[老老实实学WCF] 第三篇 在IIS中寄存服务

老老实实学WCF

第三篇 在IIS中寄宿服务

通过前两篇的学习，我们了解了如何搭建一个最简单的WCF通信模型，包括定义和实现服务协定、配置服务、寄宿服务、通过添加服务引用的方式配置客户端并访问服务。我们对WCF的编程生命周期有了一个最基本的了解。

在前两篇中演示的例子，一定要力求背着做下来，包括源程序、配置文件都要背着一行行的手写下来，这样才能有深刻的体会。WCF的知识零散复杂，必须扎扎实实的学习和练习。如果你还没有做到了然于胸，现在赶紧翻回去把例子再做一遍。

今天让我们稍微深入一点，了解一些关于寄宿的新知识：在IIS中寄宿服务。

在前两篇的例子中，我们建立了一个控制台应用程序来作为服务的宿主，这种寄宿方式叫做"自托管"，即WCF服务和应用程序是一体的。这种寄宿方式有一些优点，他需要最少的框架支持(只需要一个控制台应用程序就可以了，随处建立，随处运行)，因此配置和使用都是最简单的，此外通过控制台程序还可以对WCF服务运行中发生的错误进行监视，在开发服务阶段，这种方式能提供调试的便利。

然而，如果作为最终产品部署，自托管的寄宿方式就不那么合适，应用程序相比框架(IIS、Windows服务等)是不稳定的，WCF与应用程序共享生命周期，应用程序关闭后WCF也会停止。还有许多特性诸如进程回收、空闲关闭等自托管都是不支持的。因此，为了使我们的WCF符合产品级别的要求，应该为其选择一个更稳定、伸缩性更好的宿主。

除了自托管，WCF还可以寄宿于IIS、Windows服务、Windows进程激活服务(WAS)中。比较流行的是在IIS和Windows进程激活服务寄宿。

在IIS中寄宿，需要IIS5.1或更高版本的支持，IIS会为我们管理ServiceHost(还记得他吗，看第一篇中的代码)，同时为我们提供进程回收、空闲关闭、进程运行状况监视等特性支持，我们只需要把服务相关的文件按照一定的组织方法放入IIS的托管中(就像建立一个网站应用程序或虚拟目录)，IIS会为我们管理一切。这种托管受到支持的系统很多，从Windows XP SP2 到 WIndows Server 2008，所以它非常流行。然而他也有缺点，它只能接受http协议的绑定，对于tcp、管道、MSMQ都是不支持的。

从IIS7开始，提供了所谓Windows进程激活服务(WAS) 的功能，如果把WCF寄存在WAS中，就可以支持所有的绑定协议(TCP等)，像MSMQ这样的协议，在内网和.Net编程模型下有很大的性能优势，因此WAS应该会成为未来WCF寄宿的主要方式，但是IIS7要求Windows Vista以上版本的系统才能支持，他的普及可能尚需时日吧。

