Web Services Using C# and .NET

Student Guide

Revision 4.0

Object Innovations Course 4150

Web Services Using C# and .NET
Rev. 4.0

Student Guide

® is a registered trademark of Object Innovations.

Authors: Robert J. Oberg, Arun Ganesh, Will Provost

Special Thanks: Paul Nahay, Sharman Staples, Julian Templeman

Object Innovations
877-558-7246
www.objectinnovations.com

Published in the United States of America.

Table of Contents (Overview)

Directory Structure

- **Install the course software by running the self-extractor *Install_WsNetCs_40.exe*.**
- **The course software installs to the root directory *C:\OIC\WsNetCs*.**
  - Example programs for each chapter are in named subdirectories of chapter directories **Chap02**, **Chap03** and so on.
  - The **Labs** directory contains one subdirectory for each lab, named after the lab number. Starter code is frequently supplied, and answers are provided in the chapter directories.
  - The **Demos** directory is provided for performing in-class demonstrations led by the instructor.
  - **Deploy** is for testing deployment of a Web service to a different directory from the one in which it was created.
- **Data files install to the directory *C:\OIC\Data*.**
- **Log files are written to the directory *C:\OIC\Logs*.**

Table of Contents (Detailed)

Chapter 1

What Are Web Services?

What Are Web Services?

Objectives

After completing this unit you will be able to:

- **Answer the questions "What are Web services?" and "What is the need for Web services?"**
- **Illustrate how Web services are different from other distributed technologies.**
- **Provide examples of the different benefits of Web services.**
- **Outline the baseline Web services specifications of XML, SOAP, WSDL and UDDI.**
- **Identify the major players who are shaping the Web services architecture for the industry.**
- **Describe some business models for Web services.**
- **Discuss Service Oriented Architecture (SOA) and the use of Web services for its implementation.**
- **Discuss ASP.NET as a technology for implementing Web services and other .NET alternatives to Web services.**
- **Discuss WCF, which can be used to implement both SOAP and REST Web services.**

What is a Web Service?

- **A Web service is a message delivered over some transport mechanism.**
  - XML is the most common message format.
  - HTTP has been the most common transport.
  - SOAP has become an industry standard protocol running on top of HTTP.
- **The producer of the Web service maps an object, service, program or database to the XML message.**
- **The consumer of the Web service maps the XML message to an object, service, program, or database.**
- **A Web service is executed via the application that consumes it.**

Why are Web Services Needed?

- **Web services provide a neutral means to integrate programs built on different development platforms.**
  - Web services are superior to distributed object technologies.
- **Web services provide a means to evolve applications as customer requirements change.**
  - Web services enable applications built with a Service Oriented Architecture (SOA).
- **Let us examine each of these in more detail.**

Distributed Object Computing

- **Building scalable solutions require applying the appropriate level of computing power to different parts of the application.**
- **Distributed object computing extends an object-oriented programming system by allowing objects to coexist across a heterogeneous network and interact as if they resided on the same machine.**
- **In the early 1990's many companies including Microsoft, Sun Microsystems, the Object Management Group and IBM recognized the need of distributed computing and developed their own technologies to facilitate communication among distributed components.**
  - Microsoft's DCOM (Distributed Component Object Model)
  - Sun Microsystems' RMI (Remote Method Invocations)
  - Object Management Group's CORBA (Common Object Request Broker Architecture)
  - IBM's DSOM (Distributed System Object Model)
- **Distributed object systems have several problems.**

OOP and Network Latency

- **The style of object-oriented programming directly conflicts with the performance of distributed systems.**
  - Object-oriented classes are fine grained, with property and field accessors.
  - Object interactions lead to many round trips across the network. Network latency causes severe performance degradation. You want to minimize the number of interactions between client and server.

```
class Store
{
   public string Name(){get;set}
   public string Location() {get;set}
   public string Hours(){get;set}
   public void CreateStore();
   public void SaveStore();
}
```

