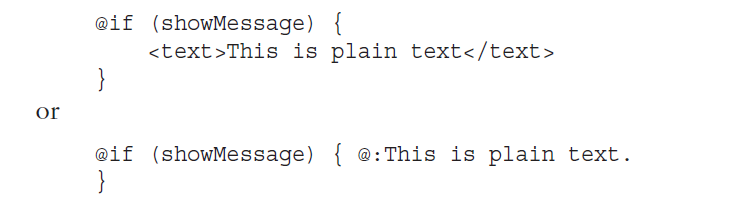
1. Controller继承自System.Web.Mvc.Controller，Controller中每一个public方法叫做action方法。Each controller’s class name ends with Controller。Views that controllers use live in a subdirectory of the Views main directory and are named according to the controller name (minus the Controller suffi x). All reusable UI elements live in a similar structure, but in a Shared directory in the Views folder
2. 默认的URL映射到HomeController的Index的action方法。
3. Razor表示@。@告诉RAZOR引擎后面的是代码，不是HTML.
4. HttpHandler是extension(拓展名) based 预处理，HttpModule是event based预处理
5. 来自query string 或者 form post名为param的参数。asp.net mvc 路由默认将action后的部分命为名ID，如果action 中有一个名为id的参数，mcv将其赋给id。
6. 可以传递给页面一个model。在页面中使用@model type全限定，在View中使用Model来引用该强类型。也可以使用@using来引用命名空间
7. 可以通过ViewBag和ViewData从controller传递数据到View。ViewData[“CurrentTime”] = DateTime.Now ViewBag.CurrentTime = DateTime.Now ViewBag是dynamic版的ViewData.
8. Razor also supports explicit code expressions by wrapping them in parentheses. @(param) 尤其是调用泛型方法时使用
9. @@表示一个@字符或者使用@#64;
10. Razor表达式自动Html Encoded。@Html.Raw(message)是用来显示原生Html. @Ajax.JavaScriptStringEncode(ViewBag.Username)来在js中Encoded
11. @{}代码块
12. Razor使用<text>标签作为特定标签输出内容。@:输出单行文本。