我们今天先学习在IIS中寄宿，记住，IIS寄宿只支持http协议的绑定。

实验环境在说明一下：
Windows 7 家庭高级版 SP1
IIS7
Visual Studio 2010 旗舰版 SP1

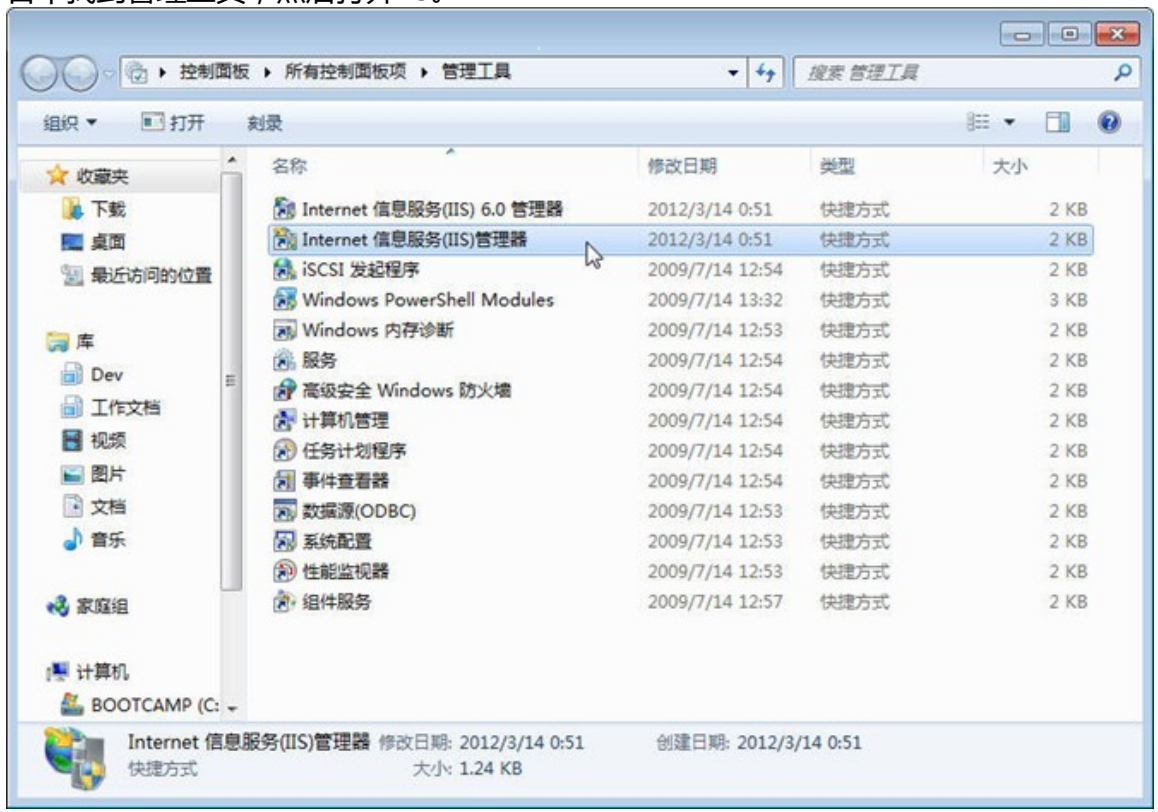
老老实实的学习，我们今天不借助IDE帮助建立的项目，完全手写一个寄宿于IIS的服务。

1. 为IIS应用建立物理位置

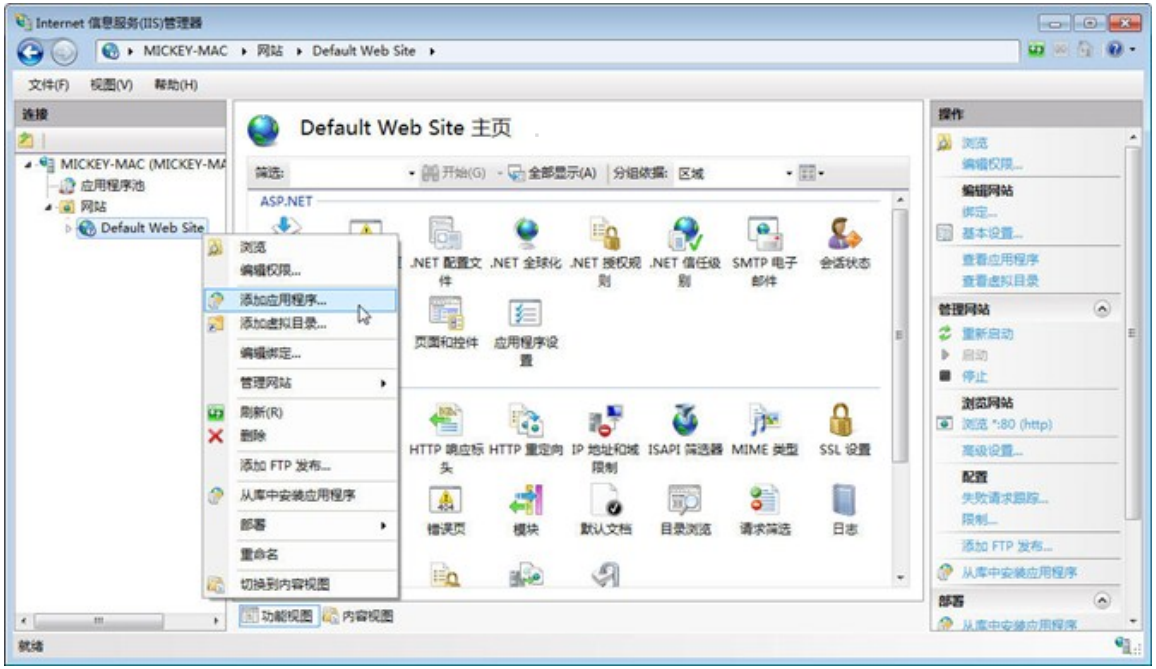
IIS应用程序需要映射到本地驱动器的一个物理路径上，我们先把它建好。
我把这个文件夹建立在了C:\WCF\下，取名为IISService。(HelloWCF是我们在前两篇中建立的，还记得么)

