第 18 章　WCF 开发基础

WCF（Windows Communication Foundation）是.NET Framework 的扩展，WCF 提供了创建安全的、可靠的、事务服务的统一框架，WCF 整合和扩展了现有分布式系统的开发技术，如 Microsoft .NET Remoting、Web Services、Web Services Enhancements (WSE)等等，来开发统一的可靠的应用程序系统。

18.1　了解 WCF

WCF 是.NET Framework 的扩展，同时 WCF 提供了一种在 Windows 环境下进行客户端开发和服务端开发的 SDK，并且为服务提供了运行环境。WCF 提供了创建安全的、可靠的、事务服务的统一框架，整合了现有的分布式技术，开发人员能够使用 WCF 快速创建基于服务的应用程序。

18.1.1　什么是 WCF

WCF 是基于 Windows 平台下开发和部署服务的软件开发包（Software Development Kit，SDK）。WCF 提供了服务的运行环境，这样就让开发人员能够将 CLR 类型公开为服务，也能够通过使用 CLR 类型来使用服务。WCF 框架模型如图 18-1 所示。

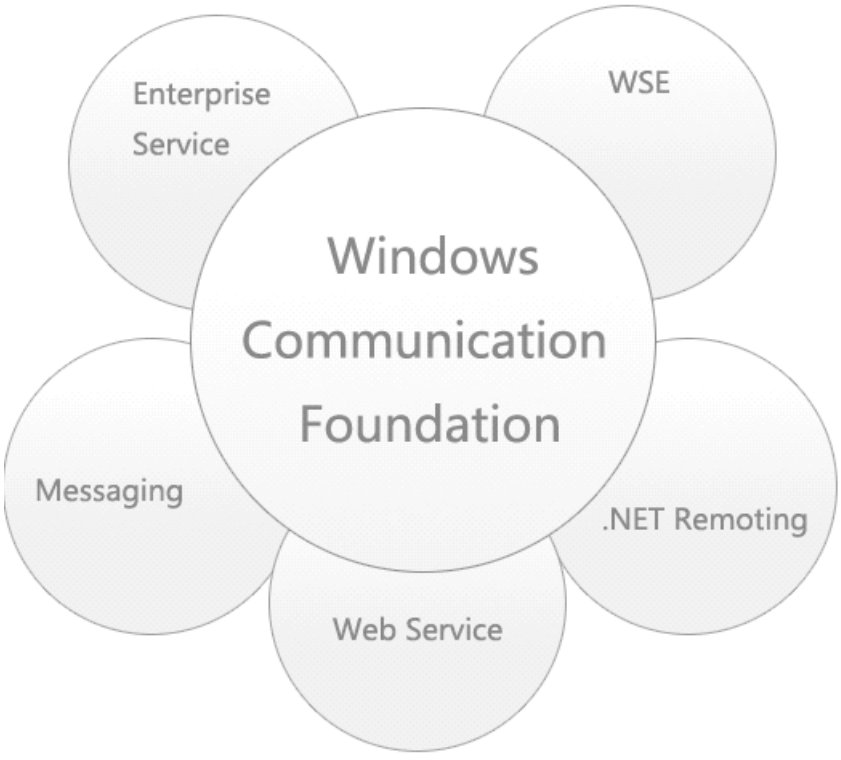

图 18-1　WCF 框架模型

WCF 提供了创建安全的、可靠的、事务服务的统一框架，WCF 整合和扩展了现有分布式系统的开发技术，如 Microsoft .NET Remoting、Web Services、Web Services Enhancements (WSE)等等，来开发统一的可靠系统。WCF 简化了 SOA 框架的应用，同时也统一了 Enterprise Services、Messaging、.NET Remoting、Web Services、WSE 等技术，极大的方便了开发人员进行 WCF 应用程序的开发和部署，同时也降低了 WCF 应用开发的复杂度。

WCF 支持大量的 Web Service 标准，这些标准包括 XML、XSD、SOAP、Xpath、WSDL 等标准和规范，所以对于现有的标准，开发人员能够方便的进行移植。同时 WCF 可以使用 Attribute 属性进行 WCF 应用程序配置，提高了 WCF 应用的灵活性。WCF 遵循客户端/服务器模型在应用程序之间进行通信，客户端程序能够通过服务器端提供的 EndPoint 端直接访问服务，如图 18-2 所示。

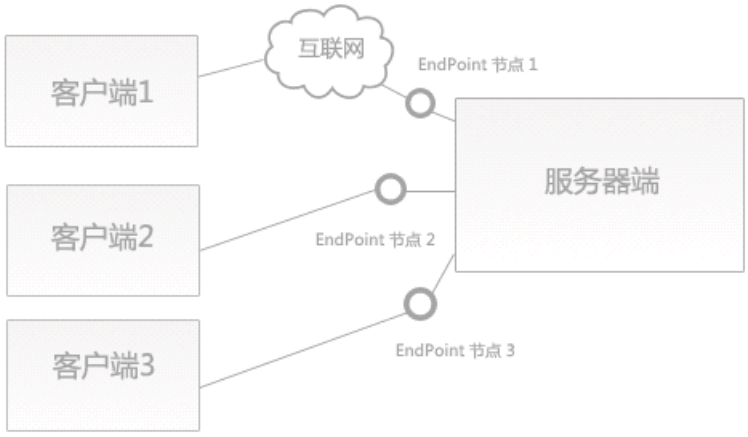

图 18-2　WCF 通信模型

虽然开发人员需要实现服务可以不使用 WCF，但是 WCF 封装了现有的类和结构，提供了服务实现的便捷手段，通过使用 WCF 能够快速的实现服务并让其他的应用程序使用服务。WCF 是微软提供的一系列协议的标准，包括服务交互、类型转换等。

WCF 中绝大部分的实现和功能都包含在一个单独的程序集 System.ServiceModel.dll 中，命名空间为 System.ServiceModel。通过使用 System.ServiceModel 命名空间能够快速搭建 WCF 应用程序环境。WCF 是.NET 3.0 的一部分，但是.NET 3.0 是基于.NET 2.0 为基础而存在的，如果需要搭建和使用 WCF 应用，则服务器应该具备.NET 3.0 环境。

18.1.2　为什么需要 WCF

在传统的应用程序开发中，例如在为麦当劳开发一个餐饮统计的应用程序，这个应用程序能够统计麦当劳的餐饮系统，包括每天客户购买的餐饮、餐饮的价格以及当天的餐饮统计。这个应用程序通常是安装在麦当劳店面主机中的，但是有很多的应用程序将需要对此餐饮统计应用程序进行访问和数据提取，这些应用程序有的是基于.NET 的，有的是基于 J2EE 的，另一些可能是基于 ASP.NET 的 Web 应用，这样就造成了应用程序访问困难。如图 18-3 所示。