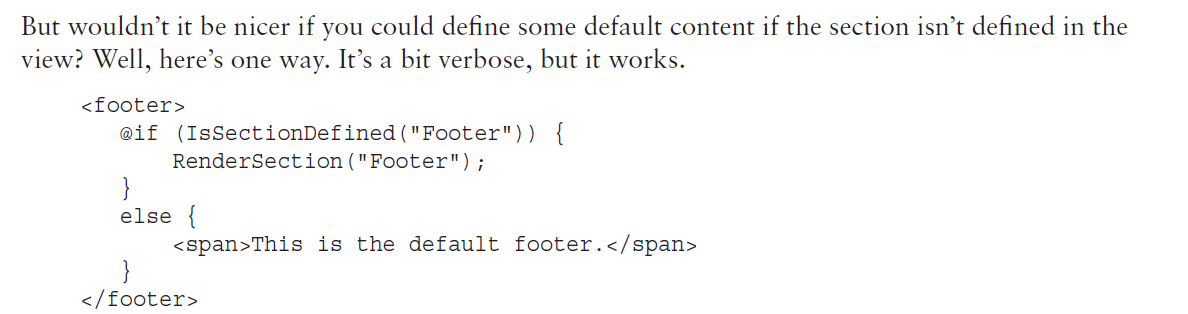
1. @\* \*@用来进行多行注释

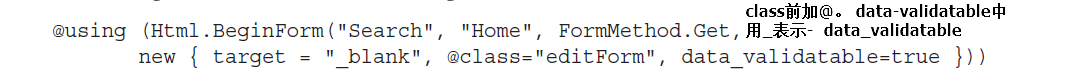
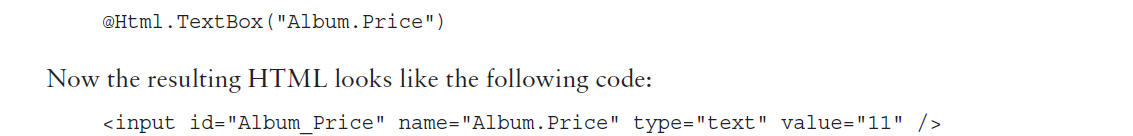
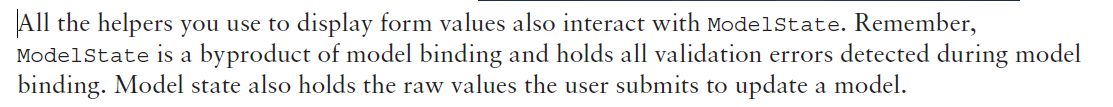
Layout类似于MasterPage为其它views定义了模板。如模板view定义了<div id="main-content">@RenderBody()</div> 。子类View定义Layout为：Layout = "~/Views/Shared/SiteLayout.cshtml"; 一个Layout可以有多个section。 如**<footer>@RenderSection("Footer",required:true)</footer>**。默认情况下，子类View必须为每个section提供定义。

如：**@section Footer {**

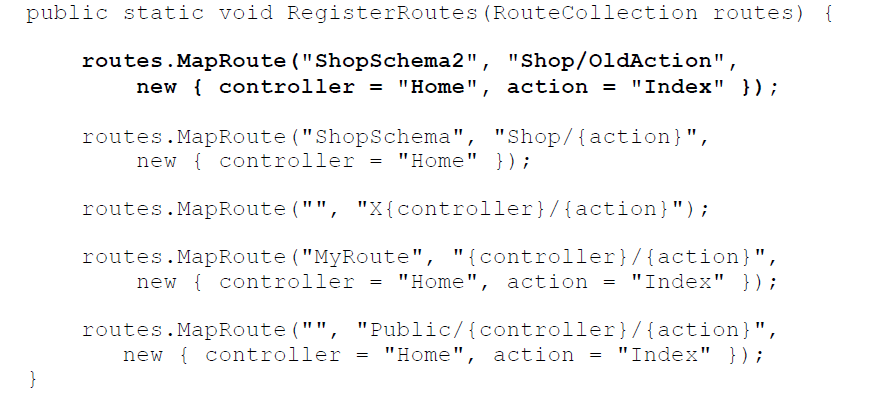
**This is the <strong>footer</strong>.**

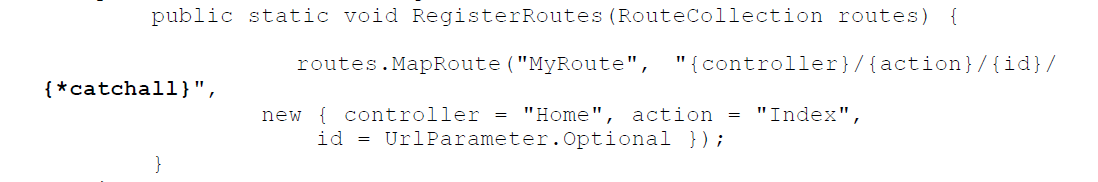
**}**



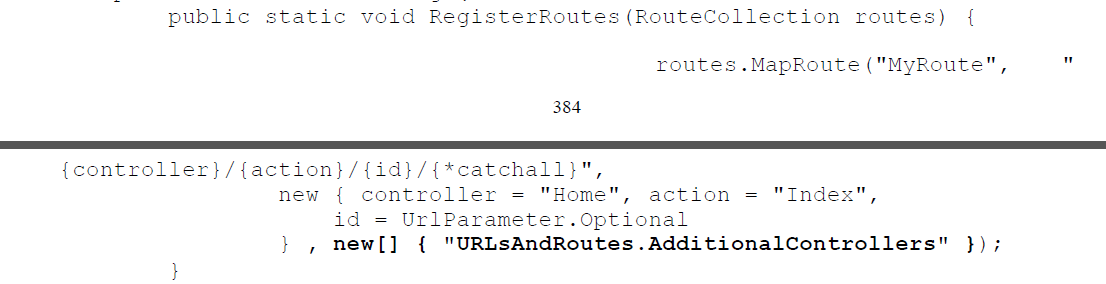
1. \_ViewStart.cshtml中的代码在相同目录中其他view的代码之前执行。并且递归的应用于子目录。
2. partial view跟正常view的区别是它不指定Layout，即使在\_ViewStart.cshtml中定义了Layout.
3. 当Action中有参数时，MVC使用Model Builder去构建参数。默认的Model Builder是DefaultModelBuilder。
4. Model binding implicitly goes to work when you have an action parameter. You can also explicitly invoke model binding using the UpdateModel and TryUpdateModel methods in your controller. UpdateModel throws an exception if something goes wrong during model binding or the model is invalid.
5. 
6. Html Helper通过Html来访问，Url Helper通过Url来访问，Ajax Helper通过Ajax来访问。Html输出到view的内容自动Html Encoded.
7. Html Helper获取数据从ViewData中，ViewData包含ViewBag和Model. Notice that the id attribute of the resulting input element uses an underscore instead of a dot。属性可以通过.来级联
8. 
9. ChildActionOnlyAttribute. The attribute preventsthe runtime from invoking the action directly via a URL.
10. Views中以\_开头的文件名不能返回给用户，它们是Support Files，例如Layout。

