**第十一篇：REST调用**

上篇写的是Ajax调用WCF，今天写一篇如何以REST方式调用WCF服务。不知道REST是什么的同学，可以去google一下。对某些类型的应用，REST还是相当不错的方式，所以专门写一篇来说明一下开发方法。

老规矩，上代码，直接在代码注释里讲解。

**1、服务端：**

服务契约，我们定义CRUD4个方法（增查改删），对应HTTP METHOD分别为PUT/GET/POST/DELETE：

1. **using** System;
2. **using** System.ServiceModel;
3. **using** System.ServiceModel.Web;  //这个命名空间要求引入System.ServiceModel.Web.dll
5. **namespace** Server
6. {
7. [ServiceContract(Namespace = "WCF.Demo")]
8. **public** **interface** IData
9. {
10. //WebInvoke中标明REST的相关属性，以这个方法为例，调用的Url是 ..../Data/key/data，HTTP方法是PUT，响应为Json格式（也可以换成xml）
11. //这样如果客户端用PUT方法访问 ..../Data/1/100，就会映射到CreateData方法上来，并且传入key=1，data=100
12. [OperationContract]
13. [WebInvoke(UriTemplate = "Data/{key}/{data}", Method = "PUT", ResponseFormat = WebMessageFormat.Json)]
14. **void** CreateData(**string** key, **string** data);
16. [OperationContract]
17. [WebInvoke(UriTemplate = "Data/{key}", Method = "GET", ResponseFormat = WebMessageFormat.Json)]
18. **string** RetrieveData(**string** key);
20. [OperationContract]
21. [WebInvoke(UriTemplate = "Data/{key}/{data}", Method = "POST", ResponseFormat = WebMessageFormat.Json)]
22. **void** UpdateData(**string** key, **string** data);
24. [OperationContract]
25. [WebInvoke(UriTemplate = "Data/{key}", Method = "DELETE", ResponseFormat = WebMessageFormat.Json)]
26. **void** DeleteData(**string** key);
27. }
28. }

然后是实现类，这个简单，没什么可说的。

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.ServiceModel;
5. **namespace** Server
6. {
7. //这个例子中用了Single Instance模式，这样m\_DataDict的值才能保留住
8. [ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]
9. **public** **class** DataProvider : IData
10. {
11. **private** Dictionary<**string**, **string**> m\_DataDict = **new** Dictionary<**string**, **string**>();
13. **public** **void** CreateData(**string** key, **string** data)
14. {
15. m\_DataDict[key] = data;
16. }
18. **public** **string** RetrieveData(**string** key)
19. {
20. **return** m\_DataDict.ContainsKey(key) ? m\_DataDict[key] : "NOT FOUND";
21. }
23. **public** **void** UpdateData(**string** key, **string** data)
24. {
25. m\_DataDict[key] = data;
26. }
28. **public** **void** DeleteData(**string** key)
29. {
30. m\_DataDict.Remove(key);
31. }
32. }
33. }

配置文件最关键了，注意里面绿色的注释部分：

1. **<?xml** version="1.0" encoding="utf-8" **?>**
2. **<configuration>**
3. **<system.serviceModel>**
4. **<services>**
5. **<service** name="Server.DataProvider"**>**
6. <!--必须使用webHttpBinding，而且要定义此endpoint的behaviorConfiguration(见后)-->
7. **<endpoint** address="" binding="webHttpBinding" contract="Server.IData" behaviorConfiguration="restBehavior" **/>**
8. **<host>**
9. **<baseAddresses>**
10. **<add** baseAddress="http://localhost:8080/wcf" **/>**
11. **</baseAddresses>**
12. **</host>**
13. **</service>**
14. **</services>**
16. **<behaviors>**
17. <!--定义endpoint的behavior，webHttp节点表示启用web方式访问，这对REST是非常关键的-->
18. **<endpointBehaviors>**
19. **<behavior** name="restBehavior"**>**
20. **<webHttp/>**
21. **</behavior>**
22. **</endpointBehaviors>**
23. **</behaviors>**
24. **</system.serviceModel>**
25. **</configuration>**

最后发布服务，没什么特殊的，和以前一样：

1. **using** System;
2. **using** System.ServiceModel;
4. **namespace** Server
5. {
6. **class** Program
7. {
8. **static** **void** Main(**string**[] args)
9. {
10. **using**(ServiceHost host = **new** ServiceHost(**typeof**(Server.DataProvider)))
11. {
12. host.Open();
13. Console.WriteLine("Running ...");
14. Console.ReadKey();
15. host.Close();
16. }
17. }
18. }
19. }

这个服务端没有用IIS做HOST，直接用自己的进程做的宿主（当然了，本质还是http.sys在工作）。

**2、客户端**

我们这回要用REST形式访问服务端，所以不是普通意义上的WCF客户端了，再也用不着那么麻烦的写配置文件创建Channel或者代理了。

1. **using** System;
2. **using** System.Net;
4. **namespace** Client
5. {
6. **class** Program
7. {
8. **static** **void** Main(**string**[] args)
9. {
10. //用一个WebClient就可以搞定了
11. var client = **new** WebClient();
13. //以PUT方式访问Data/1/100，会映射到服务端的CreateData("1", "100")
14. client.UploadString("http://localhost:8080/wcf/Data/1/100", "PUT", **string**.Empty);
16. //以GET方式访问Data/1，会映射到服务端的RetrieveData("1")，应该返回"100"
17. Console.WriteLine(client.DownloadString("http://localhost:8080/wcf/Data/1"));
19. //以POST方式访问Data/1/200，会映射到服务端的UpdateData("1", "200")
20. client.UploadString("http://localhost:8080/wcf/Data/1/200", "POST", **string**.Empty);
22. //再GET一次，应该返回"200"
23. Console.WriteLine(client.DownloadString("http://localhost:8080/wcf/Data/1"));
25. //以DELETE方式访问Data/1，会映射到服务端的DeleteData("1")
26. client.UploadString("http://localhost:8080/wcf/Data/1", "DELETE", **string**.Empty);
28. //再GET一次，应该返回"NOT FOUND"
29. Console.WriteLine(client.DownloadString("http://localhost:8080/wcf/Data/1"));
30. }
31. }
32. }