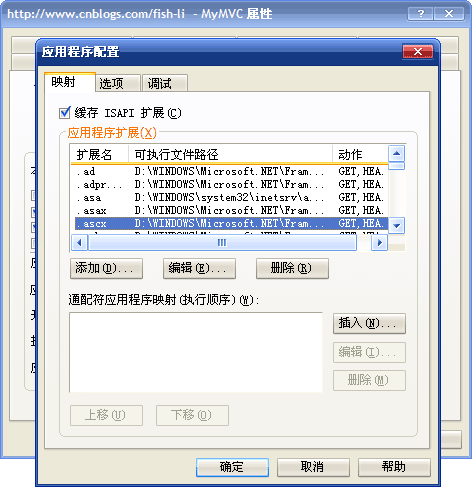
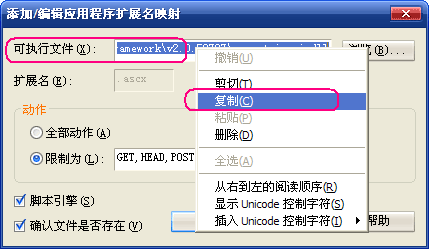
[回到顶部](http://www.cnblogs.com/fish-li/archive/2012/02/26/2368989.html" \l "_labelTop)

**IIS6 添加扩展名映射**

回到IIS，在网站节点上，右击，从弹出的菜单中，选择【属性】，然后在弹出的对话框中，选择【主目录】选项卡，并点击【配置】按钮。 此时的界面应该是这样的：

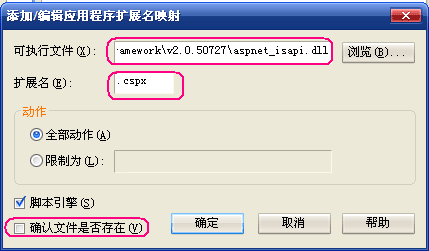


找到【ascx】这个扩展名，双击它，是不是弹出下面这个对话框？



照着图片的操作去完成：复制【可执行文件】的设置路径。**然后点击【取消】关闭对话框。**

此时将回到【应用程序配置】对话框，点击【添加...】按钮，



照着图片操作吧。对话框中的【可执行文件】的参数，此时已在Windows剪切板中，现在只要粘贴就可以了。

全部【确定】，关闭所有对话框，再回到浏览器，然后再试一次。

现在可以操作了吧？

先不要点击其它链接，还是来对刚才的操作做个小结吧。

在ASP.NET中，有时候我们可能会需要创建自己所需的HttpHandler来处理一些特殊的请求。  
我的MyMVC框架就有这个需求：将AJAX请求与页面的请求分开来处理。  
所以我们需要一些特殊格式的URL。通常选择一个没用过的扩展名会比较方便，因此我选择了 cspx  
为了能告诉ASP.NET将以下格式的ULR映射到AjaxHandlerFactory

/Fish.AA.AjaxTest/Add.cspx

/Fish.BB.AjaxTest.Add.cspx

/Fish/BB/AjaxTest/Add.cspx

/AjaxDemo/GetMd5.cspx

/AjaxDemo.GetMd5.cspx

我就需要在web.config中注册这种URL模式，并且为了能最好的匹配这些URL，我可以使用下面的配置：

<httpHandlers>

<add path="\*Ajax\*/\*.cspx,\*Ajax\*.\*.cspx" verb="\*"

type="MyMVC.AjaxHandlerFactory, MyMVC" validate="true"/>

</httpHandlers>

我们可以用Visual Studio自带的WebDev.WebServer.EXE来运行网站程序。  
本来，这一切都是很完美的。  
可是，IIS中并不支持这么复杂的path设置，它只能支持简单的扩展名映射。  
所以，我也只好使用简单的扩展名：【.cspx】来向IIS注册。

