阅读本教程的第一部分之后，你已经决定，Symfony的是价值10分钟。在第二部分中，您将了解更多关于Twig的知识，一个快速，灵活，安全的模板引擎的PHP应用 程序。Twig，将使你的模板更易读,更简洁;这也使得他们的网页设计师更友好。

熟悉Twig(Getting familiar with Twig)

官方的Twig文档是学习一切有关模板引擎的最佳资源。本节只是给你它的主要概念的简要概述。

Twig模板是可产生任何类型的内容（HTML，CSS，JavaScript中，XML，CSV乳胶等）的文本文件,他的节点元素使用如下这些分隔符与模板内容的其余部分做区分：

`{{ ... }}`	打印变量的内容或计算表达式的结果
`{% ... %}`	控制所述模板的逻辑;它用于例如用于 `for` 循环和if语句来执行。
`{# ... #}`	允许包括注释里面的模板。相反HTML注释，它们不包括渲染模板中。

下面是一个最小的模板示出的一些基本知识，使用两个变量 `page_title` 和 `navigation` ，这将被传递到模板：

```
<!DOCTYPE html>
<html>
    <head>
        <title>{{ page_title }}</title>
    </head>
    <body>
        <h1>{{ page_title }}</h1>

        <ul id="navigation">
            {% for item in navigation %}
                <li><a href="{{ item.url }}">{{ item.label }}</a></li>
            {% endfor %}
        </ul>
    </body>
</html>
```

要渲染的Symfony模板，从控制器中使用 `render` 方法。如果模板需要的变量产生的内容，将它们作为使用第二个可选的参数数组：

```
$this->render('default/index.html.twig', array(
    'variable_name' => 'variable_value',
));
```

传递给一个模板变量可以是字符串，数组甚至对象。Twig提取它们之间的差别，并允许您访问变量的“属性”以点（ `.` ）记号。下面的代码清单显示了类型控制器如何通过一个变量显示内容：

```
{# 1. Simple variables #}
{# $this->render('template.html.twig', array('name' => 'Fabien') ) #}
{{ name }}

{# 2. Arrays #}
{# $this->render('template.html.twig', array('user' => array('name' => 'Fabien')) ) #}
{{ user.name }}

{# alternative syntax for arrays #}
{{ user['name'] }}

{# 3. Objects #}
{# $this->render('template.html.twig', array('user' => new User('Fabien')) ) #}
{{ user.name }}
{{ user.getName }}

{# alternative syntax for objects #}
{{ user.name() }}
{{ user.getName() }}
```

装饰模板(Decorating Templates)

通常情况下，在一个项目中有着共同的元素，如著名的页眉和页脚模板。Twig使用一个名为“模板继承”的概念优雅的解决了这个问题。此功能允许你建立一个包含你的 站点的所有公共元素，并定义内容的子模板可以覆盖的“块”基本模板。

所述 `index.html.twig` 模板使用 `extends` 标记以表明它从 `base.html.twig` 模板继承：

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

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

打开应用 `app/Resources/views/base.html.twig` 对应于 `base.html.twig` 模板文件，你会发现下面的Twig代码：

```
{# app/Resources/views/base.html.twig #}
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}{% endblock %}
        <link rel="icon" type="image/x-icon" href="{{ asset('favicon.ico') }}" />
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}{% endblock %}
    </body>
</html>
```

该 `{% block %}` 标签告诉模板引擎，子模板可以覆盖模板的这些部分。在这个例子中， `index.html.twig` 模板覆盖 `body` ，但不是 `title` ，这 将显示在 `base.html.twig` 模板中定义的默认内容。

使用标签，过滤器和函数(Using Tags, Filters, and Functions)