图 18-3　麦当劳业务模拟图

如图 18-3 中所示，麦当劳的餐饮业务也许需要支持很多其他的设备，在现在智能手机发达的今天，很多客户可能可以从移动客户端访问麦当劳的餐饮业务，这些移动客户端可能是 PDA、Windows Mobile、GPhone 或者 IPhone。在其他的客户端访问时，例如总部可能需要提取分部的数据，用户可以从网站中购买餐饮，分部经理需要对当天的数据进行统计，或者购物中心应用程序访问餐饮应用程序以增删数据，这些流程都必须考虑到平台、协议和通信等诸多因素。

WCF 可以看作是 ASMX、.NET Remoting、Enterprise Service、WSE、MSMQ 这些技术的并集，虽然在复杂度上 WCF 很可能比这些技术更加复杂，因为 WCF 是面向服务构架的，所以对于上述的麦当劳餐饮业务的例子，如果使用 WCF 就能够很好的实现不同平台，不同设备之间的安全性、可依赖性、互操作性等特性，而因为 WCF 对现有技术的封装，开发人员可以无需关心 ASMX，、Net Remoting 这些技术的实现细节。

18.2　WCF 基础

在了解了 WCF 的概念和通信原理，以及为什么要使用 WCF 之后，就能够明白 WCF 在现在的应用程序开发中所起到的作用，WCF 能够实现不同技术和平台之间的安全性、可依赖性和用户操作性的实现，对大型应用程序开发起到促进作用。

18.2.1　服务

服务是一组公开的功能的集合。在软件开发领域，从传统的面向过程，到面向对象，然后历经了面向组件的开发一致发展到当今的面向服务开发。

1. WCF 服务

面向服务开发也并不是什么新技术，面向服务开发只是之前的面向过程、面向对象、组件开发和面向服务开发一种补充。面向服务开发有如下优点：

- 重用性：面向服务的开发提升了应用程序的重用性，通过创建可用于服务的接口能够实现不同应用程序中使用相同或类似程序实现的代码。
- 注重效率：面向服务的开发可以使用现有的服务的集合，这样能够让开发人员能够快速的进行

数据交换和开发，而无需关注底层服务的实现。

- 松耦合：面向服务的应用程序是独立于服务执行环境的应用程序，这样就让应用程序成为一个松耦合的应用。
- 职责划分：通过使用面向服务的开发能够进行职责的划分，例如经理和业务人员只需关心业务和统计数据即可，开发人员能够关注应用程序的开发。

一个面向服务的应用程序会将众多的服务集成到一起，形成单个逻辑单元，如图 18-4 所示。

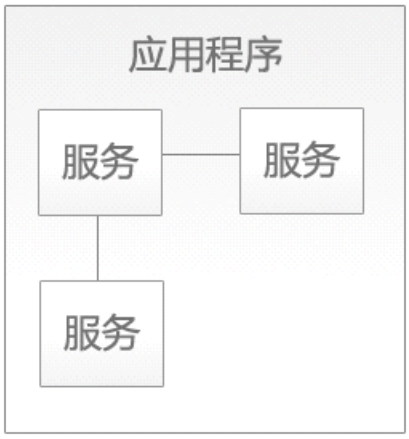

图 18-4　面向服务的应用

WCF 中的服务可以是本地的，也可以使用远程的服务。对于客户端而言，客户端只需要通过使用服务来实现应用程序功能，这些客户端也可以是不同的类型，包括 Windows 应用程序，ASP.NET 应用程序甚至是移动终端。

对于客户端而言，客户端是通过使用消息与服务器进行通信。消息可以直接在客户端与服务之间进行传递，也可以通过中间方进行传递。在服务器和客户端之间的消息是通过 SOAP 进行通信的，SOAP 与 Web 应用开发中不同的是，Web 应用通常需要某个具体的协议进行相应功能的实现，例如 HTTP、FTP 协议等，而在 WCF 中，WCF 服务可以在不同的协议中进行传递，并不局限于某个协议。正是因为如此，客户端与服务器之间的要求往往不是必须的，这也就是说，WCF 客户端可以与一个非 WCF 服务器进行信息通信，而一个非 WCF 客户端也可以与一个 WCF 服务器进行信息通信。

为了保障 WCF 服务器的安全性，WCF 服务器不允许直接对服务的调用。对于 WCF 客户端，只允许使用代理（Proxy）将调用信息转发给服务器。代理向客户端公开的操作和服务器端的操作相同。

2．服务的执行边界

WCF 能够让客户端跨越执行边界与 WCF 服务进行通信，WCF 客户端和 WCF 服务器进行通信必须使用带来与服务进行通信，即使是与本地服务进行通信，如图 18-5 所示。

图 18-5 展示了 WCF 客户与本机服务进行通信，WCF 不仅能够支持不同应用程序域之间的服务的访问，也能够支持不同进程之间的服务的访问。这就让 WCF 客户端可以调用一个应用程序中的服务，也可以调用不同应用程序甚至不同进程中的 WCF 服务。不仅如此，WCF 还支持客户端对远程计算机的中服务的调用，在远程服务调用中，WCF 允许客户端可以跨越 Intranet 或 Internet 的边界进行远程服务的访问和调用，如图 18-6 所示。