2. 建立IIS应用程序

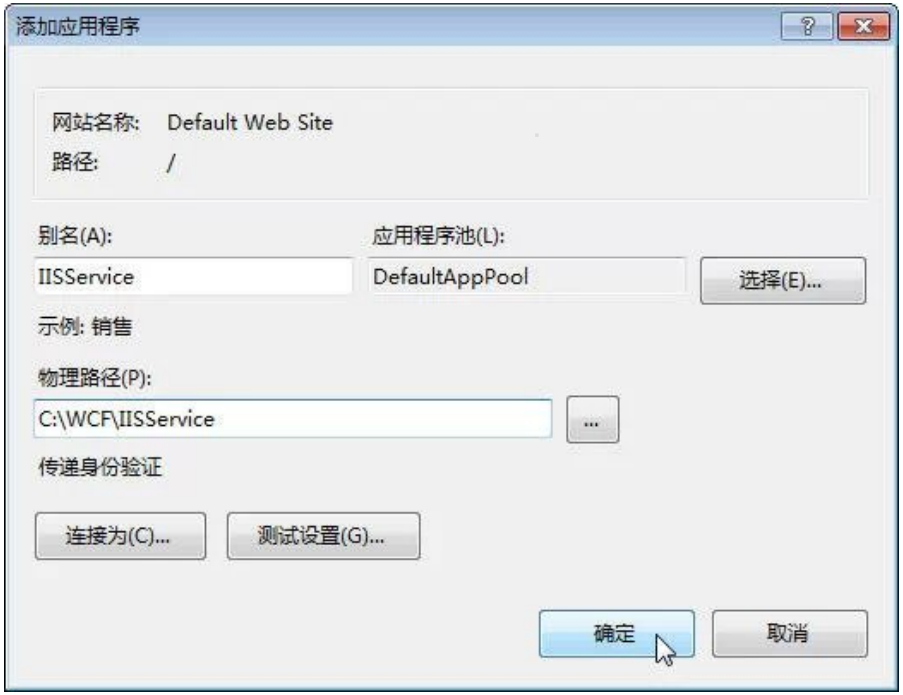
物理路径建好了，现在我们在这个位置上建立一个应用程序。点击开始->控制面板，在项目中找到管理工具，然后打开IIS。