- **Distributed systems require "chunky" rather than "chatty" interfaces between the distributed pieces.**
  - Such interfaces have methods that transfer larger quantities of data.

```
class Store
{
   public void CreateStore(string name, string
                           location, string hours);
   public void SaveStore(string name, string
                           location, string hours);
}
```

Object State and Scalability

- **Scalability is the performance of the system under increased load.**
  - Systems are scalable if performance can be increased by just adding hardware.
- **Object-oriented systems tend to have many fine-grained objects.**
  - Keeping these objects alive consumes resources that prevent servicing more requests.
- **Tracking distributed object lifetimes across the network so that objects can release their proxies, increases network traffic, and degrades performance.**
- **Persistent distributed objects also interfere with load balancing to distribute processing evenly among computers.**
  - Stateless objects can be created or destroyed between method calls. State is stored in the non-distributed part of the application.
  - For example, an object that holds on to a database connection interferes with scalability.

Interoperability

- **Each distributed object system used its own transport mechanism.**
  - The various distributed object systems of the different vendors did not interoperate.
  - Although in principle CORBA was a vendor-neutral specification, in practice customers were locked into their particular choice of vendor implementation.
- **The various distributed object systems used different wire protocols.**
  - This is similar to having different procedure calling conventions between method calls.
- **The various distributed object systems require keeping non-standard Internet ports open for remote object invocation.**
  - These protocols are not transparent to the usual settings for firewalls.

Industry Standard Messaging

- **HTTP is a ubiquitous transport standard.**
  - Port 80 is almost always open on a firewall.
- **Since HTTP is a text-based protocol, XML was a natural choice for implementing an object-neutral messaging protocol.**
- **By exchanging messages instead of object method calls, applications built with different object models, and different implementation technologies can interact.**
- **Messaging technologies are more robust and scale better in distributed scenarios.**
- **This looser coupling, however, makes it harder for type information and transactional context to be transmitted.**
- **SOAP and WSDL have emerged as the fundamental specifications for Web services.**
- **Additional specifications such as WS-Security and WS-Addressing are designed to solve some additional issues associated with Web services.**
- **Future standards such as WS-Eventing (for publish and subscribe) will impact application development.**

Major Players in Web Services

- **The major players deeply involved and invested in Web services are Microsoft, IBM, Oracle, and many other companies.**
  - IBM, Microsoft, and BEA have been the originating forces behind the two key Web service specifications, **SOAP** and **WSDL**.
  - The World Wide Web Consortium, or W3C, controls some key Web services specifications as XML and XML Schema.
  - W3C has approved SOAP 1.2 as a recommendation. SOAP 1.1 and 1.0 are W3C notes.
  - WSDL 1.1 is a W3C note, not a recommendation. WSDL 2.0 is currently a W3C Candidate Recommendation (January 6, 2006).
  - The IETF (Internet Engineering Task Force) originally specified the Hypertext Transfer Protocol, or HTTP. This is now managed jointly with the W3C.
  - Other specifications such as WS-Security were developed under the aegis of the OASIS consortium.
  - Thus far, the Web services initiative is unique among distributed-computing standards efforts in that it operates under no single authority.
  - The organization works across the industry and standards organizations to respond to customer needs by providing guidance, best practices, and resources for developing Web services solutions.

Web Services Interoperability

- **Given the various specifications and their revisions it is necessary to define levels of interoperability.**
  - For example, SOAP 1.0, 1.1, and 1.2 systems exist, and not everything is backward compatible.
  - The current WSDL specification has some ambiguities. WSDL 2.0 is not a recommendation yet.
  - WS-Security is a newly approved specification, and not supported everywhere yet.
- **The Web Services Interoperability Organization (WS-I) is an open, industry organization chartered to promote Web services interoperability by defining levels of compliance.**
- **The Basic Profile 1.1 defines a set of guidelines for minimal interoperability of Web services.**
  - In addition the major vendors have created reference implementations that correspond to the Basic Profile 1.1.