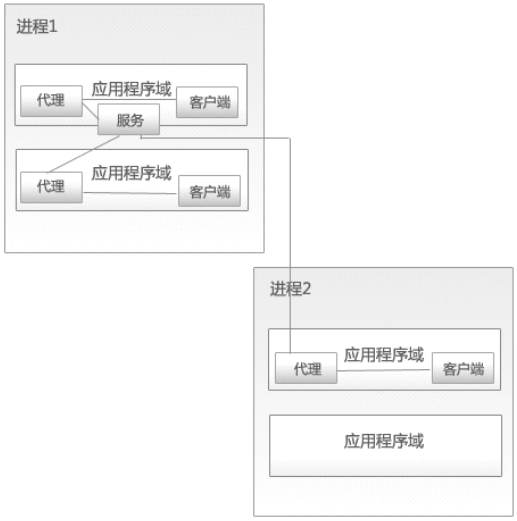

图 18-5　WCF 与本机服务进行通信

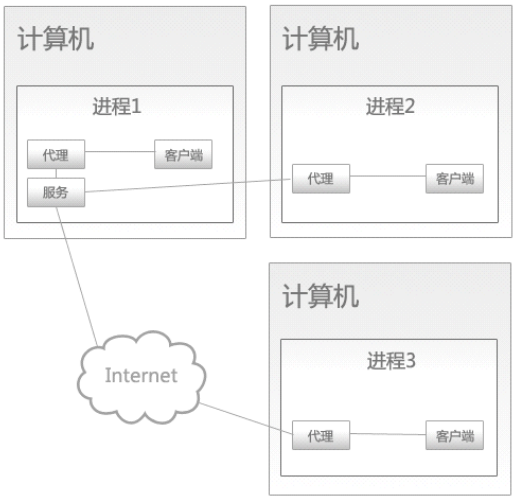

图 18-6　WCF 与远程服务进行通信

图 18-6 展示了 WCF 客户端与远程服务进行通信，无论 WCF 客户端是与远程服务进行通信还是与本地进程进行通信，都需要使用代理。

18.2.2　地址

在 Internet 中，为了标识每个计算机，就需要使用 IP 进行地址划分，在生活中也有此实例，例如每个家庭都有一个门牌号，为了方便找到某个人，则必须通过门牌号找到这个人，同样对于 WCF 服务而言，每个 WCF 服务都有一个自己的地址。

1．WCF 地址

WCF 地址包含两个元素，服务位置与传输协议，服务位置包括目标机器名、站点或网络、通信端口、管道或队列，以及一个可选的特定路径或者 URI。WCF 地址也可以是用于服务通信的传输样式。WCF 支持的传输样式包括：

- HTTP：超文本传输协议。
- TCP：传输控制协议。
- Peer network：对等网。
- IPC：基于命名管道的内部进程通信协议。
- MSMQ：微软消息队列。

地址通常通过[基地址]/[可选的 URI]的格式进行 WCF 地址描述，示例地址如下所示。

```
http://localhost:8731
http://localhost:8731/18-2
net.tcp://localhost:8731/server/18-2
net.pipe://localhost/18-2
net.msmq://localhost/18-2
```

其中关于 http://localhost:8731 这个地址可以称作使用 http 协议，访问计算机为 localhost 的端口 8731 正在等待客户端的调用。而对于 http://localhost:8731/18-2 这个地址可以称作使用 http 协议，访问计算机为 loacalhost 的端口为 8731 的 18-2 服务正在等待客户端的调用。

2．TCP 地址

TCP 地址使用 TCP 传输控制协议作为通信协议，使用 TCP 地址的示例地址如下所示。

```
net.tcp://localhost:8731/server/18-2
```

如果端口号没有指定，则 TCP 会使用默认端口号 808 作为其默认端口，示例地址如下所示。

```
net.tcp://localhost/server/18-2
```

3．HTTP 地址

HTTP 地址使用 HTTP 传输控制协议作为其通信协议，使用 HTTP 地址的示例地址如下所示。

```
http://localhost:8731/18-2
```

如果端口号没有指定，则 HTTP 会使用默认的端口号 80 作为其默认端口。

注意：无论是 TCP 协议还是 HTTP 协议，不同的服务可以公用相同的端口号。

4．IPC 和 MSMQ 地址

IPC 地址使用 net.tcp 作为通信协议，使用 net.tcp 地址的示例地址如下所示。

```
net.pipe://localhost/18-2
```

正是因为 IPC 地址使用 net.pipe 进行传输，所以 IPC 地址将使用 Windows 的命名管道机制。在 WCF 中，如果服务使用命名管道，则该服务只能接收来自同一台客户端计算机的调用。因此，在使用时必须明确的指定 WCF 提供服务的计算机名，从而为管道名提供一个惟一的标识字符串。而 MSMQ 地址使用 net.msmq 进行传输，即使用了微软消息队列机制，MSMQ 地址的示例地址如下所示。

```
net.msmq://localhost/18-2
```

18.2.3 契约

在 WCF 中，所有的 WCF 服务都会被公开成为契约。契约是服务的功能的标准描述方式，通常情况下 WCF 包含四种类型的契约，这些契约如下所示。

- 服务契约（Service Contract）：服务契约定义了客户端能够执行的操作，服务契约是 WCF 中使用最为广泛的一种契约。
- 数据契约（Data Contract）：数据契约定义了客户端与服务器之间交互的数据类型。
- 错误契约（Fault Contract）：错误契约定义了操作中出现的异常，包括定义服务出现的错误并传递返回给客户端。
- 消息契约（Message Contract）：消息契约允许服务直接与消息交互，但是 WCF 很少使消息契约。

WCF 使用特性 ServiceContractAttribute 标识服务契约，而使用 OperationContractAttribute 标识服务方法。示例代码如下所示。

```
[ServiceContract]                                                    //标识服务契约
public interface IService1                                           //实现接口
{
    [OperationContract]                                              //方法声明
    string GetData(int value);                                       //创建方法
    [OperationContract]
    CompositeType GetDataUsingDataContract(CompositeType composite);
    // 任务: 在此处添加服务操作
}
```

上述代码使用 ServiceContractAttribute 标识服务契约，而使用 OperationContractAttribute 标识服务方法，OperationContract 只能用于方法，指明客户端是否能够调用此方法。使用 OperationContract 标识可以标识私有方法以使用 SOA 的方式进行构架，虽然这样是可以实现客户端调用，但是作为面向对象的

设计是不推荐使用该方法的。由于能够使用 ServiceContractAttribute 来标识服务契约，开发人员能够自定义标识指定相应的方法是否能够被客户端调用，示例代码如下所示。

```
[OperationContract]
CompositeType GetDataUsingDataContract(CompositeType composite);        //标识方法
string Post(string content);
```

在上述代码中的 Post 方法不会成为契约。WCF 允许开发人员使用 DataContractAttribute、DataMemberAttribute 来标识自定义数据类型和属性，示例代码如下所示。

```
[DataMember]                                                            //设置 DataMember
string stringValue = "Hello ";                                          //创建 string 变量
[DataMember]                                                            //设置 DataMember
public bool BoolValue                                                   //设置属性
{
    get { return boolValue; }
    set { boolValue = value; }                                          //设置属性默认值
}
[DataMember]                                                            //设置 DataMember
public string StringValue                                               //设置属性
{
    get { return stringValue; }
    set { stringValue = value; }                                        //设置属性默认值
}
```

上述代码使用了 DateMember 定义了属性和相应的字段，这样就可以在服务方法中传递复杂的数据体了。

18.3 WCF 应用

在了解了基本的 WCF 概念后，先不用着急继续了解 WCF 应用体系，通过创建 WCF 应用可以深入的了解服务、地址和契约的概念。WCF 还允许开发人员创建和声明契约，通过契约的声明，客户端可以通过远程调用以实现自身的程序。