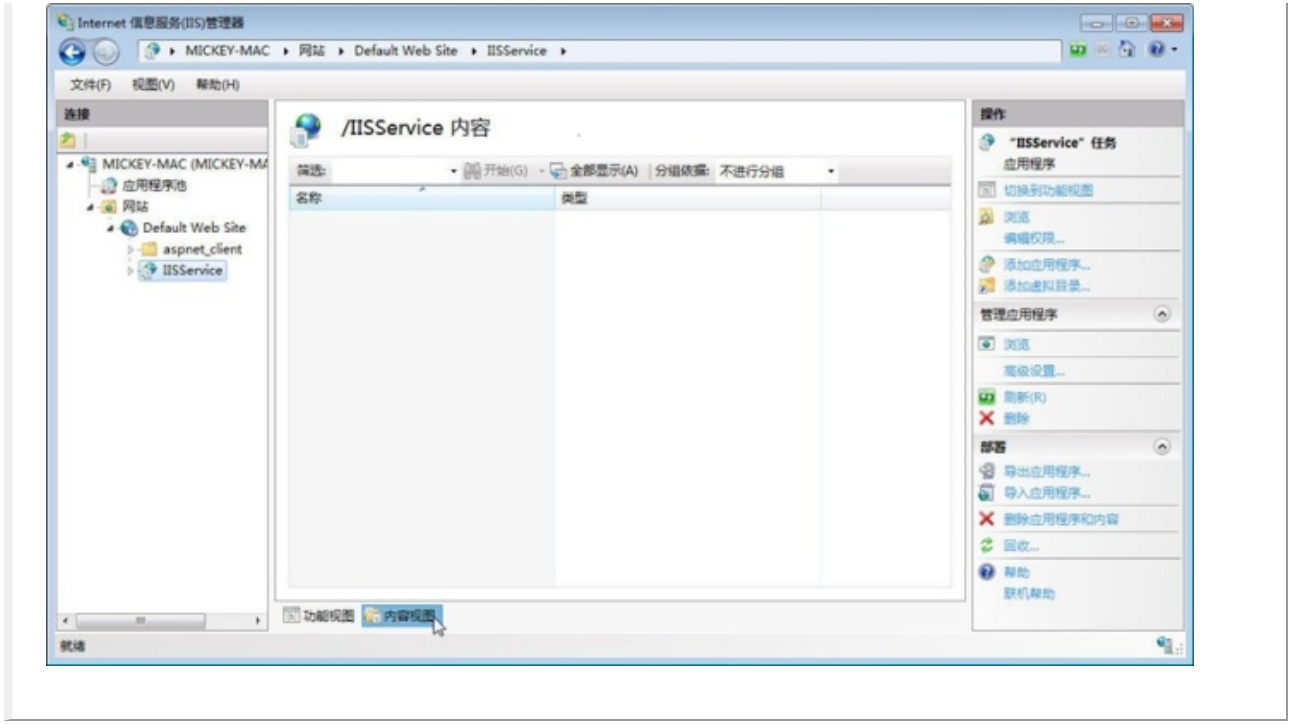
展开左边的节点，在默认网站节点上点击右键，选择“添加应用程序”