Benefits of Web Service Integration

- **Let's look at some business advantages of Web services.**
  - Web services offer new business opportunities by making it easy to connect with partners. (Business to Business Integration).
  - Reduce application development and maintenance costs. (Save time and money).
  - Create new sources of revenue by making the present useful functionality into Web services.
  - Web services provide a solution to systems interoperability problems.
  - Implement cross-platform, program-to-program communications (application integration).
  - Deliver significantly more personal, integrated experiences to users using Web services.
  - Web services connect information, applications, people, systems and devices.

SOAP

- **SOAP (Simple Object Access Protocol) is a lightweight protocol for the exchange of information in a decentralized, distributed environment.**
  - SOAP is a protocol that defines how to send messages in a platform-independent manner using XML.
  - SOAP, the Simple Object Access Protocol, uses XML as a wire protocol to describe how the data and its associated type definitions are transmitted.
  - SOAP was developed by Microsoft, IBM and others, and then handed over to the W3C for further development.

WSDL

- **The WSDL (Web Service Description Language) is an XML-based grammar for describing the syntax of Web services, their functions, parameters, and return values.**
  - WSDL defines the methods and the data associated with a Web service.
  - Since WSDL is XML-based, it is both human and machine readable. However WSDL is designed to be used by machines for automated implementation of interface contracts.
- **WSDL can be thought of as the IDL of Web Services.**

WSDL Section	*Use*
Types	**Defines types**
Messages	**Abstract message signature**
Operations	**Abstract method definition**
Port Type	**Abstract interface based on operations**
Binding	**Interface and method implementation**
Port	**Associates binding with a specific address (instantiated object)**
Service	**Collection of ports (component: is a collection of instantiated objects)**

UDDI

- **The UDDI (Universal Description, Discovery, and Integration) specification defines a SOAP-based Web service for locating WSDL-formatted protocol descriptions of Web services**
  - It is the standard way to publish and discover information about Web services).
- **UDDI is a business registry which allows businesses and developers to programmatically locate information about Web services exposed by other organizations.**
  - At present, UDDI is not a standard, but is a joint initiative of businesses and vendors.
  - The latest UDDI Version 3 has some unique features such as multi-registry topologies, increased security features, improved WSDL support, a new subscription API and core information model advances.

Web Services Business Models

- **Many companies are using XML Web services to improve the way they do business, cut development costs, and increase customer satisfaction.**
  - Web services are a powerful business tool, allowing unprecedented customer integration and partnering opportunities.
  - A lot of companies are espousing Web services to simplify internal business process and to distributing their services and products.
  - Let us investigate some business models of Web services.

Business Models Classified

- **B2C, or business-to-consumer, is a Web business catering to individual consumers.**
  - Amazon.com and eCommerce sites in general are classic examples.
- **B2B, or business-to business, is a Web business catering to other businesses.**
  - Suppliers and vendors to manufacturing companies are examples of B2B businesses.
- **Classic B2C and B2B have human interaction to Web pages.**
- **Web services facilitate two new models:**
  - S2C, or service-to-consumer, provides a programmatic interface to a consumer-type function.
  - S2B, or service-to-business, provides a programmatic interface to a business-type function served by one business to another business.

Service Oriented Architecture (SOA)

- **Objects do not always easily reflect real-world processes and relationships.**
- **Distributed object systems have trouble evolving to reflect the dynamic nature of customer requirements and businesses.**
- **Building applications out of services allows for a more flexible architecture in building complex applications, especially if the services are owned by different divisions, or even different businesses.**
- **Service orientation allows you to build applications out of new services, services cleaved out of old applications, or applications wrapped as services.**
- **SOA is important when applications have to cross trust boundaries.**
- **Object technology can be used in implementing services.**
- **Web services are the natural implementation technology for SOA.**