18.3.1 创建 WCF 应用

在 Visual Studio 2008 中，可以方便的创建 WCF 应用。在菜单栏中选择【文件】选项，在下拉菜单中单击【新建项目】选项，在弹出的【新建项目】窗口中选择 WCF，如图 18-7 所示。

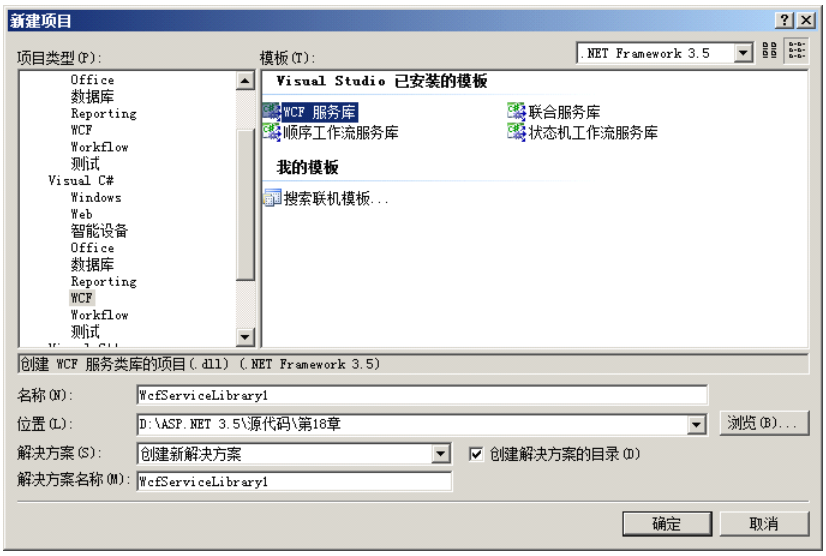

图 18-7　创建 WCF 服务库

创建 WCF 服务库后，应用程序会自动生成 Server1.cs 和 IServer1.cs 接口，IServer1.cs 接口示例代码如下所示。

```
using System;
using System.Collections.Generic;
using System.Linq;                                          //使用必要的命名空间
using System.Runtime.Serialization;                         //使用必要的命名空间
using System.ServiceModel;                                  //使用必要的命名空间
using System.Text;
namespace _18_2
{
    // 注意: 如果更改此处的接口名称“IService1”，也必须更新 App.config 中对“IService1”的引用。
    [ServiceContract]
    public interface IService1
    {
        [OperationContract]                                 //声明契约
        string GetData(int value);
        [OperationContract]
        CompositeType GetDataUsingDataContract(CompositeType composite);
        // 任务: 在此处添加服务操作
    }
    // 使用下面示例中说明的数据协定将复合类型添加到服务操作
    [DataContract]
    public class CompositeType                              //创建类
    {
        bool boolValue = true;                              //创建字段
        [DataMember]
        string stringValue = "Hello ";                      //声明 string 变量
        [DataMember]
        public bool BoolValue                               //创建属性
        {
            get { return boolValue; }
            set { boolValue = value; }
```

```
        }
        [DataMember]
        public string StringValue                                    //创建属性
        {
            get { return stringValue; }
            set { stringValue = value; }
        }
    }
}
```

上述代码创建了一个 IServer1 接口，并通过 ServiceContractAttribute 标识服务契约，同样也通过 DataContractAttribute、DataMemberAttribute 来标识自定义数据类型和属性，示例代码如下所示。

```
    [ServiceContract]                                                //标识服务契约
    public interface IService1                                       //创建接口
    [DataContract]
    public class CompositeType
```

创建接口后则需要实现接口，接口的实现在 Server.cs 文件内，示例代码如下所示。

```
    public class Service1 : IService1                                //实现接口
    {
        public string GetData(int value)                            //实现接口方法
        {
            return string.Format("You entered: {0}", value);        //返回 string 值
        }
        public CompositeType GetDataUsingDataContract(CompositeType composite)  //实现接口方法
        {
            if (composite.BoolValue)                                 //实现方法代码
            {
                composite.StringValue += "Suffix";                   //添加相应数据
            }
            return composite;                                        //返回值
      }
```

上述代码实现了 IServer1 接口中的方法，单击【运行】按钮或快捷键【F5】，WCF 应用程序就能够运行，如图 18-8 所示。

从图 18-8 中可以看出，在 Server1.cs 文件中实现的方法都能够在测试客户端中看到，单击测试客户端中相应的方法就能够在客户端测试调用服务器端的方法，如图 18-9 所示。

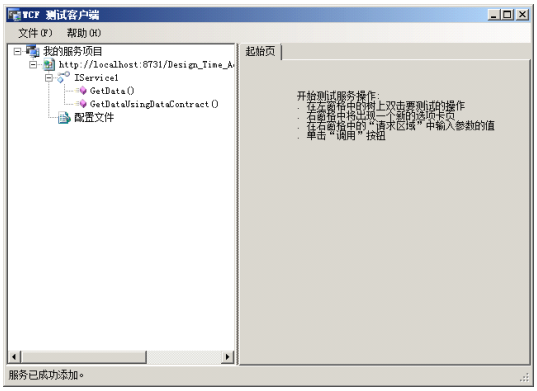

图 18-8　服务测试客户端

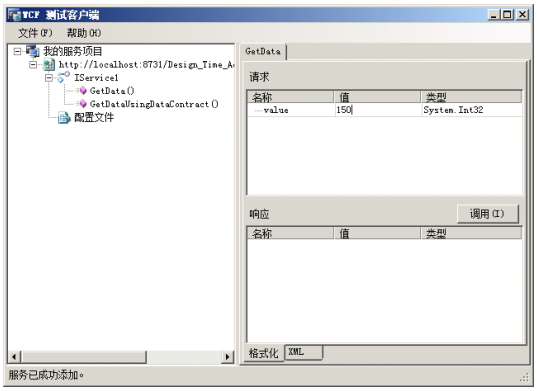

图 18-9　测试方法调用

双击 GetData 方法后，在右侧选项卡中就会分为两层，这两层分别为请求和响应。在请求框中可以

在值那一栏编写需要传递的值，编写完毕后单击【调用】按钮就能够实现服务器端方法的调用并在响应框中呈现相应的值。

18.3.2 创建 WCF 方法

一个简单的 WCF 应用程序运行后，就会发现其实 WCF 并没有想象中的复杂。WCF 允许开发人员通过使用 ServiceContractAttribute 标识服务契约，同样开发人员还能够创建服务契约以提供客户端的方法的调用。在 IServer1 接口中首先需要定义该方法，示例代码如下所示。

```
int GetSum(DateTime time);                                    //定义接口方法
[OperationContract]                                           //标识调用
string GetShopInformation(string address);
```