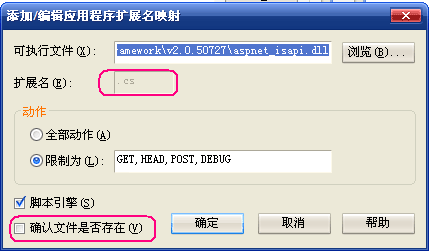
我再来解释一下，为什么在ASP.NET中，前面那个path能够识别我上面所说的5种格式的URL？  
在ASP.NET管线的处理器映射阶段，ASP.NET会将【\*Ajax\*/\*.cspx,\*Ajax\*.\*.cspx】转换成下面的正则表达式， 再来检查每个请求的URL是否匹配。

(?:\A|(?<=/))[^/]\*Ajax[^/]\*/[^/]\*\.cspx\z|(?:\A|(?<=/))[^/]\*Ajax[^/]\*\.[^/]\*\.cspx\z

有兴趣的话，您也可以检验一下，这个正则表达式与前面的URL都能匹配。  
从这里也可以看出ASP.NET对处理器的映射实现，是可以支持比较复杂的URL模式的。  
关于处理器的映射过程可以参考我的博客[【细说 HttpHandler 的映射过程】](http://www.cnblogs.com/fish-li/archive/2012/01/29/2331477.html" \t "_blank)

前面解释了我为什么最终选择【.cspx】来向IIS注册处理器了。  
在注册时，还有二个参数也比较重要：  
1. 可执行文件：其实就是一个实现了ISAPI的模块，IIS会将匹配的请求交给它，然后由它再交给ASP.NET。 这个参数的路径比较长，我们根本不需要记住它，只需要找个已有配置中，将它COPY出来就可以了。  
2. 确认文件是否存在：**这个参数一定不要选择。**因为我们请求的URL并没有对应的文件存在。

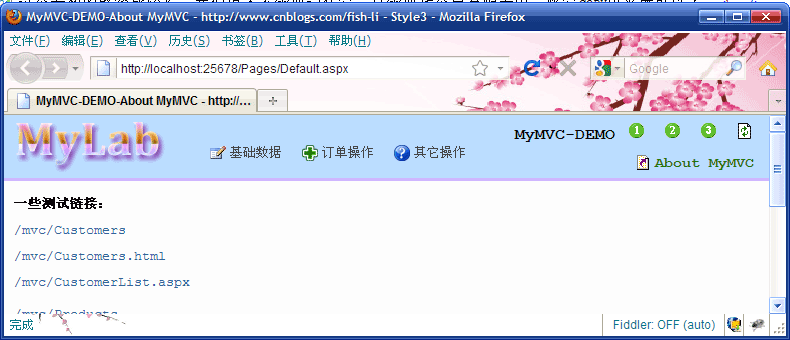
这里要补充一点：  
我以前发布的[FishWebLib DEMO](http://www.cnblogs.com/fish-li/archive/2011/05/02/2034010.html" \t "_blank)中， 使用了【cc】这样的扩展名，可以按照上面的方法注册。  
但我还使用了【cs】和【ascx】这二个扩展名。由于这二个扩展名的注册已经存在了， 所以，我需要修改它们的配置：双击配置项，确保不要勾选【确认文件是否存在】即可。



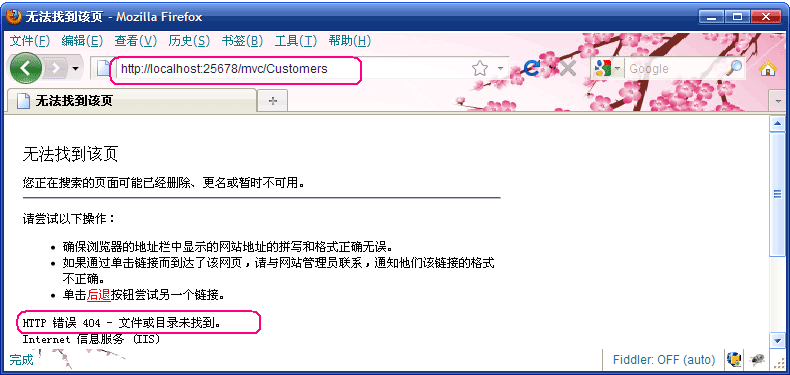
[回到顶部](http://www.cnblogs.com/fish-li/archive/2012/02/26/2368989.html#_labelTop)