Services Are Independent

- **A service is composed of code and data.**
  - Completely separate from another services' code and data.
- **Services are connected through messaging.**
  - MQ Series technology has been around for a long time
  - Messages are defined by contracts and XML Schema.
  - The order of the message is part of the contract.
  - Implementation is not part of the contract.
- **Boundaries are explicit**
  - Code, data, database tables are either within or outside of the service, there is no sharing
  - Database transactions are not shared among services.
  - Failure isolation is mandatory.
- **A service is deployed independently of other services.**
- **Service compatibility is based on policy.**
  - WS-Policy allows services to specify polices such as how authentication should be done.
  - Policy is a set of assertions that a Service expects to be true.
- **A Service is separate from the User Interface.**

ASP.NET Web Services

- **ASP.NET Web services are built on top of ASP.NET; therefore you can benefit from the features of ASP.NET to build Web services.**
  - Distinctively, ASP.NET not only takes advantage of performance enrichments found in the .NET Framework and the common language runtime, it has also been designed to offer significant performance improvements over ASP and other Web development platforms.
- **ASP.NET Web services facilitate Web services' accessing the many features of the .NET Framework, such as authentication, caching, tracing and state management.**
  - Developers can focus on creating or accessing Web services without needing to write infrastructure code, because ASP.NET and the .NET Framework are the foundation for Web services in managed code.
- **Visual Studio 2010 is a complete set of development tools for building Web applications, Web services, desktop applications, and mobile applications.**
  - Therefore with the assistance of Visual Studio, you can quickly create, consume and deploy Web services.

Alternative Technologies

- **Within the Microsoft platform, there are several alternative technologies to ASP.NET Web Services (asmx).**
  - ASP.NET Web Services is a good match to SOA because you get explicit service boundaries, and has the scalability features of ASP.NET.
- **Web Service Enhancements (WSE) is designed to take advantage of leading edge Service Oriented Architecture technology for ASMX.**
- **.NET Remoting**
  - Distributed objects when both ends host the CLR.
  - Works best for AppDomain to AppDomain communication within the same process.
  - Can integrate with an existing proprietary protocol using extensibility features.
- **Enterprise Services**
  - Need the features of COM+.
  - Communicate with objects on local machine when you have performance issues with ASMX and WSE
- **MSMSQ**
  - Asynchronous Messaging

Windows Communication Foundation

- **Windows Communication Foundation (WCF) is a new service-oriented programming framework for creating distributed applications.**
  - It was previously known as ‘Indigo’ and is part of .NET 3.5 and .NET 4.0
- **WCF is designed to provide one mechanism for building connected applications:**
  - Within app domains
  - Across app domains
  - Across machines
- **WCF builds upon and extends existing ways of building distributed applications:**
  - ASMX Web services, .NET Remoting, COM, MSMQ.
- **All these do the same basic job (connecting elements in distributed applications) but they are very different at the programming level, with complex APIs and interactions.**
  - WCF provides one model for programming distributed applications. Developers only need to learn one API.
- **WCF leverages existing mechanisms.**
  - It uses TCP, HTTP and MSMQ for transport.

WCF and REST

- **An alternative to SOAP for Web services is Representational State Transfer or REST.**
- **REST is based on the architectural style of the Web.**
  - You call a REST Web service simply by making an HTTP request.
  - There is no protocol such as SOAP that has to be implemented on top of HTTP.
- **This makes REST services easy to implement on both the server and client.**
- **However, ASP.NET is geared exclusively towards SOAP Web services.**
- **With WCF you can create REST Web services as well as SOAP Web services.**
- **Visual Studio 2010 and .NET 4.0 make it especially easy to implement REST Web services by using some special Visual Studio templates that are available online.**
  - We discuss REST Web services in Chapter 11.