为应用程序指定一个别名，这个可以随意起的，这个名字将成为将来服务地址的一部分，我把它起作IISService，物理路径就选择我们刚才建立的文件夹。

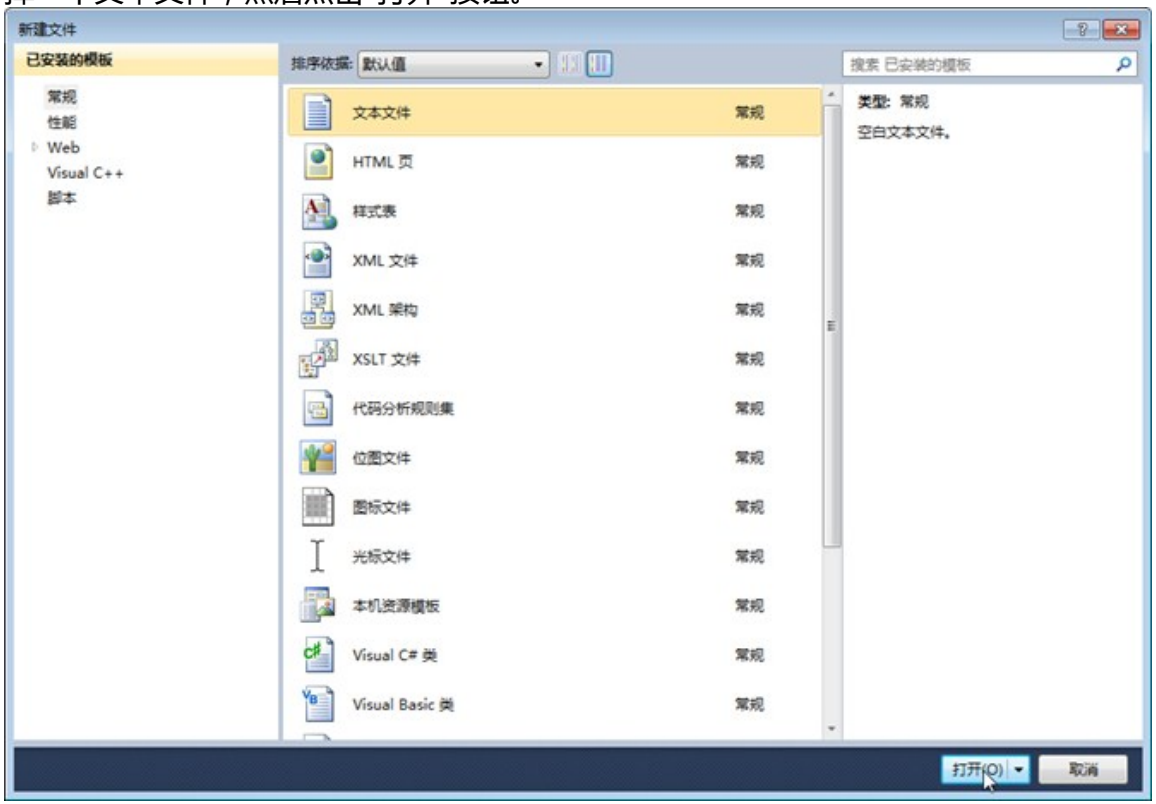


点击确定，IIS应用程序就建好了，我们可以看到在默认网站下多了这个应用程序，但是里面还什么都没有。

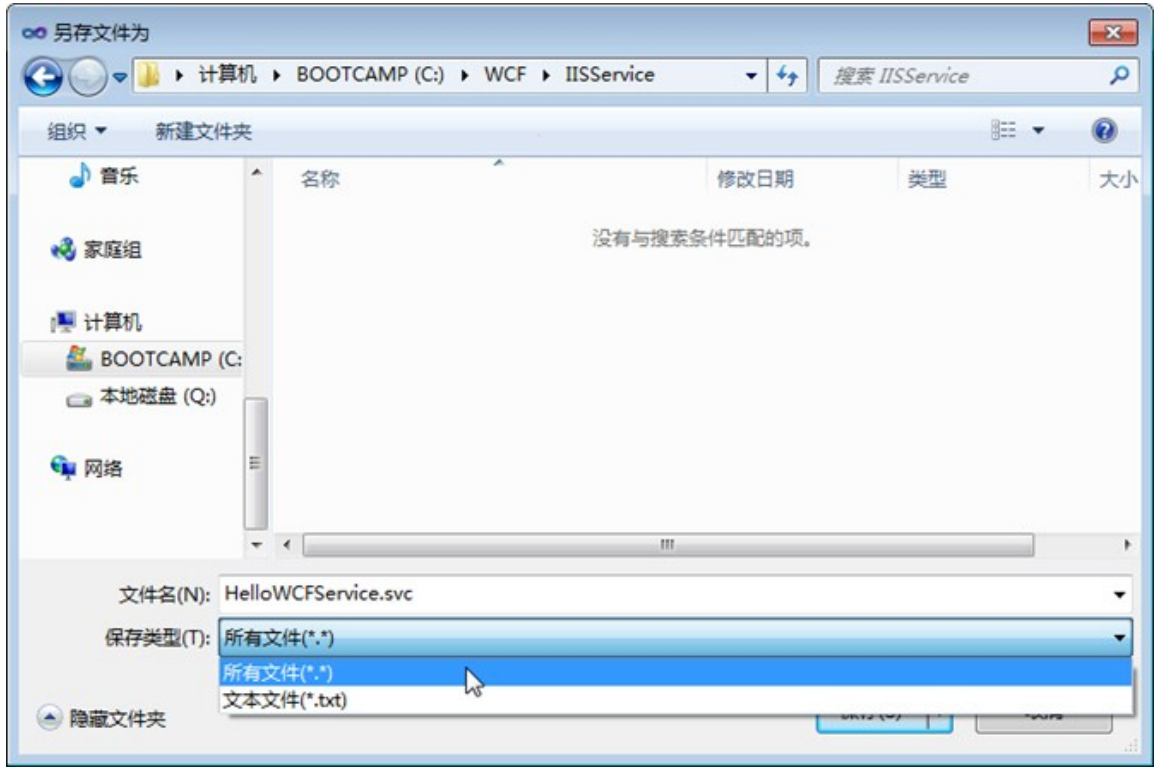

Internet 信息服务(IIS)管理器
MICKEY-MAC ▸ 网站 ▸ Default Web Site ▸ IISService ▸
文件(F) 视图(V) 帮助(H)
连接
MICKEY-MAC (MICKEY-MA
应用程序池
网站
Default Web Site
aspnet_client
IISService
/IISService 内容
筛选:
开始(G)
全部显示(A)
分组依据: 不进行分组
名称
类型
功能视图
内容视图
就绪
操作
"IISService" 任务
应用程序
切换到功能视图
浏览
编辑权限...
添加应用程序...
添加虚拟目录...
管理应用程序
浏览
高级设置...
刷新(R)
删除
部署
导出应用程序...
导入应用程序...
删除应用程序和内容
回收...
帮助
联机帮助

3. 建立服务文件

我们需要按照IIS宿主的要求建立几个文件放到IIS中才能承载起我们的服务，当然服务的相关信息也是描述在这些文件中的。
(1) svc文件。
svc就是service的意思了，我们需要首先建立一个XXX.svc的文件放到IIS应用程序目录下，这个文件是服务的入口，客户端需要这个文件的地址来访问服务(当然也包括原数据交换)。我们来手动建立这个文件，打开VS2010，选择文件菜单->新建->文件。在常规栏目中，选择一个文本文件，然后点击"打开"按钮。



这应该是个svc文件，而不是.txt文件，所以我们另存一下，另存的时候要注意保存类型选为所有文件。我把这个文件起名为HelloWCFService.svc，这个名字可以随意起，这个名字也将成为服务地址的一部分。保存位置就是我们刚刚建立IIS应用程序的位置。