上述代码声明了两个方法，这两个方法分别为 GetSum 和 GetShopInformation,，GetSum 用于获取某一天麦当劳餐厅中出售总量，而 GetShopInformation 用于获取麦当劳地址和一些商店的信息，GetSum 和 GetShopInformation 具体实现如下所示。

```
public int GetSum(DateTime time)                              //实现方法
{
    int BreadNum = 10;                                        //声明必要字段
    int Milk = 5;                                             //声明必要字段
    int HotDryNuddle = 20;                                    //声明必要字段
    int today = BreadNum + Milk + HotDryNuddle;               //实现计算
    return today;                                             //返回值
}
public string GetShopInformation(string address)              //实现方法
{
    if (address == "武汉")                                    //判断地址
    {
        return "武汉麦当劳连锁店";                            //返回相应结果
    }
    else if (address == "北京")                               //判断地址
    {
        return "北京麦当劳连锁店";                            //返回相应结果
    }
    else if (address == "上海")                               //判断地址
    {
        return "上海麦当劳连锁店";                            //返回相应结果
    }
    else
    {
        return "没有该连锁店";                                //返回默认结果
    }
}
```

在 GetSum 方法中，用于获取当天的销售总量，而 GetShopInformation 实现了商店信息的反馈，运行后如图 18-10 所示。

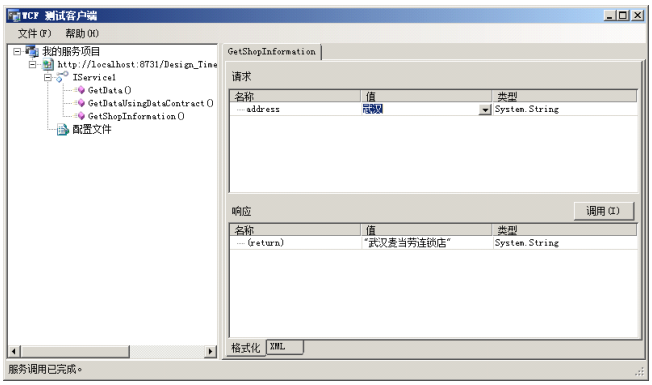

图 18-10　创建方法

从图 18-10 中可以看出，GetShopInformation 已经在测试客户端中服务器项目中显式了，并且输入了“武汉”这个信息，就能够返回“武汉麦当劳连锁店”。而 GetSum 方法并没有呈现在服务器项目中，这是因为 GetSum 方法并没有使用[OperationContract]标识进行声明，所以不会作为契约的一部分，若需要在客户端调用 GetSum 方法就必须使用[OperationContract]标识进行声明，示例代码如下所示。

```
[OperationContract]
int GetSum(DateTime time);                                    //标识客户端方法
```

WCF 应用程序运行后如图 18-11 所示。

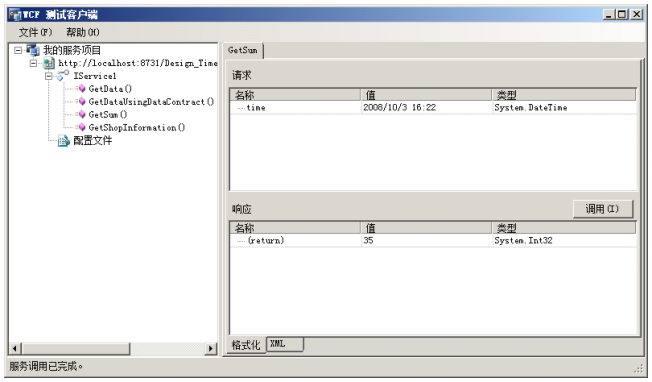

图 18-11　使用[OperationContract]标识

开发人员能够使用 WCF 快速的创建 WCF 应用程序并从客户端调用该方法，这样就能够在客户端隐藏服务器方法并且让客户端只关注逻辑实现而不需要关注底层是如何进行消息通信的。

18.4　WCF 消息传递

通过了解了 WCF 的一些基本概念并创建和编写 WCF 应用中的相应方法，实现了 WCF 服务和客户端之间的调用，就能够理解 WCF 应用是如何进行通信的。了解了一些基本的 WCF 概念后，还需要深入了解 WCF 消息的概念。

18.4.1　消息传递

客户端与服务器之间是通过消息进行信息通信的，通过使用消息，客户端和服务器之间能够通过使

用消息交换来实现方法的调用和数据传递。

1．Request/Reply 消息传递模式

Request/Reply 模式是默认的消息传递模式，该模式调用服务器的方法后需要等待服务的消息返回，从而获取服务器返回的值。Request/Reply 模式是默认模式，在声明时无需添加其模式的声明，示例代码如下所示。

```
[OperationContract]
string GetShopInformation(string address);                                    //默认模式
```

上述代码就使用了一个默认的 Request/Reply 模式进行消息传递，GetShopInformation 方法同样需要实现，示例代码如下所示。

```
public string GetShopInformation(string address)
{
    if (address == "武汉")                                                    //判断地址
    {
        return "武汉麦当劳连锁店";                                            //返回相应结果
    }
    else if (address == "北京")                                               //判断地址
    {
        return "北京麦当劳连锁店";                                            //返回相应结果
    }
    else if (address == "上海")                                               //判断地址
    {
        return "上海麦当劳连锁店";                                            //返回相应结果
    }
    else
    {
        return "没有该连锁店";                                                //返回默认结果
    }
}
```

GetShopInformation 方法返回一个 string 的值给客户端，客户端调用服务器的方法时，首先会向服务器发送消息，以告诉服务器客户端需要调用一个方法，当服务器接收消息后会返回消息给客户端。在这一段过程中，客户端会等待服务器端的相应，当客户端接受到服务器的相应后，则会呈现在客户端应用程序中。如图 18-12 所示。