Summary

- **Web services are used to implement a distributed computing architecture.**
- **Web services provide some application logic, and are available to any number of potentially dissimilar systems through the use of standards, such as SOAP, XML and HTTP.**
- **WSDL is used for describing the Web services, and UDDI provides a registry service for publishing and locating Web services.**
- **Web services trim down development cost and maintenance cost, and provide a solution to interoperability issues.**
- **Web services can create new business opportunities and used in various businesses models such as S2C and S2B scenarios.**
- **Service Oriented Architecture (SOA) is the future of enterprise application architecture, and ASP.NET Web Service is an effective current Microsoft technology match for implementation.**
- **Windows Communications Foundation (WCF) is the newest Microsoft technology for implementing SOA.**
- **With .NET 4.0 and Visual Studio 2010 it is easy to implement REST Web services.**

Chapter 10

Introduction to WCF

Introduction to WCF

Objectives

After completing this unit you will be able to:

- **Explain how WCF unites and extends existing distribution technologies.**
- **Understand the importance of address, binding and contract to WCF services.**
- **Create a simple WCF service hosted in IIS.**
- **Implement a client of a WCF service.**
- **Understand the issues of interoperability between Web services created using ASP.NET and those created using WCF.**
- **Understand and implement data contracts.**

What Is WCF?

- **Windows Communication Foundation (WCF) is a new service-oriented programming framework for creating distributed applications.**
  - It was previously known as ‘Indigo’ and is part of .NET 3.5 and above.
- **WCF is designed to provide one mechanism for building connected applications:**
  - Within app domains
  - Across app domains
  - Across machines
- **WCF builds upon and extends existing ways of building distributed applications:**
  - ASMX Web services, .NET Remoting, COM, MSMQ.
- **All these do the same basic job (connecting elements in distributed applications) but they are very different at the programming level, with complex APIs and interactions.**
  - WCF provides one model for programming distributed applications. Developers only need to learn one API.
- **WCF leverages existing mechanisms.**
  - It uses TCP, HTTP and MSMQ for transport.

WCF Services

- **When using WCF, you create and consume services.**
  - A service comprises a set of related operations, which the programmer sees as method calls.
- **Services are described by metadata, which clients can use to determine what operations are available, and how the service can be contacted.**
  - Metadata for WCF services is similar to the WSDL used by web services.
- **Clients and services exchange messages.**
  - A client (which can be another service) communicates with a WCF service by sending and receiving messages. WCF uses SOAP as its messaging mechanism, and SOAP messages can be sent using a number of transports.
  - Using SOAP does not imply that WCF communication is inefficient; efficient binary encodings are employed whenever possible.
- **WCF supports several transports out of the box.**
  - TCP, HTTP, HTTPS and MSMQ.
  - Custom transports can be added.

WCF Services

- **WCF supports the WS-* family of Web service protocols.**
  - The WS-* family of protocols have been developed by various bodies (including OASIS and W3C) to provide features such as security, transactions and reliable messaging to web services.
- **WCF is very good at interop.**
  - Support for a wide range of transports, encodings and the WS-* protocols means that WCF services can interoperate with a wide range of platforms and technologies, including J2EE and web services using WS-* protocols.
- **WCF supports the implementation of REST-based Web services.**
  - REST stands for Representational State Transfer, a style of software architecture discussed in Chapter 11.
- **WCF provides a foundation for service orientation.**
  - WCF helps developers write distributed applications in which loosely coupled services are called by clients and one another.