当框架查找View时，首先查找按Controller命名的Views/Controller名，然后查找Views/Shared文件夹。This means that I can put views that will be used by more than one controller in the /Views/Shared folder and the framework will find them.

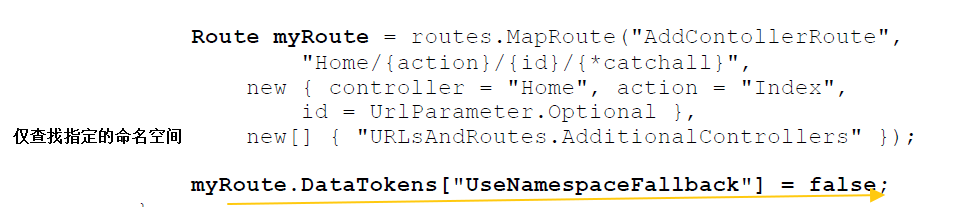
1. Route是按照顺序匹配的。
2. 
3. 可变数量的Segment，使用{\*catchall}来实现。catchall捕获其它所有的参数。



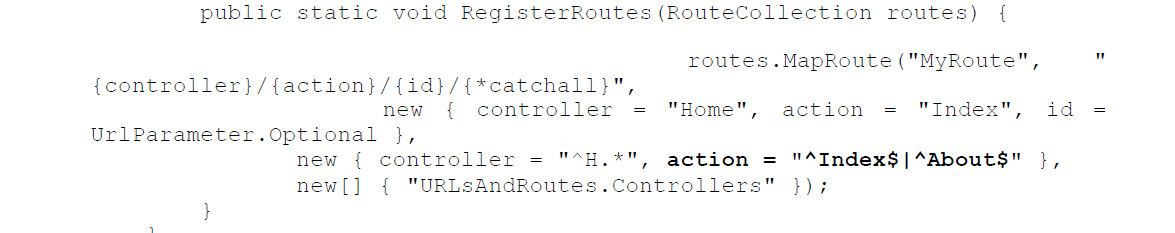
1. 指定Namespace过滤Controller，new[]中的Namespace是等价的，它们给予了优先查找权，如果查找不到再去查找其它的Namespace。