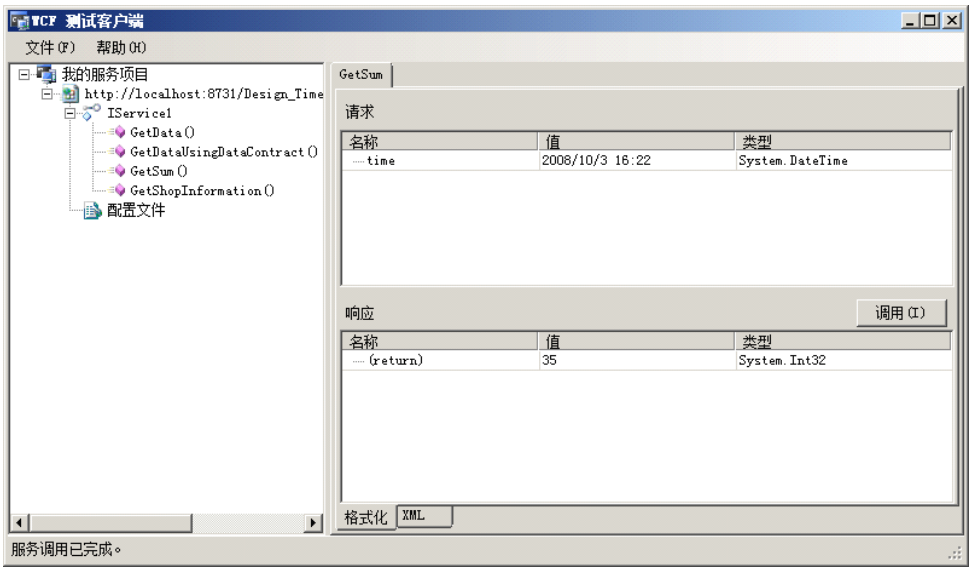

图 18-12　Request/Reply 模式

2．One-way 消息传递模式

One-way 模式和 Request/Reply 模式不同的是，如果使用 One-way 模式定义一个方法，该方法被调用后会立即返回。使用 One-way 模式修饰的方法必须是 void 方法，如果该方法不是 void 修饰的方法或者包括 out/ref 等参数，则不能使用 One-way 模式进行修饰，示例代码如下所示。

```
[OperationContract(IsOneWay = true)]                    //标识 One-way 模式
void OutputString();                                    //定义方法
```

该方法使用了 One-way 模式，则不能有参数的输出，只允许 void 关键字修饰该方法，OutpuString 方法的具体实现如下所示。

```
public void OutputString()                              //实现方法
{
    Console.WriteLine("IsOneWay=true");
}
```

运行 WCF 应用后，执行 OutpuString 方法后结果如图 18-13 所示。

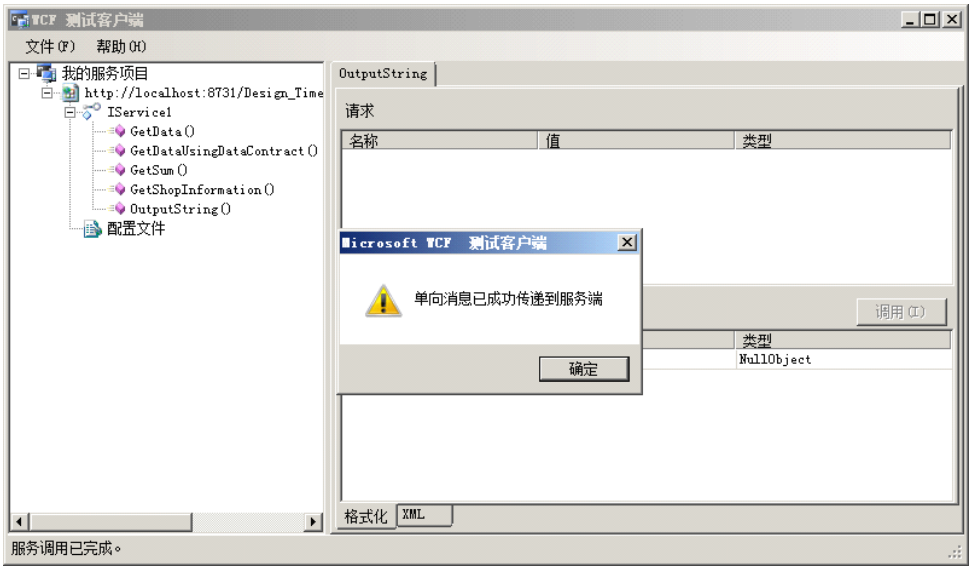

图 18-13　One-way 模式

WCF 的消息传递模式不仅包括这两种模式，还包括 duplex 模式，duplex 是 WCF 消息传递中比较复杂的一种模式，由于篇幅限制，本书不再进行详细的介绍。

18.4.2　消息操作

由于 WCF 的客户端和服务器之间都是通过消息响应和通信的，那么在 WCF 应用的运行过程中，消息是如何在程序之间进行操作的，这就需要通过 XML 文档来获取相应的结果。在客户端和服务器之间出现信息通信，并且客户端调用了服务器的方法时，就会产生消息，如 GetSum 方法。GetSum 方法在接口中的代码如下所示。

```
[OperationContract]                                    //标识方法
int GetSum(DateTime time);                             //定义方法
```

在 GetSum 方法的实现过程中，只需要进行简单的操作即可，示例代码如下所示。

```
public int GetSum(DateTime time)                       //实现方法
{
    int BreadNum = 10;                                 //声明必要字段
    int Milk = 5;                                      //声明必要字段
    int HotDryNuddle = 20;                             //声明必要字段
    int today = BreadNum + Milk + HotDryNuddle;        //实现计算
    return today;                                      //返回值
}
```