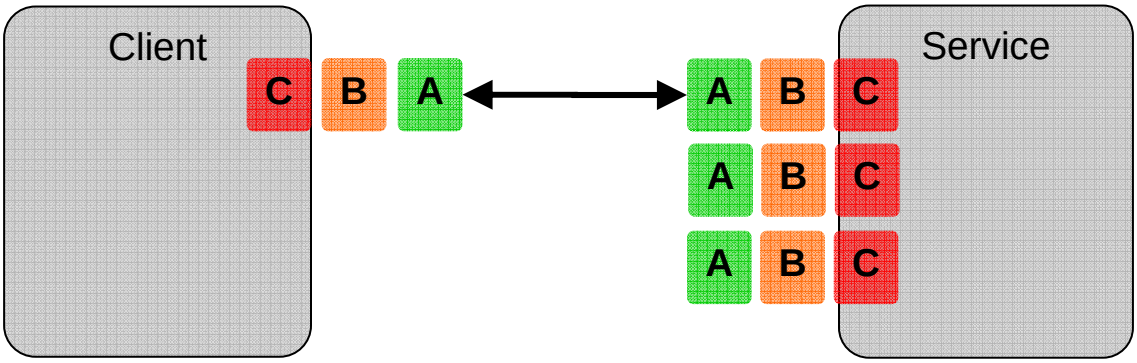
Service Orientation

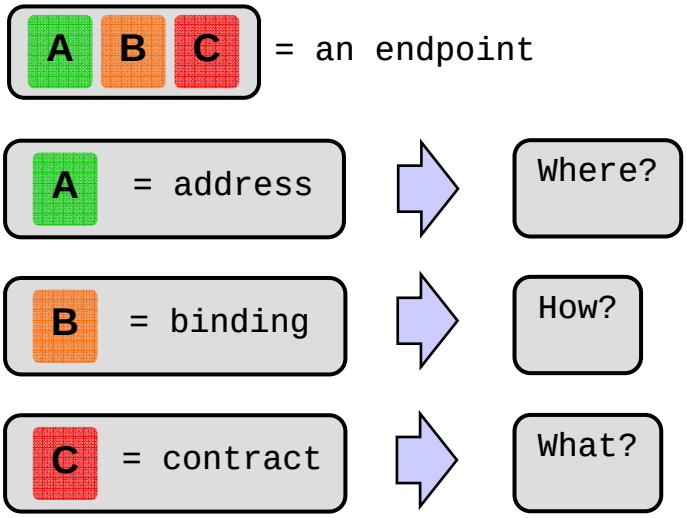
- **Service orientation is characterized by four concepts.**
- **Boundaries are explicit**
  - The boundary between client and service is explicit and highly visible, because calls are made via SOAP messages. There is no pretending – as there is in DCOM and Java's RMI – that you are simply playing with a remote object, and can thus ignore the cost of remoting.
- **Services are autonomous**
  - Services are independent entities that each have their own life cycle; they may have been developed completely independently of one another. There is no run-time making sure that services work well together, and so services must be prepared to handle failure situations of all sorts.
- **Share schemas and contracts, not classes**
  - Services are not limited to implementation in OO languages, and so service details cannot be provided in terms of classes. Services should share schemas and contracts, and these are typically described in XML.
- **Use policy-based service compatibility**
  - Services should publish their requirements (i.e. Requiring message signing or HTTPS connections) in a machine readable form. This can be used at runtime to ensure compatibility between service and client.

WCF and Web Services

- **Even though it uses SOAP messaging, WCF is more than simply another way of writing Web services.**
  - WCF can be used to write traditional Web services, as well as more sophisticated services that can use the WS-* protocols. But the design of WCF means that it provides a far more general solution to distribution than Web services.
- **WCF can use several transports.**
  - Web services tend to use HTTP or HTTPS, while WCF is configured to use TCP, HTTP, HTTPS and MSMQ. It is also possible to add new transports, should the need arise.
- **WCF can work throughout the enterprise.**
  - WCF services can be hosted in-process, by a Windows Service or IIS. Message exchange will be optimized to use the most efficient method for exchanging data for a particular scenario.
- **WCF is good at interop.**
  - WCF services can interoperate with a number of different platforms and technologies.
- **WCF is highly customizable.**
  - It is possible to customize almost every part of WCF, adding or modifying transports, encodings and bindings, and plugging in new ‘behaviors’ that affect the way WCF services work.

WCF = ABC





Address, Binding, Contract

- **An *address* defines where a service can be found.**
  - It will often be an HTTP address, although other addressing schemes are supported.
- **A *binding* defines how a service can be contacted**
  - Via HTTP, TCP, MSMQ or some custom mechanism.
- **A *contract* defines what a service can do.**
  - In terms of method calls, their arguments and return types.
- **A combination of an address, a binding and a contract is called an *endpoint*.**
  - A service can expose more than one endpoint, and endpoint data can be made available to clients in the form of metadata.