**IIS6 无扩展名的映射**

让我们再回到前面已配置好的示例中，此时页面的显示应该是这个样子的：

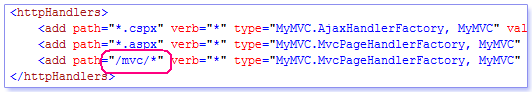


点击一下页面上的链接【/mvc/Customers】试试。

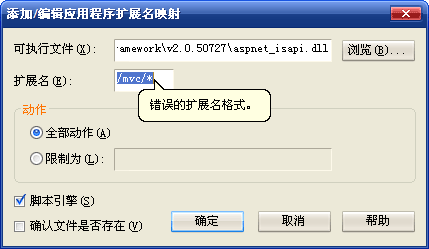


不要以为我是在故意设置陷阱哦。我在设计示例程序时，根本也没想到会这样。

到这里，可能有人会想，在httpHandlers中不是还有个【path="/mvc/\*"】没有配置吗？  
继续按照前面的方法去配置就能解决问题了。

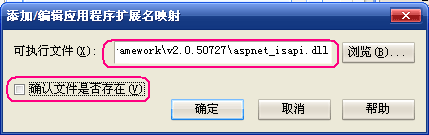


好吧，我再照着前面的方法再试着注册【path="/mvc/\*"】试试，结果如下：

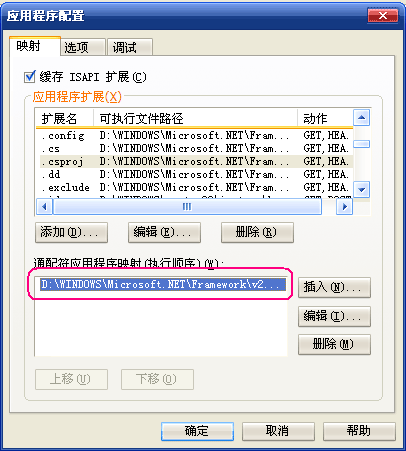


因此，前面的方法对于这类【无扩展名】的URL来说，**是无效的。**

对于这类无扩展名的URL，在IIS6中可以使用添加【通配符应用程序映射】的方法来解决。  
回到【应用程序配置】对话框，点击【插入...】按钮，弹出下面的对话框：



按照图片来设置一下吧。然后，【确定】关闭对话框。此时的设置应该是这样的：



全部点击【确定】关闭所有对话框。

让我们再次回到示例程序，此时可以发现，什么问题都没有了。

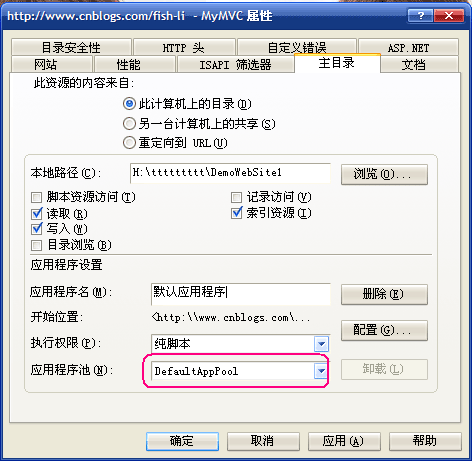
再补充一句：如果使用这种方法，前面注册cspx的过程就不需要了。因为此时所有的请求都会交给ASP.NET，而ASP.NET会识别我在web.config中所做的配置。

[回到顶部](http://www.cnblogs.com/fish-li/archive/2012/02/26/2368989.html#_labelTop)

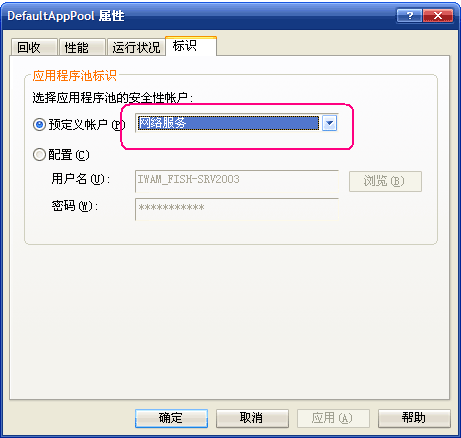
**目录的写入权限**

为了方便[MyMVC DEMO](http://www.cnblogs.com/fish-li/archive/2012/02/21/2361982.html)的部署过程， 这次我选择了XML文件做为数据源。写入XML的时机是在ASP.NET被停止运行的时候（Application\_End事件中）。

