



李林峰的园子

活到老,学到老,练到老.....

首页 管理 订阅 XML

公告



51Js 认证

昵称：李林峰的园子
园龄：3年10个月
粉丝：1799
关注：37
+加关注

随笔分类(64)


- C#教程之自己动手写映射(7)
- JS触屏教程(2)
- 常用工具类(1)
- 创业之路(1)
- 归那总节(2)
- 生命感悟(1)
- 无废话ExtJs系列教程(24)
- 无废话MVC系列教程(13)
- 无废话SharePoint系列教程(5)
- 无废话WCF系列教程(6)
- 无废话软件设计(2)

阅读排行榜

- 1. 无废话WCF入门教程一[什么是WCF](131798)
- 2. MVC入门教程(60771)
- 3. 无废话MVC入门教程二[第一个小Demo](42570)
- 4. 无废话ExtJs 入门教程一[学习方法](37553)
- 5. ExtJs 入门教程(37498)

Copyright ©2016 李林峰的园子

无废话WCF入门教程六[一个简单的Demo]

wcf技术交流，同学习共进步，欢迎加群： 群号:274746316

一、前言

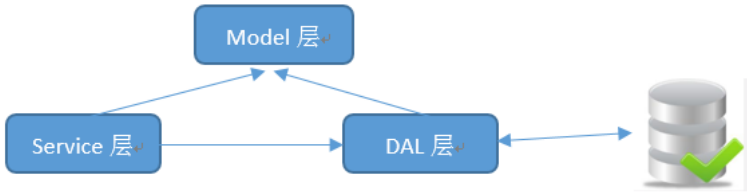
前面的几个章节介绍了很多理论基础，如：什么是WCF、WCF中的A、B、C。WCF的传输模式。本文从零开始和大家一起写一个小的WCF应用程序Demo。

大多框架的学习都是从增、删、改、查开始来学习的，我们学习WCF也是一样的。从简单来看(不包括安全、优化等相关问题)，WCF的增删改查和WebForm相差无几。WCF只是把具体“实现”写在“Service端”，而“调用”放在了“Client端”。觉得有帮助别忘了点个赞哈，谢谢哦~

二、Demo说明

- 1) Demo的“Service端”以本机IIS为宿主，“Client端”以WebForm项目为例。
- 2) Demo的“Service端”提取数据采用初学者比较容易接受的分层结构进行搭建，分别分为服务层、实体层、数据层。

