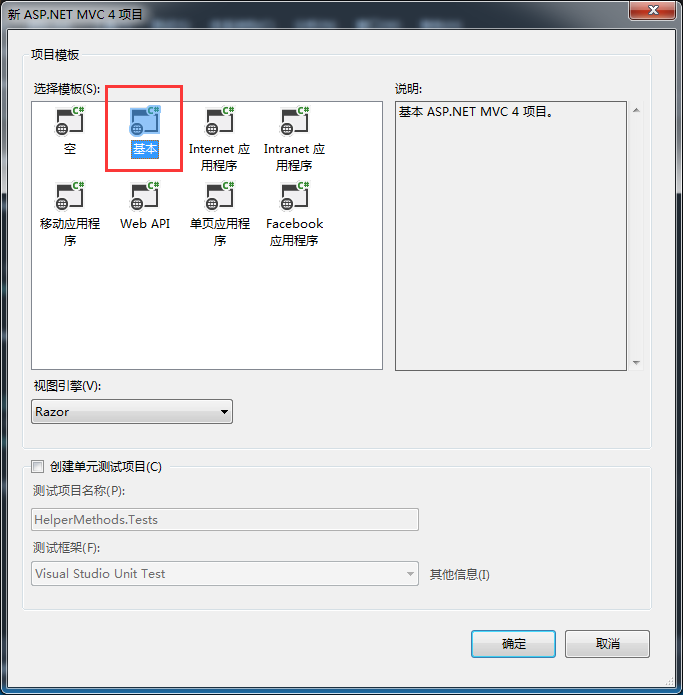
# 辅助器方法

在开发ASP.NET MVC项目时，经常会用到辅助器方法，它能够提供很多遍历的工作，比如可以对代码块和标记进行打包等。下面通过新建一个示例项目来看看其在实际使用中的情况。

## 示例项目

* 项目名：HelperMethods
* 模板：Basic（基本）



首先要在项目中添加一个控制器——Home：

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace HelperMethods.Controllers

{

public class HomeController : Controller

{

public ActionResult Index()

{

ViewBag.Fruits = new string[] { "Apple", "Orange", "Pear" };

ViewBag.Cities = new string[] { "New York", "London", "Paris" };

string message = "This is an HTML element: <input>";

return View((object)message);

}

}

}

该控制器的Index动作方法的视图为一个强类型视图，且未使用布局，内容如下：

@model string

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<div>

Here are the fruits:

@foreach (string str in (string[])ViewBag.Fruits)

{

<b>@str</b>

}

</div>

<div>

Here are the cities:

@foreach (string str in (string[])ViewBag.Cities)

{

<b>@str</b>

}

</div>

<div>

Here is the message:

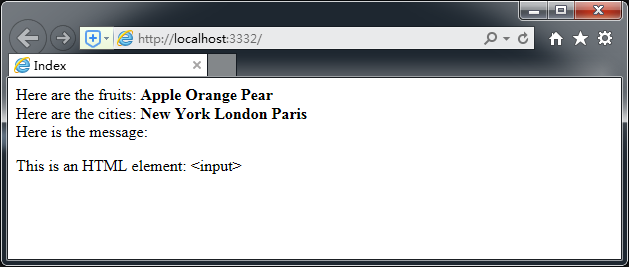
<p>@Model</p>

</div>

</body>

</html>

效果如图：



## 创建自定义辅助器方法

后面通过创建自定义的辅助器方法了解内联辅助器方法和外部辅助器方法。

### 内联的辅助器方法

故名思议，内联辅助器方法是在视图中定义的一种辅助器方法，这也是最简单的一种辅助器方法。下面创建的自定义内联辅助器方法使用了@helper标签，内容如下：

@model string

@{

Layout = null;

}

**@helper ListArrayItems(string[] items)**

**{**

**foreach (string str in items)**

**{**

**<b>@str</b>**

**}**

**}**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<div>

Here are the fruits: @**ListArrayItems(ViewBag.Fruits)**

</div>

<div>

Here are the cities: @**ListArrayItems(ViewBag.Cities)**

</div>

<div>

Here is the message:

<p>@Model</p>

</div>

</body>

</html>

从上面的代码中可以看出内联辅助器方法的定义类似于规则的C#方法。内联辅助器具有名称和参数。但是这和一般的方法不同的是它没有返回值。

提示：在使用内敛的辅助器时，不需要将ViewBag的动态属性转换成相应的类型，辅助器方法会在运行时进行评估。

内联辅助器的方法体遵循Razor视图其余部分的通用语法。即：文字字符串被视为静态HTML，需要有Razor处理的语句需要以@字符为前缀。

### 外部辅助器方法

外部的HTML辅助器方法会被表示成C#的扩展方法。它的使用更为广泛，但由于C#不能方便的直接处理HTML元素的生成，所以，编写这类的方法会更难一些。下面是一个自定义的外部辅助器方法——CustomHelpers.cs，内容：