上述代码执行后，客户端会调用服务器的 GetSum 方法，服务器接受响应再返回给客户端相应的值，如图 18-14 和图 18-15 所示。

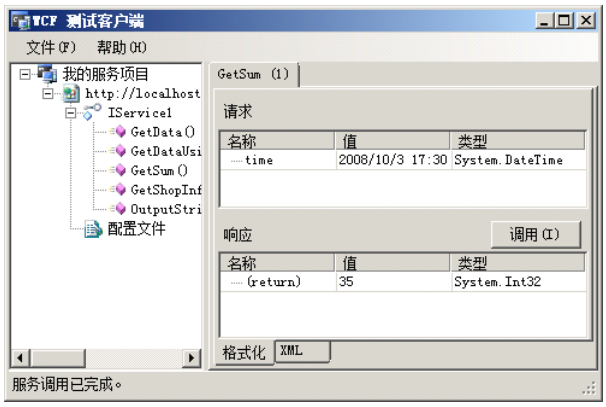

图 18-14　执行服务器方法

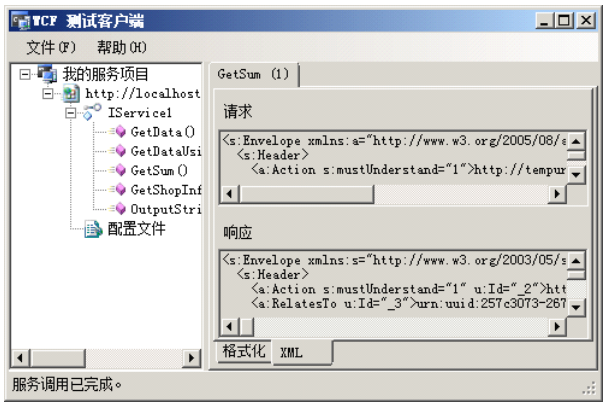

图 18-15　返回的 XML 格式文档

在运行后，测试客户端能够获取请求时和响应时的 XML 文档，其中请求时产生的 XML 文档如下所示。

```
<s:Envelope
  xmlns:a="http://www.w3.org/2005/08/addressing" xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/IService1/GetSum</a:Action>
    <a:MessageID>urn:uuid:dcc8a76e-deaf-45c4-a80c-2034b965d001</a:MessageID>
    <a:ReplyTo>
      <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
```

```
    </a:ReplyTo>
  </s:Header>
  <s:Body>
    <GetSum xmlns="http://tempuri.org/">
      <time>2008-10-03T17:30:00</time>
    </GetSum>
  </s:Body>
</s:Envelope>
```

从上述代码可以看到在 Action 节中，使用了相应的方法 GetSum，在 WCF 服务库编程中可以通过使用 OperationContract.Action 捕获相应的 Action 消息，示例代码如下所示。

```
[OperationContract(Action = "GetSum", ReplyAction = "GetSum")]
Message MyProcessMessage(Message m);
```

MyProcessMessage 实现示例代码如下所示。

```
public Message MyProcessMessage(Message m)
{
    CompositeType t = m.GetBody<CompositeType>();                    //获取消息
    Console.WriteLine(t.StringValue);                                //输出消息
    return Message.CreateMessage(MessageVersion.Soap11,
    "Add", new CompositeType("Hello World!"));                       //返回消息
}
```

上述代码将操作转换为消息后发送，开发人员可以通过 Windows 应用程序或 ASP.NET 应用程序获取修改后消息的内容。在进行消息的操作时，WCF 还允许开发人员使用 MessageContractAttribute / MessageHeaderAttribute 来控制消息格式，这比 DataContractAttribute 要更加灵活。

18.5 使用 WCF 服务

创建了一个 WCF 服务之后，为了能够方便的使用 WCF 服务，就需要在客户端远程调用服务器端的 WCF 服务，使用 WCF 服务提供的方法并将服务中方法的执行结果呈现给用户，这样保证了服务器的安全性和代码的隐秘性。

18.5.1 在客户端添加 WCF 服务

为了能够方便的在不同的平台，不同的设备上使用执行相应的方法，这些方法不仅不能够暴露服务器地址，同样需要在不同的客户端上能呈现相同的效果，这些方法的使用和创建不能依赖本地的应用程序，为了实现跨平台的安全应用程序开发就需要使用 WCF。

创建了 WCF 服务，客户端就需要进行 WCF 服务的连接，如果不进行 WCF 服务的连接，则客户端无法知道在哪里找到 WCF 服务，也无法调用 WCF 提供的方法。首先需要创建一个客户端，客户端可以是 ASP.NET 应用程序也可以是 WinForm 应用程序。右击解决方案管理器，单击【项目】，在下拉菜单中选择【添加新项】，分别为该项目添加一个 ASP.NET 应用程序和一个 WinForm 应用程序，如图 18-16 和图 18-17 所示。

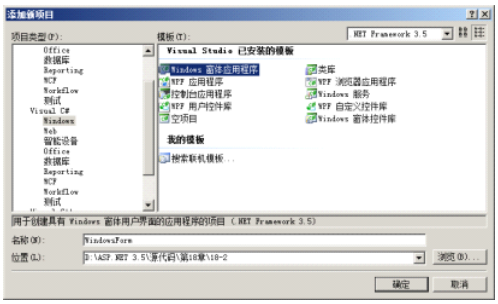

图 18-16　添加 Win Form 应用程序

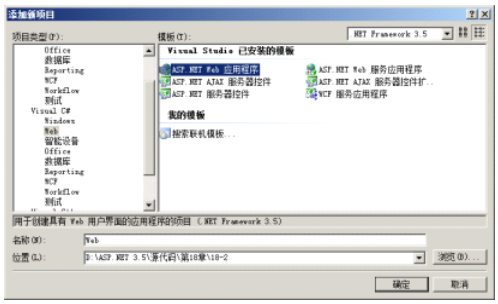

图 18-17　添加 ASP.NET 应用程序

添加完成后在项目中就会出现这两个项目，分别为这两个项目添加 WCF 引用，右击当前项目，在下拉菜单中单击【添加服务引用】选项，在弹出窗口中单击【发现】按钮，即可发现 WCF 服务，如图18-18所示。添加完成后 WCF 服务就会被挂起，等待客户端对 WCF 服务中的方法进行调用，如图 18-19 所示 。

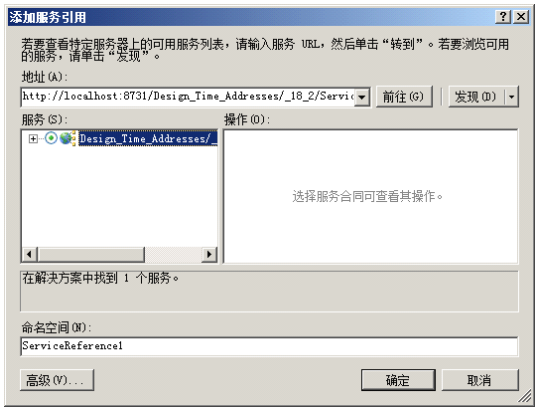

图 18-18　添加服务引用

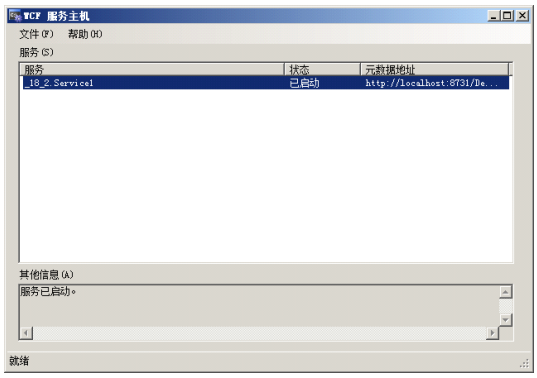

图 18-19　WCF 服务主机已经启动

分别为 ASP.NET 应用程序和 Win Form 应用程序添加 WCF 引用后，就可以在相应的应用程序中使用 WCF 服务提供的方法了。

18.5.2　在客户端使用 WCF 服务

当客户端添加了 WCF 服务的引用后，就能够非常方便的使用 WCF 服务中提供的方法进行应用程序开发。在客户端应用程序的开发中，几乎看不到服务器端提供的方法的实现，只能够使用服务器端提供方的方法。对于客户端而言，服务器端提供的方法是不透明的。