引用关系如下图所示：

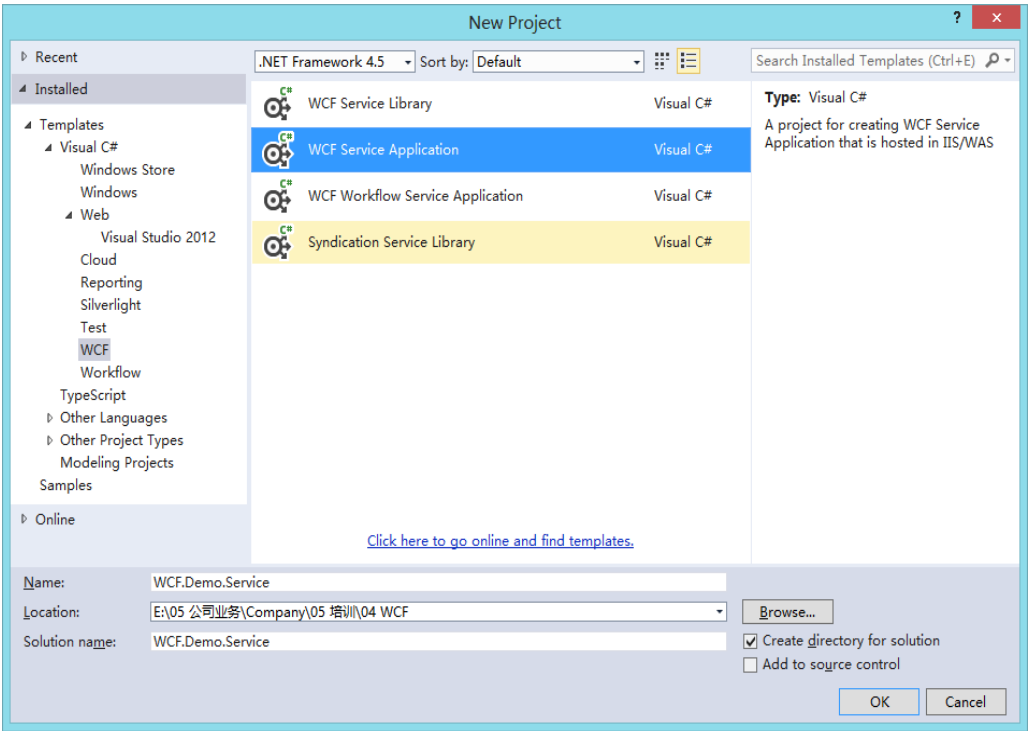


- 3) Demo以数据库为SqlServer，表User为例(sql语句在下载的压缩包中Init.sql)，表结构如下所示：

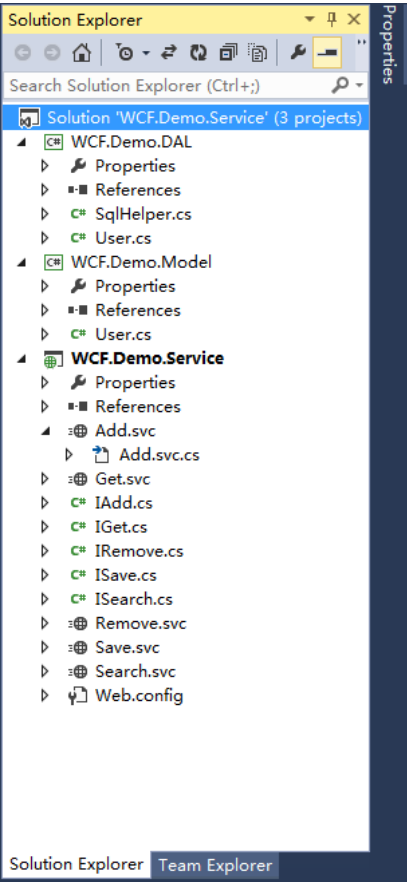
字段名	列名	数据类型	约束	生成方式
用户编号	UserID	int	主键，必须输入	自动增+1
姓名	Name	varchar(200)	非必须输入	人工输入
密码	Password	varchar(200)	非必须输入	人工输入
描述	Discribe	varchar(800)	非必须输入	人工输入
提交时间	SubmitTime	datetime	非必须输入	人工输入

三、创建Service端宿主

- 1) 创建WCF应用程序命名为：WCF.Demo.Service，如下图：



2) 删除默认文件IService.cs与服务.svc。并分别创建增、删、改、查“Add.svc”、“Save.svc”、“Remove.svc”、“Get.svc”、“Search.svc”，分别对应4个功能的服务应用程序WCF服务应用程序，并创建数据操作层和数据实体层，如下图：