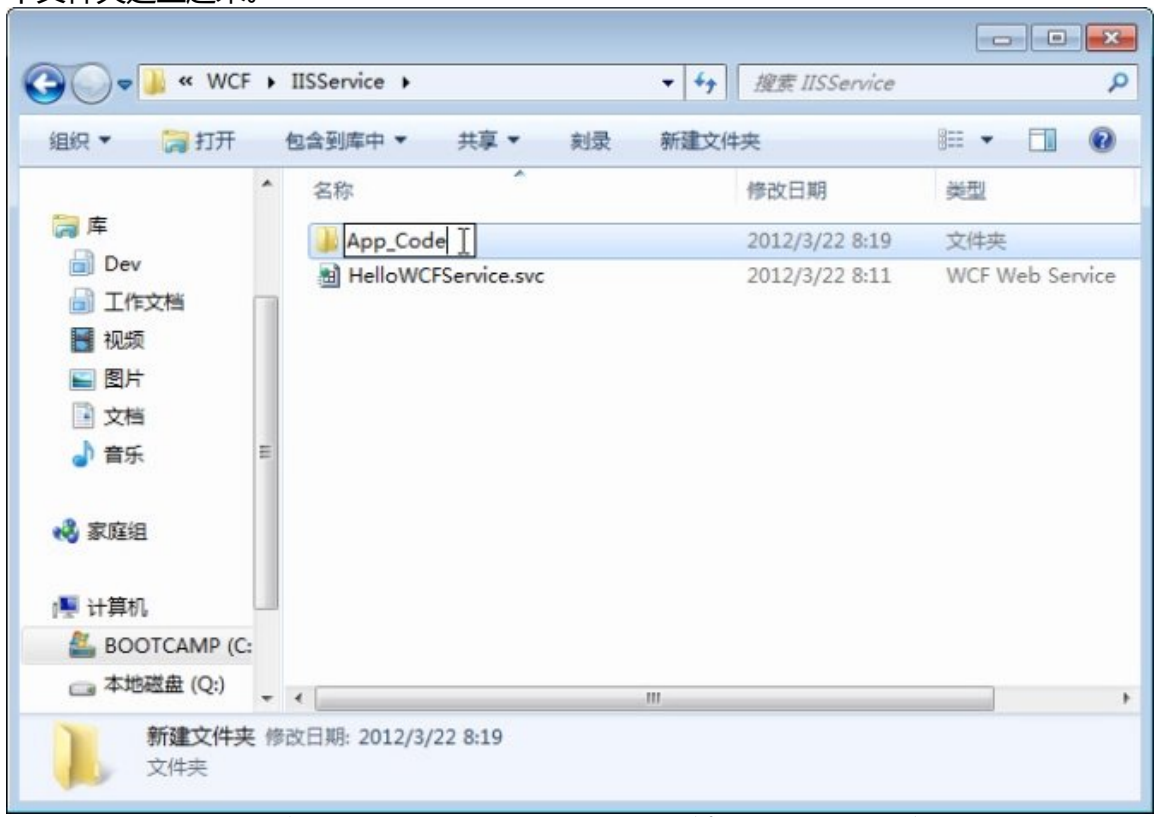


现在我们来编辑这个文件的内容，很简单，就只有一行代码。

```
[html]

1. <%@ServiceHost language=c# Debug="true" Service="LearnWCF.HelloWCFService"%>
```

被<%%>框住的表示这个是一个服务器端包含，@ServiceHost 标签表示这是个WCF的服务，联想一下前两篇代码中的ServiceHost 对象。language=c# 表示我们用C#语言来写代码，Debug=true顾名思义了，最主要的是Service这个属性，他表示这个服务的实现类是什么，这里要用完全限定名，即要包括命名空间。我起了一个命名空间名LearnWCF，我们把服务定义都放在这个命名空间下，后面的HelloWCFService就是服务的实现类了。我们接下来要去完善这个类的内容。
可以看出。.svc文件就相当于一个向导，帮我们在IIS宿主中找到服务的位置，具体的代码，我们可以写在另一个地方(其实也可以写在svc文件中，不推荐)。

把写的内容保存一下，我们继续前进。
接下来我们要写这个定义服务的类文件了。但是在这之前，我们先为类文件建立一个存放的位置，IIS的代码文件应该存放在IIS应用程序目录的App_Code子目录下，所以我们先把这个文件夹建立起来。