1. ASP.NET 客户端

在 ASP.NET 客户端中，可以使用 WCF 提供的服务实现相应的应用程序开发，例如通过地名获取麦当劳的商店的信息，而不想要在客户端使用数据库连接字串等容易暴露服务器端的信息，通过使用 WCF 服务提供的方法能够非常方便的实现这一点。Aspx 页面看代码如下所示。

```
<body>
    <form id="form1" runat="server">
    <div>
        输入地名 ：<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <br />
        获得的结果：<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
```

```
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="检索" />
    </div>
    </form>
</body>
```

上述代码在页面中拖放了两个 Textbox 控件分别用于用户输入和用户结果的返回，并拖放了一个按钮控件用于调用 WCF 服务中的方法并返回相应的值。.cs 页面代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (!String.IsNullOrEmpty(TextBox1.Text))
    {
        //开始使用 WCF 服务
        ServiceReference1.Service1Client ser = new Web.ServiceReference1.Service1Client();
        TextBox2.Text = ser.GetShopInformation(TextBox1.Text);                    //实现方法
    }
    else
    {
        TextBox2.Text = "无法检索,字符串为空";                                    //输出异常提示
    }
}
```

上述代码创建了一个 WCF 服务所提供的类的对象，通过调用该对象的 GetShopInformation 方法进行本地应用程序开发。上述代码运行后如图 18-20 和图 18-21 所示。

图 18-20　实现检索功能

图 18-21　实现异常处理

2. Win Form 客户端

在 Win Form 客户端中使用 WCF 提供的服务也非常的方便，其使用方法基本同 ASP.NET 相同，这也说明了 WCF 应用的开发极大的提高了开发人员在不同客户端之间的开发效率，节约了开发成本。在 Win Form 客户端中拖动一些控件作为应用程序开发提供基本用户界面，示例代码如下所示。

```
private void InitializeComponent()
{
    this.textBox1 = new System.Windows.Forms.TextBox();                    //创建 textBox
    this.dateTimePicker1 = new System.Windows.Forms.DateTimePicker();      //创建 TimePicker
    this.SuspendLayout();
    //
    // textBox1
```

```
        //
        this.textBox1.Location = new System.Drawing.Point(13, 13);          //实现 textBox 属性
        this.textBox1.Name = "textBox1";                                    //实现 textBox 属性
        this.textBox1.Size = new System.Drawing.Size(144, 21);              //实现 textBox 属性
        this.textBox1.TabIndex = 0;                                         //实现 textBox 属性
        //
        // dateTimePicker1
        //
        this.dateTimePicker1.Location = new System.Drawing.Point(166, 13);  //实现 TimePicker 属性
        this.dateTimePicker1.Name = "dateTimePicker1";                      //实现 TimePicker 属性
        this.dateTimePicker1.Size = new System.Drawing.Size(114, 21);       //实现 TimePicker 属性
        this.dateTimePicker1.TabIndex = 1;                                  //实现 TimePicker 属性
        this.dateTimePicker1.ValueChanged                                   //实现 TimePicker 属性
        += new System.EventHandler(this.dateTimePicker1_ValueChanged);
        //
        // Form1
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);       //实现 Form 属性
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;       //实现 Form 属性
        this.ClientSize = new System.Drawing.Size(292, 62);                 //实现 Form 属性
        this.Controls.Add(this.dateTimePicker1);                            //添加 Form 控件
        this.Controls.Add(this.textBox1);                                   //添加 Form 控件
        this.Name = "Form1";                                                //实现 Form 属性
        this.Text = "Form1";                                                //实现 Form 属性
        this.ResumeLayout(false);
        this.PerformLayout();
    }
```

上述代码在 Win From 窗体中创建了一个 TextBox 控件和一个 DataTimePicker 控件，并向窗体注册了 dateTimePicker1_ValueChanged 事件，当 DataTimePicker 控件中的值改变后，则会输出相应天数的销售值。在前面的 WCF 服务中，为了实现销售值统计，创建了一个 GetSum 方法，在 Win From 窗体中无需再实现销售统计功能，只需要调用 WCF 服务提供的方法即可，示例代码如下所示。

```
    private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
    {
        ServiceReference1.Service1Client ser = new WindowsForm.ServiceReference1.Service1Client();
        textBox1.Text = ser.GetSum(Convert.ToDateTime(dateTimePicker1.Text)).ToString();
    }
```

上述代码使用了 WCF 服务中提供的 GetSum 方法进行了相应天数的销售额的统计，运行后如图 18-22 所示。

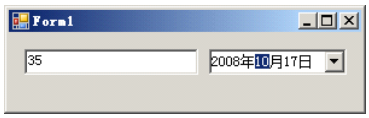

图 18-22　Win From 客户端使用 WCF 服务

创建和使用 WCF 服务不仅能够实现不同客户端之间实现相同的功能，还通过 WCF 应用提供了一个安全性、可依赖、松耦合的开发环境，对于其中任何一种客户端的实现，都不会暴露服务器中的私密信息，并且对于其中的某个客户端进行任何更改，也不会影响其他客户端，更不会影响到 WCF 服务器，这对应用程序开发和健壮性提供了良好的环境。

18.6 小结

本章简单的介绍了 WCF 的基本知识，包括什么是 WCF 和为什么需要 WCF。WCF 在现在的中大型应用程序开发中起到了非常重要的作用，使用 WCF 技术能够实现分布式的应用程序开发和管理，WCF 为应用程序开发提供了安全、可依赖和松耦合的开发环境。本章还包括：

- WCF 基础：讲解了基本的 WCF 知识，包括 WCF 技术的组成和为何需要 WCF。
- WCF 应用：通过实例讲解了如何创建一个 WCF 应用，并修改和创建相应的方法实现 WCF 服务中的方法。
- WCF 消息传递：简单的介绍了 WCF 消息传递的几种模式和消息操作。
- 使用 WCF 服务：介绍了如何在客户端中添加 WCF 引用并使用 WCF 引用。

本章只是简单的讲解了 WCF 的基础知识，本书并不是专门进行 WCF 知识的讲解，而且 WCF 知识需要一定的开发经验和编程水平，所以本书并没有详细的讲解 WCF 技术。但是在 ASP.NET 开发中，一些中大型的项目同样需要使用 WCF 进行分布式应用，从而实现不同的客户端对 ASP.NET 应用的访问，了解基本的 WCF 知识在今后的大型项目开发中会起到良好的作用。