Twig的最佳功能之一是标签，过滤器，和功能的可扩展性。看看一个使用广泛的过滤器显示给用户之前修改信息的示例模板：

```
<h1>{{ article.title|capitalize }}</h1>

<p>{{ article.content|striptags|slice(0, 255) }} ...</p>

<p>Tags: {{ article.tags|sort|join(", ") }}</p>

<p>Activate your account before {{ 'next Monday'|date('M j, Y') }}</p>
```

不要忘记检查出的官方 Twig文档，了解一切关于过滤器的功能和标签。

包括其他模板

几个模板之间共享代码片段的最好方法是创建一个可以被包含来自其他模板新的模板片段。

试想一下，我们要显示某些页面上我们的应用程序的广告。首先，创建一个 `banner.html.twig` 模板：

```
{# app/Resources/views/ads/banner.html.twig #}
<div id="ad-banner">
    ...
</div>
```

要想在页面上显示广告，可以使用 `include()` 函数和 `banner.html.twig` 模板：

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

{% block body %}
    <h1>Welcome to Symfony!</h1>

    {{ include('ads/banner.html.twig') }}
{% endblock %}
```

嵌入其他控制器

如果你想在模板嵌入另一个控制器的会怎摸样呢?即使用Ajax工作时，或当嵌入模板需要主模板没有的一些变量时,这是非常有用的。

假设你已经创建了一个 `topArticlesAction` 控制器的方法来显示你的网站最流行的文章。如果你想“渲染”的方法（通常是一些HTML内容） `index` 模板中的结果，使用 `render()` 函数：

```twig
{# app/Resources/views/index.html.twig #}
{{ render(controller('AppBundle:Default:topArticles')) }}
```

这里， `render()` 和 `controller()` 函数使用特殊的 `AppBundle:Default:topArticles` 语法来表示 `Default` 控制器的 `topArticlesAction` 操作（ `AppBundle` 部分将在稍后说明）：

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

class DefaultController extends Controller
{
    public function topArticlesAction()
    {
        // look for the most popular articles in the database
        $articles = ...;

        return $this->render('default/top_articles.html.twig', array(
            'articles' => $articles,
        ));
    }

    // ...
}
```

页面之间创建链接

对于Web应用程序来说,创建页面之间的链接是必需的。用来代替在模板硬编码的URL，该 `path` 函数知道如何生成基于所述路由的配置网址。这样一来，所有的网址，可以很容易地更新，只需改变配置：

```twig
<a href="{{ path('homepage') }}">Return to homepage</a>
```

`path` 函数将路径名作为第一个参数，你可以选择通过路由参数数组作为第二个参数。

使用url参数也可以达到目标,该 `url` 函数和 `path` 函数是非常相似的 ，但是它会生成绝对URL，使电子邮件和RSS文件时，这是非常方便：

```twig
<a href="{{ url('homepage') }}">Visit our website</a>.
```

包括其他文件：图像，JavaScript和样式表

互联网怎莫能没有图像，JavaScript和样式表呢？ symfony提供 `asset` 函数，轻松应对这个问题：

```
<link href="{{ asset('css/blog.css') }}" rel="stylesheet" type="text/css" />

<img src="{{ asset('images/logo.png') }}" />
```

`asset()` 函数查找 `web/` 目录内的网络资产。如果您将它们保存在其他目录，请阅读*this article* ，以了解如何管理网络资产。

使用 `asset` 函数 ，您的应用程序可以更轻量。其原因是，你可以在你的Web根目录下的任何地方移动应用程序根目录不改变你的模板代码。

最后的思考

Twig是非常简单的但功能特别强大。由于布局，模块，模板和行动夹杂，这种逻辑和可扩展的方式是很容易地组织你的模板的。

如果一直以来都在看我翻译的文档,那末你可能接触Symfony的大约只有20分钟，但你已经可以做一些非常了不起的东西了。这就是Symfony的的力量。学习基础知识是很容易的，你很快就会发现，这种简单是隐藏在一个非常灵活的架构下的。

但是,我自己一直都在走,却从未到达过终点.