3) 增加实体层和数据操作层代码，如下所示：

```
1 //用户实体
2 [DataContract]
3 public class User
4 {
5     [DataMember]
6     public int UserID { get; set; }
7     [DataMember]
8     public string UserName { get; set; }
9     [DataMember]
10    public string Password { get; set; }
11    [DataMember]
```

```

12     public string Discribe { get; set; }
13     [DataMember]
14     public DateTime SubmitTime { get; set; }
15 }
16 //数据操作, 调用SqlHeler
17 public class User
18 {
19     private static readonly string connectionString =
"server=.;database=wcfDemo;uid=sa;pwd=123456;";
20
21     //添加
22     public static bool Add(Model.User user)
23     {
24         string sql = string.Format("INSERT INTO [dbo].[User] ([UserName],[Password],[Discribe],
[SubmitTime]) VALUES('{0}','{1}','{2}','{3}')" , user.UserName, user.Password, user.Discribe,
user.SubmitTime);
25         int result = SqlHelper.ExecuteNonQuery(connectionString, CommandType.Text, sql);
26         if (result > 0)
27             return true;
28         else
29             return false;
30     }
31
32     //修改
33     public static bool Save(Model.User user)
34     {
35         string sql = string.Format("UPDATE [dbo].[User] SET [UserName] = '{0}',[Discribe] =
'{2}',[SubmitTime] = '{3}' WHERE UserID = {4}", user.UserName, user.Password, user.Discribe,
user.SubmitTime, user.UserID);
36         int result = SqlHelper.ExecuteNonQuery(connectionString, CommandType.Text, sql);
37         if (result > 0)
38             return true;
39         else
40             return false;
41     }
42
43     //删除
44     public static bool Remove(int UserID)
45     {
46         string sql = string.Format("DELETE FROM [dbo].[User] WHERE UserID = {0}", UserID);
47         int result = SqlHelper.ExecuteNonQuery(connectionString, CommandType.Text, sql);
48         if (result > 0)
49             return true;
50         else
51             return false;
52     }
53
54     //获取用户
55     public static Model.User Get(int UserID)
56     {
57         Model.User user = new Model.User();
58         string sql = string.Format("SELECT * FROM [dbo].[User] WHERE UserID = {0}", UserID);
59         DataSet ds = SqlHelper.ExecuteDataset(connectionString, CommandType.Text, sql);
60         if (ds != null && ds.Tables.Count > 0)
61         {
62             foreach (DataRow dr in ds.Tables[0].Rows)
63             {
64                 user.UserID = Convert.ToInt32(dr["UserID"]);
65                 user.UserName = dr["UserName"].ToString();
66                 user.Password = dr["Password"].ToString();
67                 user.Discribe = dr["Discribe"].ToString();
68                 user.SubmitTime = Convert.ToDateTime(dr["SubmitTime"]);
69             }
70         }
71         return user;
72     }
73
74     //获取用户列表
75     public static List<Model.User> GetUsers()
76     {
77         List<Model.User> Users = new List<Model.User>();
78         string sql = string.Format("SELECT * FROM [dbo].[User]");
79         DataSet ds = SqlHelper.ExecuteDataset(connectionString, CommandType.Text, sql);
80         if (ds != null && ds.Tables.Count > 0)
81         {
82             foreach (DataTable dt in ds.Tables)
83             {
84                 foreach (DataRow dr in dt.Rows)
85                 {
86                     Model.User user = new Model.User();
87                     user.UserID = Convert.ToInt32(dr["UserID"]);
88                     user.UserName = dr["UserName"].ToString();
89                     user.Password = dr["Password"].ToString();
90                     user.Discribe = dr["Discribe"].ToString();
91                     user.SubmitTime = Convert.ToDateTime(dr["SubmitTime"]);
92                     Users.Add(user);
93                 }
86

```

```
94         }
95     }
96     return Users;
97 }
98 }
```

4) 修改Add接口的代码和实现，如下所示：

```
1  [ServiceContract]
2  public interface IAdd
3  {
4      [OperationContract]
5      bool DoWork(Model.User user);
6  }
7
8  public class Add : IAdd
9  {
10     public bool DoWork(Model.User user)
11     {
12         return DAL.User.Add(user);
13     }
14 }
```

5) 修改Save接口的代码和实现，如下所示：

```
1  [ServiceContract]
2  public interface ISave
3  {
4      [OperationContract]
5      bool DoWork(Model.User user);
6  }
7
8  public class Save : ISave
9  {
10     public bool DoWork(Model.User user)
11     {
12         return DAL.User.Save(user);
13     }
14 }
```

6) 修改Remove接口的代码和实现，如下所示：

```
1  [ServiceContract]
2  public interface IRemove
3  {
4      [OperationContract]
5      bool DoWork(int UserID);
6  }
7
8  public class Remove : IRemove
9  {
10     public bool DoWork(int UserID)
11     {
12         return DAL.User.Remove(UserID);
13     }
14 }
```

7) 修改Search接口的代码和实现，如下所示：

```
1  [ServiceContract]
2  public interface ISearch
3  {
4      [OperationContract]
5      List<Model.User> DoWork();
6  }
7
8  public class Search : ISearch
9  {
10     List<Model.User> ISearch.DoWork()
11     {
12         return DAL.User.GetUsers();
13     }
14 }
```