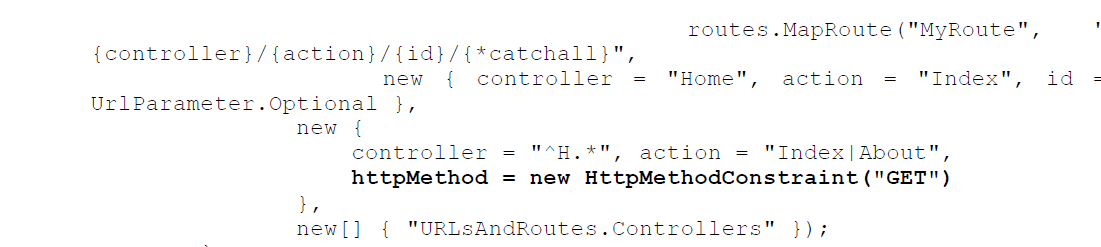
以下方法指定只查找指定的命名空间

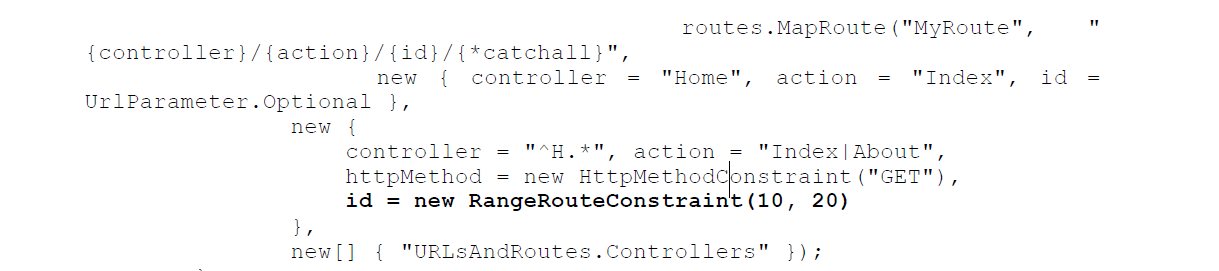
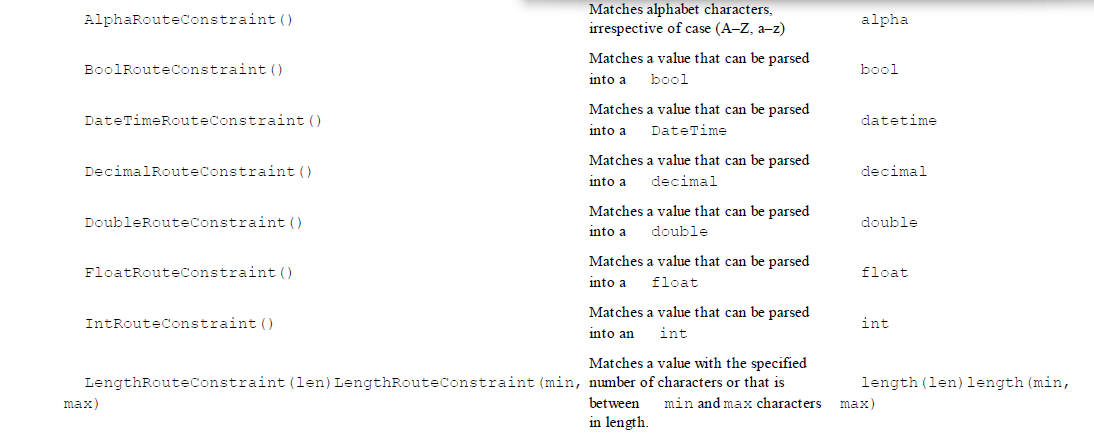


1. 限制Route，通过正则表达式来限制，在限制应用之前，先应用默认值



1. 限定使用指定的Http 方法

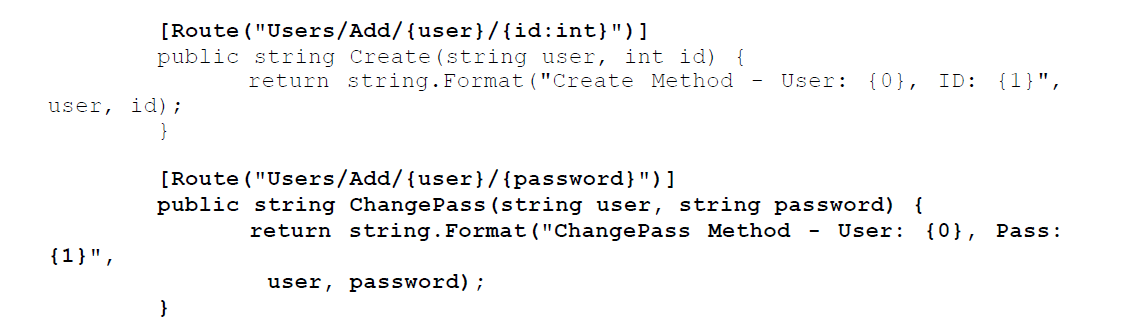


1. 有很多的预定义的限制类

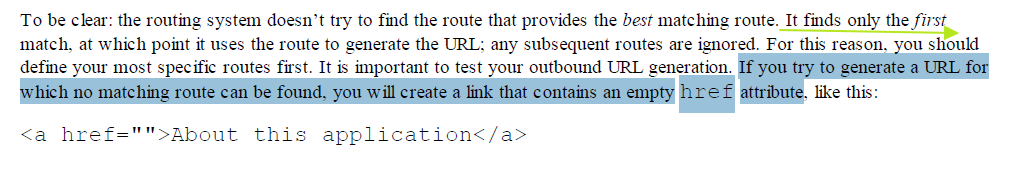
你也可以定义自己的限制类，只要实现IRouteConstraint接口就可以了。