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace HelperMethods.Infrastructure

{

public static class CustomHelpers

{

public static MvcHtmlString ListArrayItems(this HtmlHelper html, string[] list)

{

TagBuilder tag = new TagBuilder("ul");

foreach (string str in list)

{

TagBuilder itemTag = new TagBuilder("li");

itemTag.SetInnerText(str);

tag.InnerHtml += itemTag.ToString();

}

return new MvcHtmlString(tag.ToString());

}

}

}

上面的方法ListArrayItems采取的是扩展方法的方式，这一点在第一个参数中通过this关键字进行了定义。该类型的参数具有一些很有用的属性，这里简单做个列举：

**HtmlHelper：**

* RouteCollection：返回应用程序定义的路由集合
* ViewBag：返回视图包数据，这些数据是从动作方法传递给调用辅助器方法的视图的
* ViewContext：返回ViewContext对象，该对象能够对请求的细节以及请求的处理方式进行访问

当创建的内容与正在处理的请求有关时，ViewContext属性非常有用，下面是该属性具有的一些常用的属性：

**ViewContext**

* Controller：返回处理当前请求的控制器，继承自ControllerContext
* HttpContext：返回描述当前请求的HttpContext对象，继承自ControllerContext
* IsChildAction：如果已调用辅助器的视图正由一个子动作渲染，则返回true，继承自ControllerContext
* RouteData：返回请求的路由数据，继承自ControllerContext
* View：返回已调用辅助器方法的IView实现的实例

在创建HTML元素时，最简单常用的方式是使用TagBuilder类，它能够建立HTML字符串，而且可以处理各种转义及特殊字符。

该类具有的一些最有用的成员如下：

* InnerHtml：将元素内容设置为HTML字符串的一个属性。赋给这个属性的值将不进行编码，即可以将它嵌入HTML元素
* SetInnerText(string)：设置HTML元素的文本内容。String参数将被编码，以使它安全显示
* AddCssClass(string)：对HTML元素添加一个CSS的class
* MergeAttribute(string,string,bool)：对HTML元素添加一个标签属性。第一个参数是标签属性名称，第二个是其值。bool参数指定是否替换已存在的同名标签属性

说明：为了区别其他属性的说法，这里讲HTML元素的属性叫做标签属性，如<a id=”123” />，其中，id就是a元素的一个标签属性。

下面来看一下外部辅助器方法是如何使用的，这和前面的内联辅助器方法的使用不同：

@model string

**@using HelperMethods.Infrastructure**

@{

Layout = null;

}

<!--自定义的内联辅助器方法-->

@\*@helper ListArrayItems(string[] items)

{

foreach (string str in items)

{

<b>@str</b>

}

}\*@

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<!--使用内联辅助器方法-->

@\*<div>

Here are the fruits: @ListArrayItems(ViewBag.Fruits)

</div>

<div>

Here are the cities: @ListArrayItems(ViewBag.Cities)

</div>\*@

<!--使用自定义的外部辅助器方法-->

<div>

Here are the fruits: **@Html.ListArrayItems((string[])ViewBag.Fruits)**

</div>

<div>

Here are the cities: **@Html.ListArrayItems((string[])ViewBag.Cities)**

</div>

<div>

Here is the message:

<p>@Model</p>

</div>

</body>

</html>

从上面示例代码中可以看出使用外部辅助器方法的方式，下面做一下简单总结（上面示例中保留了内联的辅助器方法以及其使用的代码——注释调用部分，是为了方便比较这两种辅助器方法的使用的对比）。

使用外部辅助器方法的注意事项：

1. 需要引用辅助器扩展方法所在的命名空间。引用方式：@using，但如果定义了大量的自定义辅助器，最好将它们的命名空间添加到/Views/Web.config文件中，以方便它们在视图中总是可用的。
2. 使用@ Html.<helper>引用辅助器，<helper>表示扩展方法的名称（即定义的外部辅助器方法的名称）。在使用外部辅助器方法时需要注意的是需要将传递给方法参数的数据转换成对应的类型，这点和内联辅助器方法的使用不同。比如这里的ViewBag.Fruits需要转成string数组类型。

使用辅助器方法的时机

一个让我们容易迷惑的是该何时使用辅助器方法，而不是分部视图或子动作，尤其是在特性功能之间有重叠的时候。

使用辅助器方法只是为了减少视图中的重复量，且只用于简单的内容。对于更复杂的标记和内容，可以使用分部视图，而在需要实现对模型数据操作时，则应该使用子动作。建议要注意保持辅助器方法尽可能简单（如果辅助器不只是含有少量的C#代码，或者C#代码多余HTML元素，那么最好选择子动作）。

### 管理辅助器方法中的字符串编码