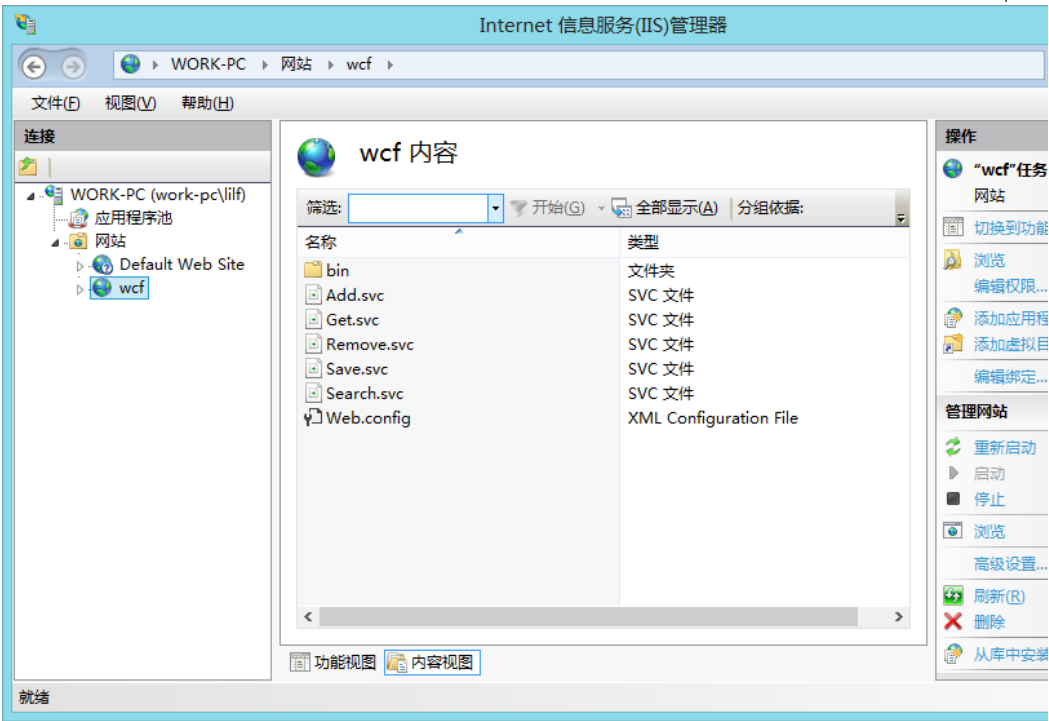


8) 修改Get接口的代码和实现，如下所示：

```
1  [ServiceContract]
2  public interface IGet
3  {
4      [OperationContract]
5      Model.User DoWork(int UserID);
6  }
7
8  public class Get : IGet
9  {
10     public Model.User DoWork(int UserID)
11     {
12         return DAL.User.Get(UserID);
13     }
14 }
```