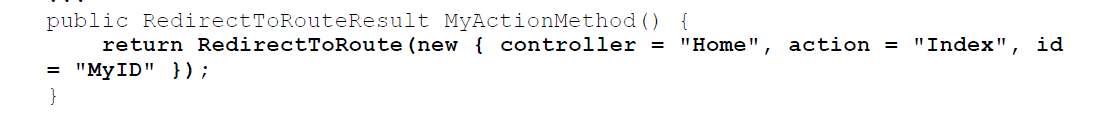
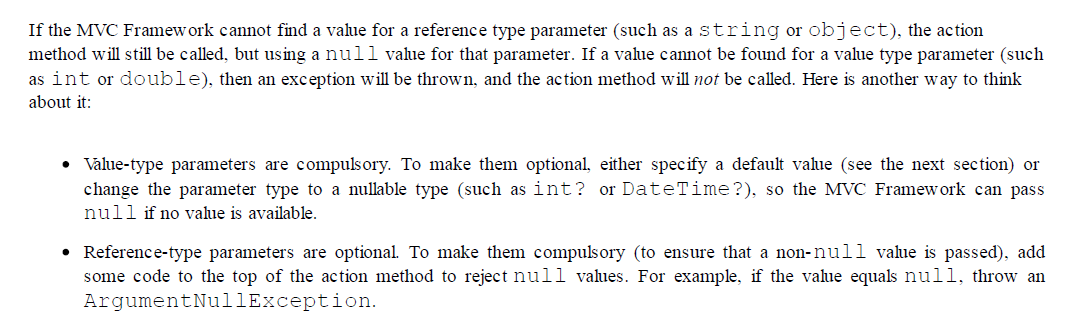
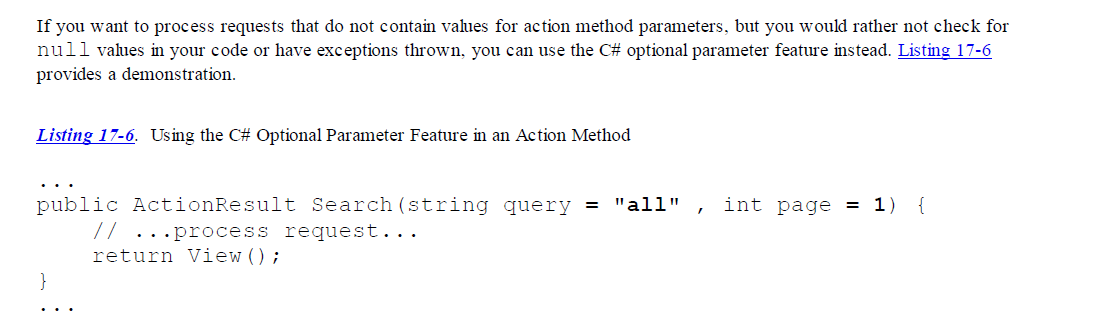
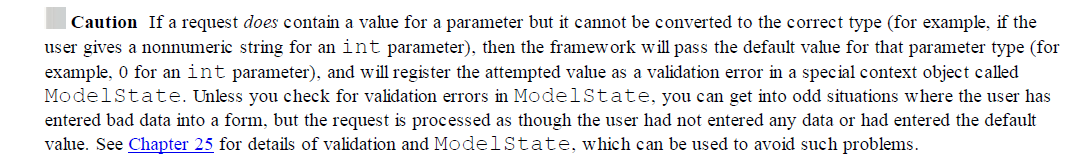
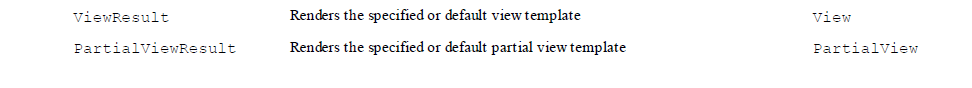
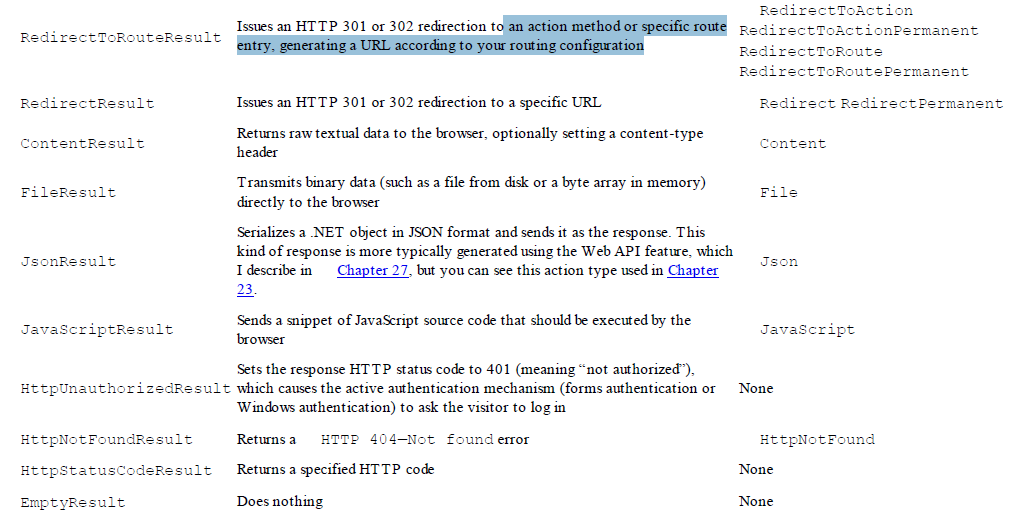
1. 使用Attribute Route，**routes.MapMvcAttributeRoutes();**