Example – Echo Service

- **A contract defines a set of operations that a service supports.**
  - Define a contract as an interface decorated with the **ServiceContract** attribute.
  - Decorate operations with the **OperationContract** attribute.

```
[ServiceContract]
public interface IEchoService
{
   [OperationContract]
   string GetGreeting(string value);
}
```

- **A service class implements the interface.**
  - And so it has to implement all the operations defined in the contract.

```
public class EchoService : IEchoService
{
   public string GetGreeting(string value)
   {
      return "Hello, " + value;
   }
}
```

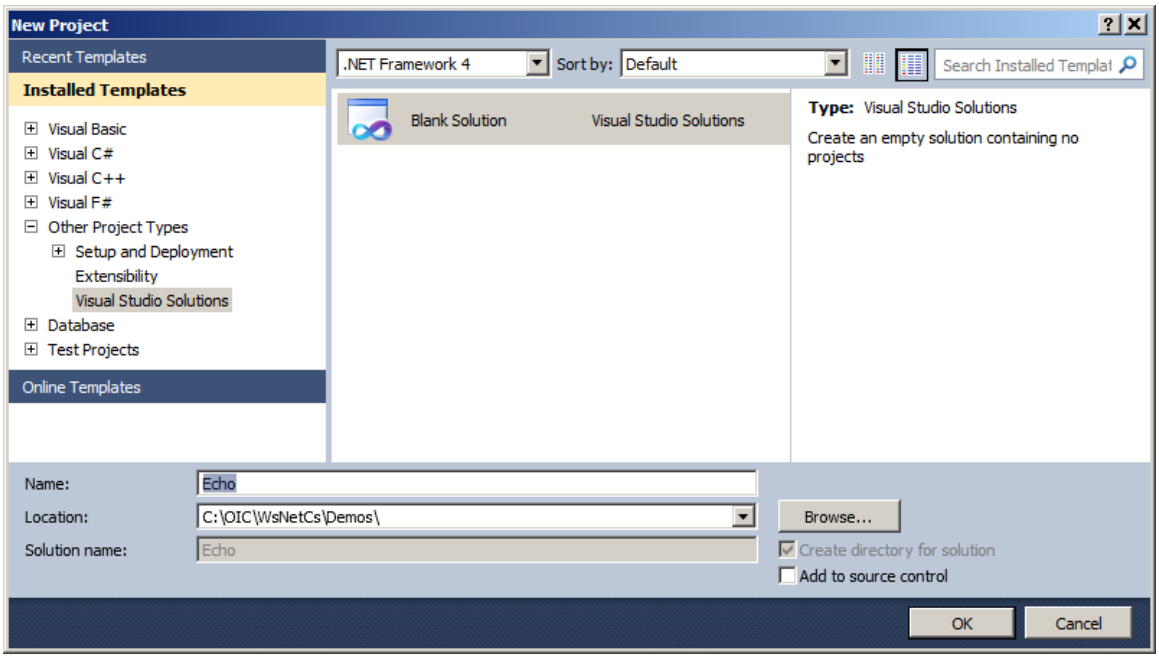
- **Note that this code says nothing about how the client communicates with the service.**
  - It is only the 'C' of the service 'ABC'.

Hosting Services

- **There are three ways to host a WCF service.**
- **Self-hosting in an EXE.**
  - Use any type of EXE: Console application, Windows application (Windows Forms or WPF), or Windows Service.
  - Need to manage service lifecycle yourself.
- **Hosting in IIS.**
  - IIS will manage the service lifecycle for you, starting the service when the first request comes in.
  - You can only use HTTP and port 80.
  - Configure the service using a **.svc** file.
  - The examples in the chapter use IIS hosting.
- **Hosting in Windows Activation Service (WAS)**
  - WAS is a new feature that is part of Vista and Windows 7.
  - Similar advantages to hosting in IIS, but you can use other transports and ports as well.
  - WAS also uses **.svc** files.
  - It is the preferred way of hosting services on Windows 7.