四、部署服务端程序

1) 将程序发布，并部署到IIS上，设置端口：8080，如下图所示：

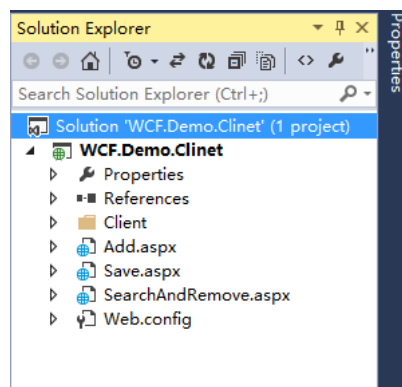


2) 以Add.svc服务应用程序为目标，测试部署是否成功，成功后如下图所示：



五、创建Client端Web应用程序

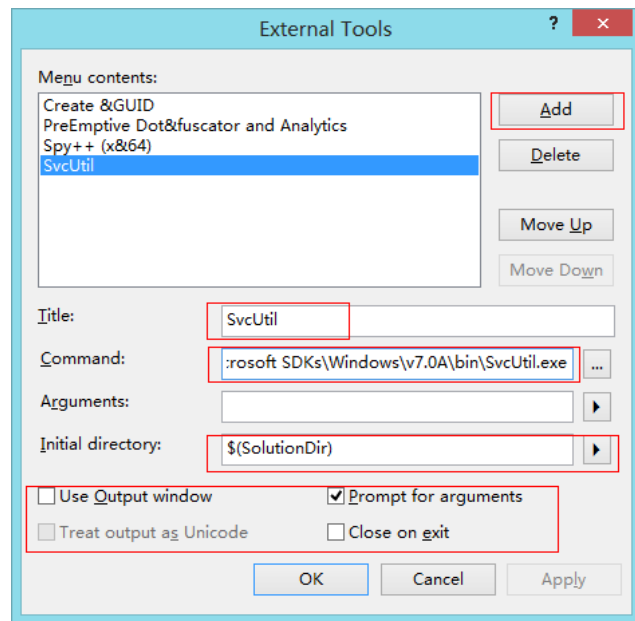
新建WebForm项目WCF.Demo.Client，并创建增删改查文件，Add.aspx，Save.aspx，SearchAndRemove.aspx。如下图所示：



六、使用SvcUtil.exe生成客户端代码和配置

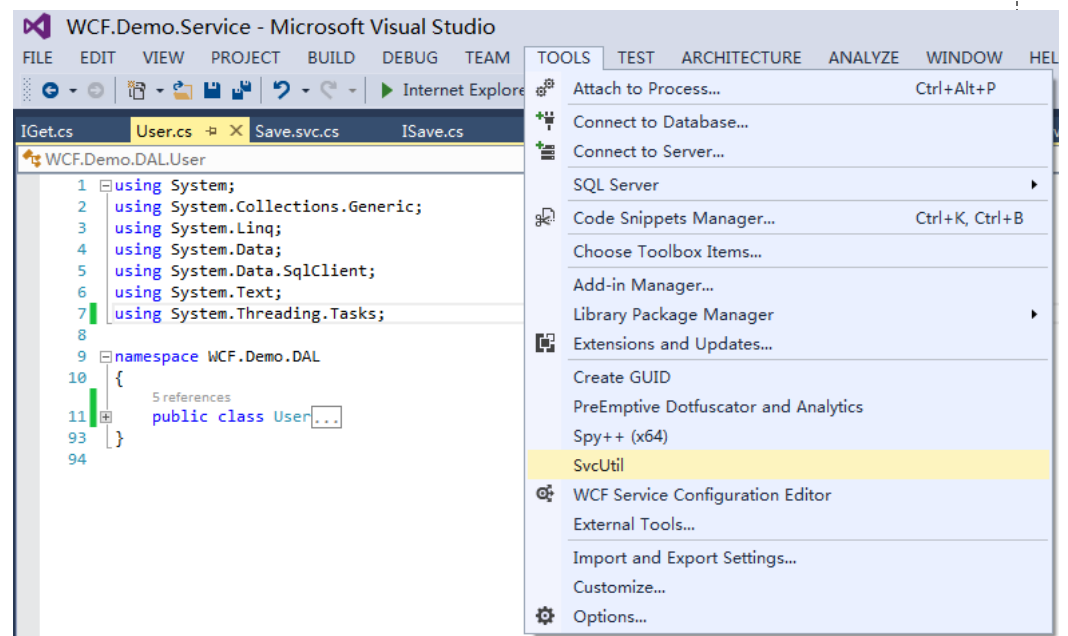
SvcUtil.exe是一个VS命令行工具，该工具位于：C:\Program Files\Microsoft SDKs\Windows\v7.0\bin 或 C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0\bin\一般情况下我们将SvcUtil.exe添加到VS开发工具中方便以后的运用（也可直接使用该命令行工具）。