可以看到我们刚刚建立的HelloWCFService.svc已经被识别为WCF服务了。

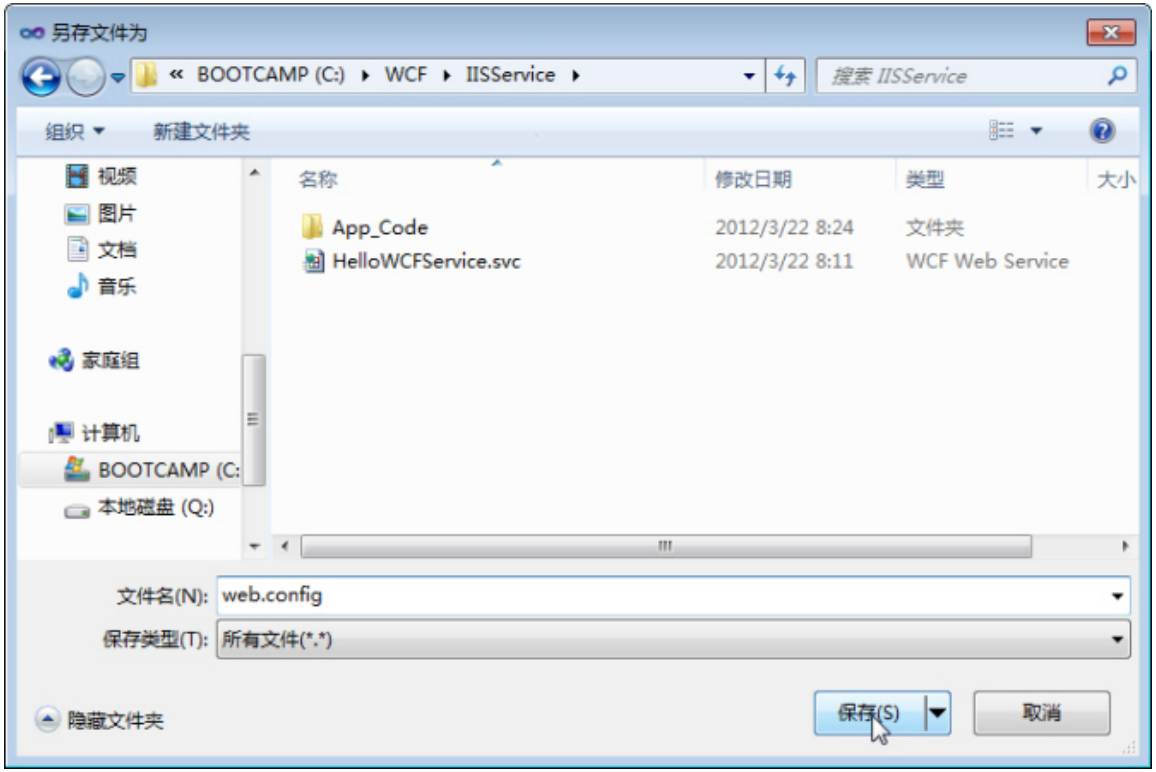
(2) cs文件
回到VS2010，还是文件->新建->文件，选择文本文件。
这次我们要建立的是类文件，其名字为HelloWCFService.cs，注意另存为的时候要把保存类型选为所有文件，路径要选择我们刚建立的App_Code文件夹
编写这个文件，我们在这里定义和实现服务协定，应该很熟悉吧，尝试着背着写下来吧。

```csharp
using System;
using System.ServiceModel;

namespace LearnWCF
{
    [ServiceContract]
    public interface IHelloWCF
    {
        [OperationContract]
        string HelloWCF();
    }

    public class HelloWCFService : IHelloWCF
    {
        public string HelloWCF()
        {
            return "Hello WCF!";
        }
    }
}
```