The Route attribute stops convention-based routes from targeting an action method even if attribute routing is disabled. Take care to call the MapMvcAttributeRoutes method in the RouteConfig.cs file or you will create unreachable action methods. You can apply the Route attribute to the same action method multiple times and each instance will create a new route



1. Html.ActionLink生成Url连接，Url.Action生成Url字符串

Url生成遵循以下条件：每个Segment必须有值，其值按如下顺序查找，按你提供的参数值，按当前请求值，按Route中的默认值。当前请求值重用，reuse values only for segment variables that occur earlier in the URL pattern rather than any parameters that are supplied to the Html.ActionLink method.

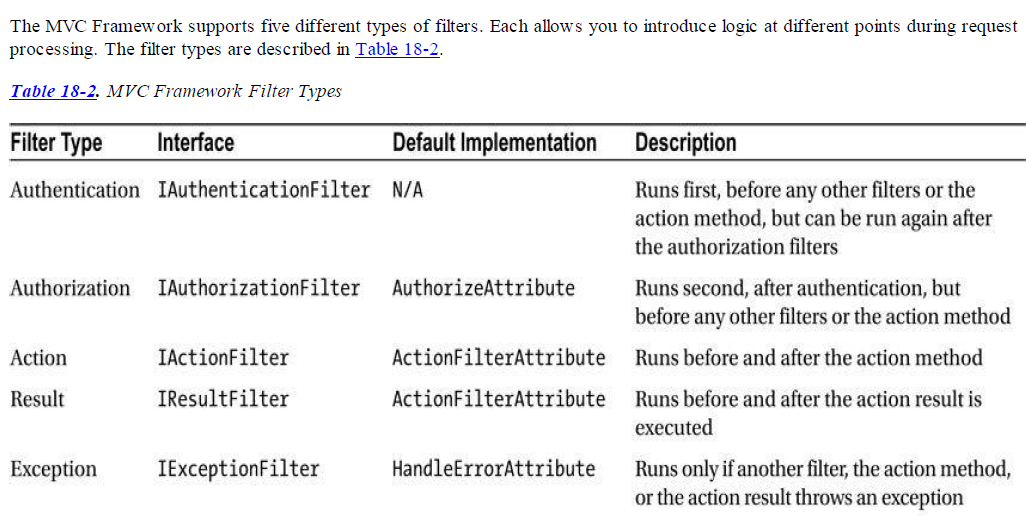
1. 
2. When an area is registered, any routes that it defines are limited to the namespace associated with the area.
3. **routes.RouteExistingFiles = true;**
4. 
5. 
6. 
7. 所有的Result类型都是ActionResult的子类。
8. 
9. 