让一个在IIS中运行的网站停止运行的方法就是停止网站所使用的【应用程序池】。  
可以在网站属性对话框中找到网站所使用的【应用程序池】：



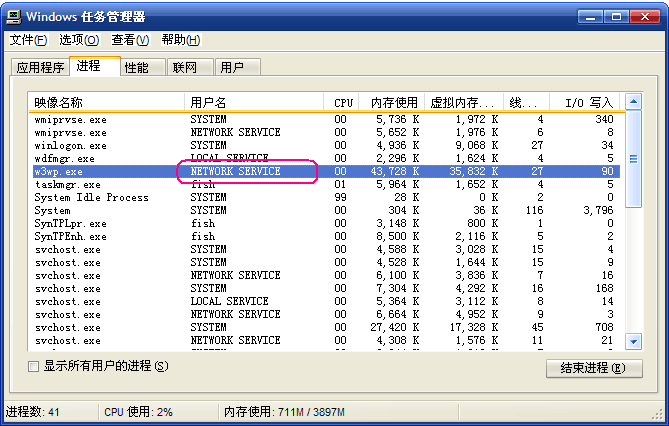
再切到IIS的【应用程序池】的列表，找到前面那个【应用程序池】，右击鼠标，  
从弹出的菜单中点击【属性】菜单，然后在出现的对话框中选择【标识】选项卡：



从这个图片中，我们可以知道网站以哪个Windows帐号在运行。记住这里，后面会用到。

好吧，**点击【取消】关闭对话框。**

补充一点：要想知道网站以什么帐号运行，还可以查看【Windows任务管理器】， 找到w3wp.exe所在进程即可：

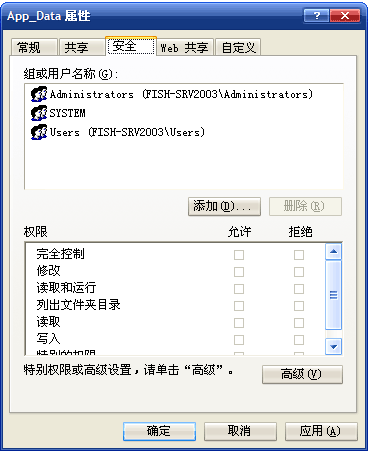


回到示例程序中，我们可以随便添加一些数据。  
然后，在【应用程序池】列表中，停止网站所使用的应用程序池。再启动它。  
重新刷新示例程序的页面。

发现什么了？是不是数据没有保存下来？

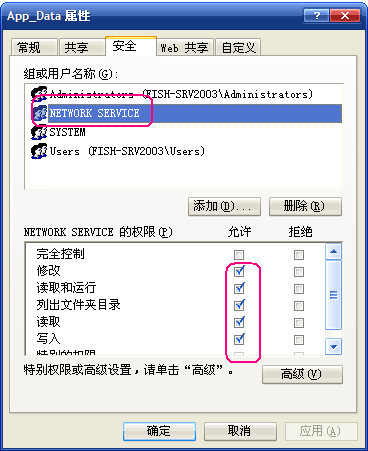
如果发现数据没有保存起来，可以继续阅读。

数据不能保存的原因并不是因为代码没有执行，而是因为，**网站运行的帐号没有权限写数据文件。**  
XML是放在网站的App\_Data目录中，为了检查网站是否有写入权限，可以在App\_Data目录上右击， 然后选择【属性】菜单，切换到【安全】选项卡：



经过前面的分析，我们已经知道网站是以【NETWORK SERVICE】帐号运行， 然而，在这个目录的安全设置中，并没有允许【NETWORK SERVICE】帐号能有写入权限， 所以，网站在停止运行时，是由于没有权限才导致不能保存数据的。

此时，我们可以赋予【NETWORK SERVICE】帐号对App\_Data目录有写入权限。 设置如下图：



好了，您可以再去重启网站所在的【应用程序池】，会发现现在数据能正常保存了。