这个代码应该很熟练的打出来，如果对这段代码还有什么不理解的地方，赶快翻回第一篇复习一下。

保存一下，我们继续前进

(3)web.config文件
我们已经很清楚，还需要一个配置文件，在其中配置终结点、服务、行为等等的信息。这个配置文件和我们之前建立的大致相同。
还是回到VS2010，还是新建一个文本文件，另存为web.config。这个文件名，是不能改的，保存路径是我们建立的IIS应用程序IISService的目录下(和svc保存在一起)



先把他写下来，再做说明：

```
[html]

<configuration>
  <system.serviceModel>
    <services>
      <service name="LearnWCF.HelloWCFService" behaviorConfiguration="metadataExchange">
        <endpoint address="" binding="wsHttpBinding" contract="LearnWCF.IHelloWCF"/>
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="metadataExchange">
          <serviceMetadata httpGetEnabled="true"/>
        </behavior>
```

```
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

这个配置文件和我们之前写的有一些不同之处：
1) 配置文件的文件名为web.config，而不是app.config
2) <Service>标签没有了基地址的描述，在IIS寄宿中，服务基地址是由IIS负责指定的。例如本例中服务的基地址为

```
[html]

http://localhost/IISService/HelloWCFService.svc
```

3) 终结点的地址指定为了空，表示就使用服务基地址作为终结点地址，当然这里也可以指定一个相对地址，但是不能指定绝对地址，必须服从IIS指定的基地址。
其他地方并没有什么区别，特别注意在指定服务实现类和协定接口类的时候一定要带上命名空间，这是一个非常容易犯的错误。

保存，大功告成

4. 完成

到这里，在IIS中的寄宿就完成了，很简单，一个IIS应用程序，3个文件。当然这只是最简单的情况。
至于运营服务，IIS都会为我们去做，只要IIS应用程序(或网站)在线，服务就在线运行。
老办法，在浏览器里面看一下，是不是成功了。
IIS寄宿的服务地址格式：

```
[html]
1. http://机器名/IIS应用程序名/XXX.svc
```

所以我们这个例子的服务地址应该是：

```
[html]
1. http://localhost/IISService/HelloWCFService.svc
```

不出意外，情况如图

因为用IIS，所以系统自动给出了交换元数据的机器地址，而没有用localhost。

5. 总结。

这一篇我们学习了如何在IIS中寄宿WCF服务，必备的要素总结如下几点：
(1) 建立IIS应用程序及物理路径
(2) 在应用程序路径下建立XXX.svc文件用于声明WCF入口和服务地址导航
(3) 在应用程序路径的子目录App_Code下建立XXX.cs文件用于定义和实现服务协定
(4) 在应用程序路径下建立web.config 用于配置服务。
(5) 保持IIS为启动状态。