访问 `Request` 对象：

```
{{ app.request.query.get('page') }}

{{ app.request.request.get('page') }}
```

在会话中持久化数据(Persisting Data in the Session)

虽然 `HTTP` 协议是无状态的，但是Symfony的提供了一个很好的会话对象表示客户（无论是使用浏览器一个真正的人，还是一个机器人，或Web服务）。在两个请求之间，Symfony的使用PHP本身的会话存储在 `cookie` 中的属性。

存储和检索来自session信息可以通过任何控制器来实现：

```
use Symfony\Component\HttpFoundation\Request;

public function indexAction(Request $request)
{
    $session = $request->getSession();

    // store an attribute for reuse during a later user request
    $session->set('foo', 'bar');

    // get the value of a session attribute
    $foo = $session->get('foo');

    // use a default value if the attribute doesn't exist
    $foo = $session->get('foo', 'default_value');
}
```

你还可以存储“提示信息(`flash messages`)”,这条信息将下一个请求后被自动删除。当需要将用户重定向至另一页面（将显示消息）之前设置成功消息时,这是非常有用的：

```
public function indexAction(Request $request)
{
    // ...

    // store a message for the very next request
    $this->addFlash('notice', 'Congratulations, your action succeeded!');
}
```

你也可以在模板中显示提示消息,像这样：

```
<div>
    {{ app.session.flashbag.get('notice') }}
</div>
```

最后的思考

这一切就是这么简单，我甚至不知道你已经花了整整10分钟。我在第一部分简要介绍了捆绑，而你迄今为止已经了解了Symfony功能核心框架包的一部分。但由于捆绑，一切都在Symfony中是可扩展或和可替换的。这将在后面讲到.

我的主页:http://aifei8.net:

我的博客:http://blog.aifei8.net

github主页:http://github.com/Pengfei-Gao

Love Princess Studio

Pengfei-Gao

Love Princess Studio
吉林省延吉市延边大学
net.aifei8@gmail.com
http://aifei8.net
Joined on 30 Dec 2015

0	4	0
Followers	Starred	Following

Contributions Repositories Public activity Edit profile

Popular repositories

YBUInformation 1
一个延边大学urp教务系统学生信息查询的示例

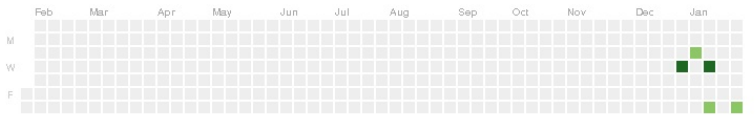
develop-reference-data 1
一些常用的开发文档

Ionic 0
Advanced HTML5 mobile development framework and SDK. Build incredible mobile apps with web technologies y…

source-insight-3-for-centos7 0
centos7 下繁成的source Insight 3

Symfony-doc-translate 0
Symfony的中文文档

Contributions



Summary of pull requests, issues opened, and commits. Learn how we count contributions. Less More