1) 在VS中的 Tools菜单---选择External Tools，打开管理窗口如下图所示：

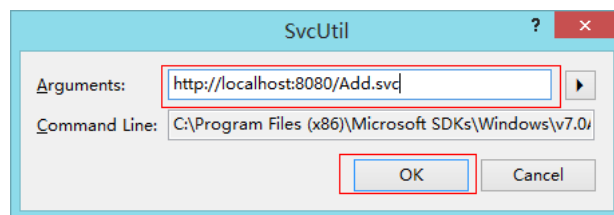


在Title中输入SvcUtil，Command中选择SvcUtil.exe全路径，Initial directory栏选择生成的客户端代码和配置文件所放的目录（此处为解决方案所在目录），选上Prompt for arguments，不选上Close on exit。点击OK。

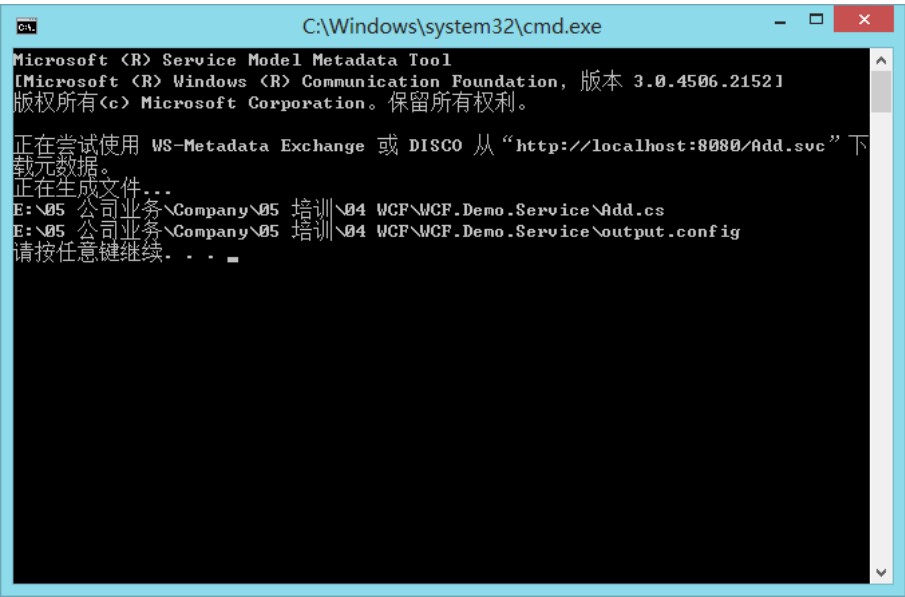
2) 添加完成后，在VS的工具下会出现这个菜单。如下图所示：



3) 在Client端添加对服务的引用。打开SvcUtil工具，在Arguments里填写服务的地址，如下图：



点击OK后出现下图，说明代码类和配置文件生成成功（在解决方案目标下），如下图所示：

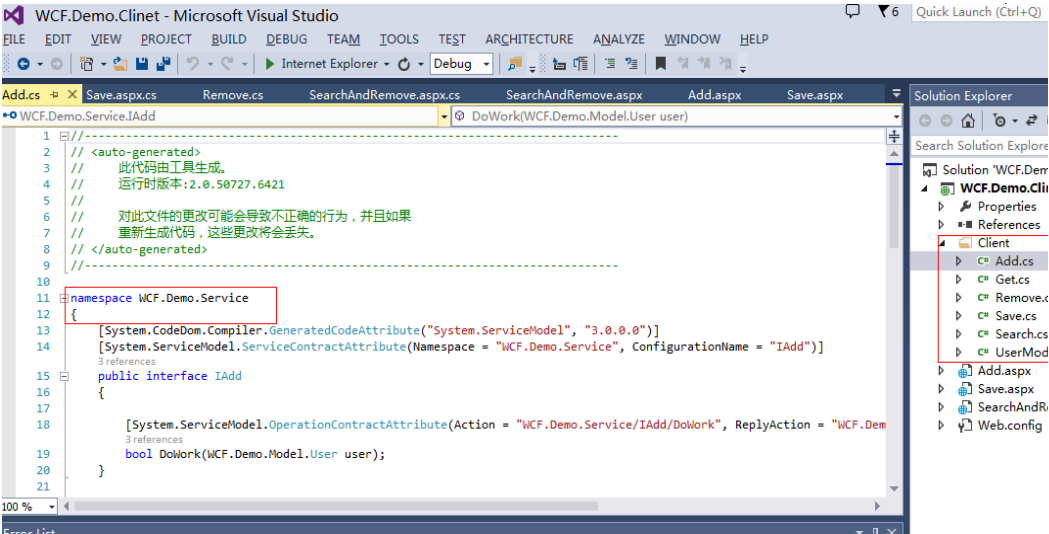


此时代理类和配置文件被下载到解决方案的物理目录中，如下图所示：

WCF.Demo.DAL	2014/11/9 21:45	文件夹	
WCF.Demo.Model	2014/11/9 19:23	文件夹	
WCF.Demo.Service	2014/11/9 20:33	文件夹	
Add.cs	2014/11/9 21:59	Visual C# Sourc...	5 KB
output.config	2014/11/9 21:59	XML Configurati...	2 KB
WCF.Demo.Service.sln	2014/11/9 20:22	Microsoft Visual...	2 KB

七、在Client端使用代理类与配置

将代理类从服务端的物理目录拷贝出来，放到Client端，并适当的修改代码，加入自己需要的名称空间，如下图所示：



使用代码如下所示：

```
1 //增加
2 public partial class Add : System.Web.UI.Page
3 {
4     Service.AddClient addClient = new Service.AddClient();
5     protected void Page_Load(object sender, EventArgs e)
6     {
7     }
8 }
9
10 //提交
11 protected void btnSubmit_Click(object sender, EventArgs e)
12 {
13     Model.User user = new Model.User();
14     user.UserName = this.txtUserName.Text;
15     user.Password = this.txtPassword.Text;
16     user.Discribe = this.txtDiscribe.Text;
17     user.SubmitTime = System.DateTime.Now;
18     addClient.DoWork(user);
19     Response.Write("添加成功!");
20 }
```



```

21     }
22     //修改
23     public partial class Save : System.Web.UI.Page
24     {
25         Service.SaveClient saveClient = new Service.SaveClient();
26         Service.GetClient getClient = new Service.GetClient();
27         protected void Page_Load(object sender, EventArgs e)
28         {