A pattern called Post/Redirect/Get. In this pattern, you receive a POST request, process it, and then redirect the browser so that a GET request is made by the browser for another URL. GET requests should not modify the state of your application, so any inadvertent resubmissions of this request won’t cause any problems.

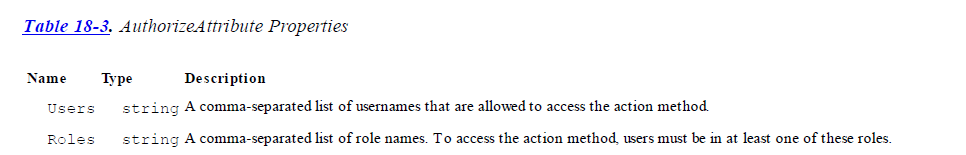
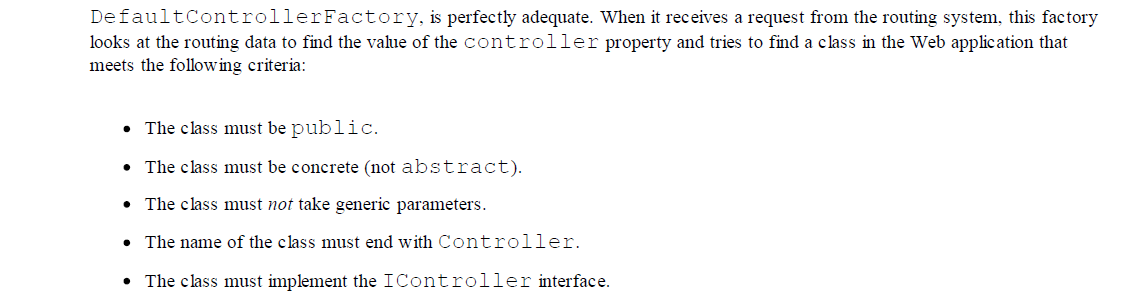
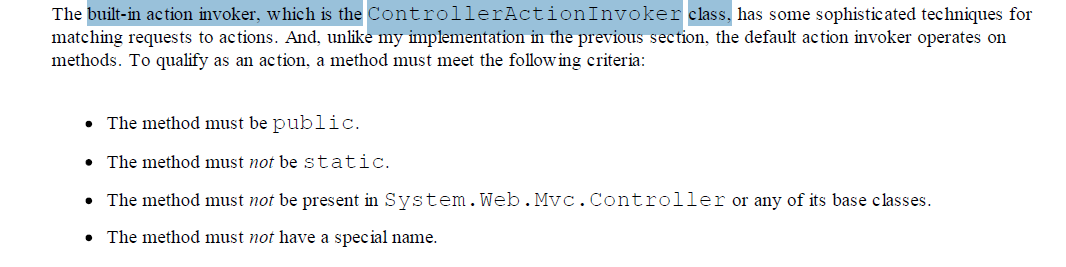
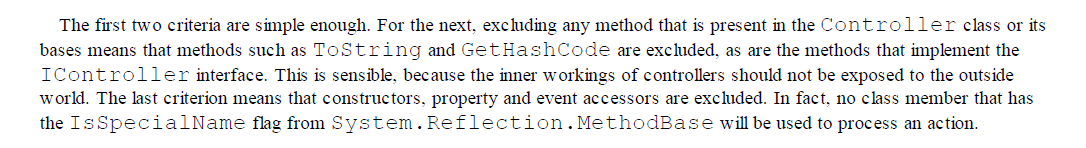
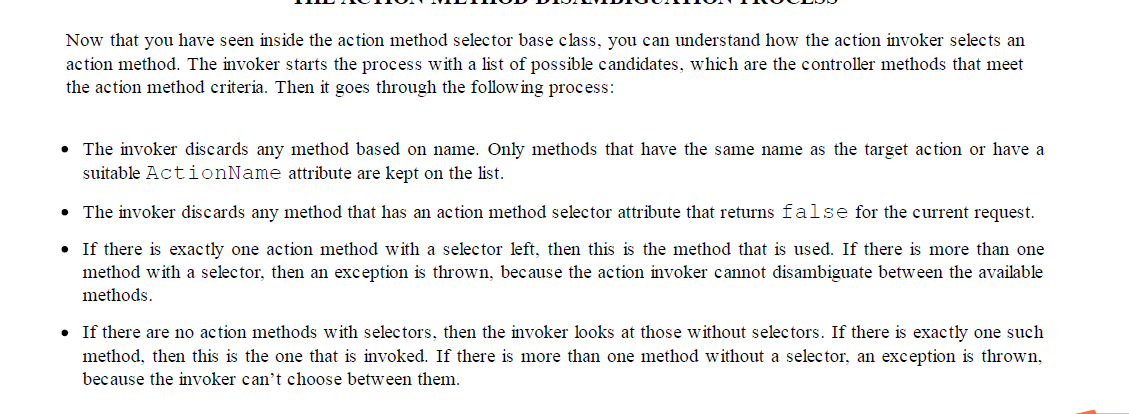
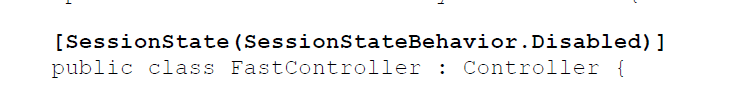
301，302 都是HTTP状态的编码，都代表着某个URL发生了转移，不同之处在于：   
301 redirect: 301 代表永久性转移(Permanently Moved)。  
302 redirect: 302 代表暂时性转移(Temporarily Moved )。   
这是很官方的说法，那么它们的区别到底是什么呢？