29             if (!Page.IsPostBack && !string.IsNullOrEmpty(Request.QueryString["UserID"]))
30             {
31                 GetUser();
32             }
33         }
34
35         protected void GetUser()
36         {
37             int UserID = Convert.ToInt32(Request.QueryString["UserID"]);
38             Model.User user = getClient.DoWork(UserID);
39             this.txtUserName.Text = user.UserName;
40             this.txtDiscribe.Text = user.Discribe;
41         }
42
43         //提交
44         protected void btnSubmit_Click(object sender, EventArgs e)
45         {
46             int UserID = Convert.ToInt32(Request.QueryString["UserID"]);
47             Model.User user = getClient.DoWork(UserID);
48             user.UserName = this.txtUserName.Text;
49             user.Discribe = this.txtDiscribe.Text;
50             saveClient.DoWork(user);
51             Response.Write("修改成功!");
52         }
53     }
54     //列表及删除
55     public partial class SearchAndRemove : System.Web.UI.Page
56     {
57         Service.SearchClient searchClient = new Service.SearchClient();
58         Service.RemoveClient removeClient = new Service.RemoveClient();
59         protected void Page_Load(object sender, EventArgs e)
60         {
61             if (!Page.IsPostBack)
62             {
63                 GetUsers();
64             }
65         }
66
67         protected void GetUsers()
68         {
69             this.repUsers.DataSource = searchClient.DoWork();
70             this.repUsers.DataBind();
71         }
72
73         protected void lbtnRemoveCommand(object sender, CommandEventArgs e)
74         {
75             int UserID = Convert.ToInt32(e.CommandName);
76             removeClient.DoWork(UserID);
77             Response.Write("删除成功~");
78             GetUsers();
79         }
80     }