**二．现实中的差异  
2.1.对于用户**301，302对用户来说没有区别，他们看到效果只是一个跳转，浏览器中旧的URL变成了新的URL。页面跳到了这个新的url指向的地方。  
**2.2.对于引擎及站长  
2.2.1 302**302转向可能会有URL规范化及网址劫持的问题。可能被搜索引擎判为可疑转向，甚至认为是作弊。  
网址规范化  
请参见：<http://www.chinamyhosting.com/seoblog/2006/04/10/url-canonicalization/>  
网址劫持  
302重定向和网址劫持（URL hijacking）有什么关系呢？这要从搜索引擎如何处理302转向说起。从定义来说，从网址A做一个302重定向到网址B时，主机服务器的隐含意思是网址A随时有可能改主意，重新显示本身的内容或转向其他的地方。大部分的搜索引擎在大部分情况下，当收到302重定向时，一般只要去抓取目标网址就可以了，也就是说网址B。  
实际上如果搜索引擎在遇到302转向时，百分之百的都抓取目标网址B的话，就不用担心网址URL劫持了。问题就在于，有的时候搜索引擎，尤其是Google，并不能总是抓取目标网址。为什么呢？比如说，有的时候A网址很短，但是它做了一个302重定向到B网址，而B网址是一个很长的乱七八糟的URL网址，甚至还有可能包含一些问号之类的参数。很自然的，A网址更加用户友好，而B网址既难看，又不用户友好。这时Google很有可能会仍然显示网址A。  
由于搜索引擎排名算法只是程序而不是人，在遇到302重定向的时候，并不能像人一样的去准确判定哪一个网址更适当，这就造成了网址URL劫持的可能性。也就是说，一个不道德的人在他自己的网址A做一个302重定向到你的网址B，出于某种原因， Google搜索结果所显示的仍然是网址A，但是所用的网页内容却是你的网址B上的内容，这种情况就叫做网址URL劫持。你辛辛苦苦所写的内容就这样被别人偷走了。

**2.2.2 301**当网页A用301重定向转到网页B时，搜索引擎可以肯定网页A永久的改变位置，或者说实际上不存在了，搜索引擎就会把网页B当作唯一有效目标。  
301的好处是:  
第一， 没有网址规范化问题。  
第二， 也很重要的，网页A的PR网页级别会传到网页B。

1. TempData is similar to Session data, except that TempData values are marked for deletion when they are read,
2. 

If you have defined a custom base class for your controllers, any filters applied to the base class will affect the derived classes. Filters can be applied to individual action methods or to an entire controller.

1. 
2. 
3. 
4. 
5. 
6. 用来管理Controller的Session.