```

将生成的配置文件中的 <system.serviceModel>复制到Client的Web.config中，代码如下：

```

1 <system.serviceModel>
2 <bindings>
3 <basicHttpBinding>
4 <binding name="BasicHttpBinding_IAdd" closeTimeout="00:01:00"
5     openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
6     allowCookies="false" bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
7     maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
8     messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
9     useDefaultWebProxy="true">
10 <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
11     maxBytesPerRead="4096" maxNameTableCharCount="16384" />
12 <security mode="None">
13 <transport clientCredentialType="None" proxyCredentialType="None"
14     realm="" />
15 <message clientCredentialType="UserName" algorithmSuite="Default" />
16 </security>
17 </binding>
18 <binding name="BasicHttpBinding_IRemove" closeTimeout="00:01:00"
19     openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
20     allowCookies="false" bypassProxyOnLocal="false"

```

```
hostNameComparisonMode="StrongWildcard"
21         maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
22         messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
23         useDefaultWebProxy="true">
24         <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
25             maxBytesPerRead="4096" maxNameTableCharCount="16384" />
26         <security mode="None">
27             <transport clientCredentialType="None" proxyCredentialType="None"
28                 realm="" />
29             <message clientCredentialType="UserName" algorithmSuite="Default" />
30         </security>
31     </binding>
32     <binding name="BasicHttpBinding_ISearch" closeTimeout="00:01:00"
33         openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
34         allowCookies="false" bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
35         maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
36         messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
37         useDefaultWebProxy="true">
38         <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
39             maxBytesPerRead="4096" maxNameTableCharCount="16384" />
40         <security mode="None">
41             <transport clientCredentialType="None" proxyCredentialType="None"
42                 realm="" />
43             <message clientCredentialType="UserName" algorithmSuite="Default" />
44         </security>
45     </binding>
46     <binding name="BasicHttpBinding_ISave" closeTimeout="00:01:00"
47         openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
48         allowCookies="false" bypassProxyOnLocal="false"
49     hostNameComparisonMode="StrongWildcard"
50         maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
51         messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
52         useDefaultWebProxy="true">
53         <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
54             maxBytesPerRead="4096" maxNameTableCharCount="16384" />
55         <security mode="None">
56             <transport clientCredentialType="None" proxyCredentialType="None"
57                 realm="" />
58             <message clientCredentialType="UserName" algorithmSuite="Default" />
59         </security>
60     </binding>
61     <binding name="BasicHttpBinding_IGet" closeTimeout="00:01:00"
62         openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
63         allowCookies="false" bypassProxyOnLocal="false"
64     hostNameComparisonMode="StrongWildcard"
65         maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
66         messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
67         useDefaultWebProxy="true">
68         <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
69             maxBytesPerRead="4096" maxNameTableCharCount="16384" />
70         <security mode="None">
71             <transport clientCredentialType="None" proxyCredentialType="None"
72                 realm="" />
73             <message clientCredentialType="UserName" algorithmSuite="Default" />
74         </security>
75     </binding>
76 </basicHttpBinding>
77 </bindings>
78 <client>
79     <endpoint address="http://localhost:8080/Add.svc" binding="basicHttpBinding"
80         bindingConfiguration="BasicHttpBinding_IAdd" contract="IAdd"
81         name="BasicHttpBinding_IAdd" />
82     <endpoint address="http://localhost:8080/Remove.svc" binding="basicHttpBinding"
83         bindingConfiguration="BasicHttpBinding_IRemove" contract="IRemove"
84         name="BasicHttpBinding_IRemove" />
85     <endpoint address="http://localhost:8080/Search.svc" binding="basicHttpBinding"
86         bindingConfiguration="BasicHttpBinding_ISearch" contract="ISearch"
87         name="BasicHttpBinding_ISearch" />
88     <endpoint address="http://localhost:8080/Save.svc" binding="basicHttpBinding"
89         bindingConfiguration="BasicHttpBinding_ISave" contract="ISave"
90         name="BasicHttpBinding_ISave" />
91     <endpoint address="http://localhost:8080/Get.svc" binding="basicHttpBinding"
92         bindingConfiguration="BasicHttpBinding_IGet" contract="IGet"
93         name="BasicHttpBinding_IGet" />
94 </client>
95 </system.serviceModel>
```



-----最终效果-----

添加：