Demo – Echo Service

- **We'll create an "Echo" service, hosted in IIS.**
- **Our solution will have two projects:**
  - A website implementing the service and exposing it via HTTP
  - A Windows client that invokes the service
- **Use Visual Studio 2010 to create a new blank solution *Echo* in the *Demos* folder[1].**

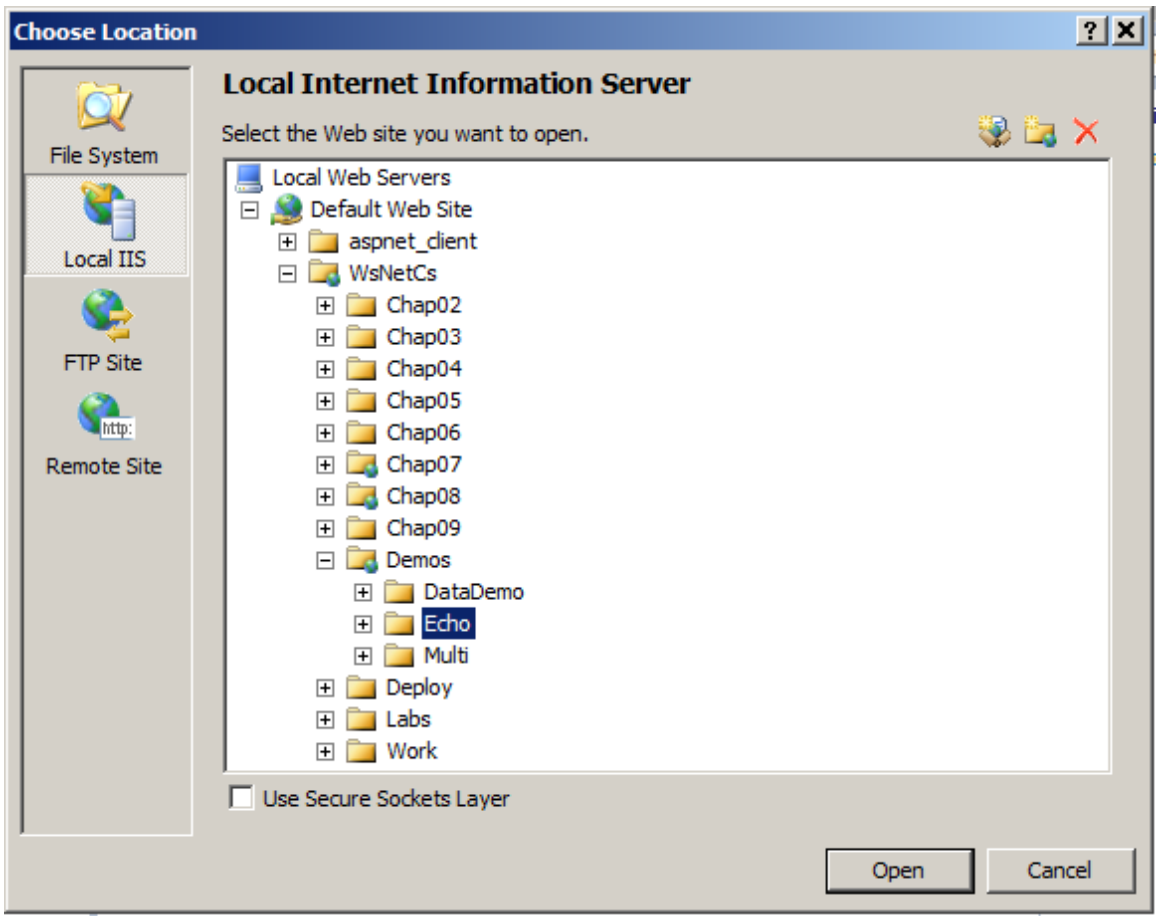


[1] If you are using Vista or Windows 7, you should run Visual Studio 2010 as Administrator.

A Website for the Service

- **We will host our WCF service using IIS.**
- **To do this, we'll add a website project to our solution.**

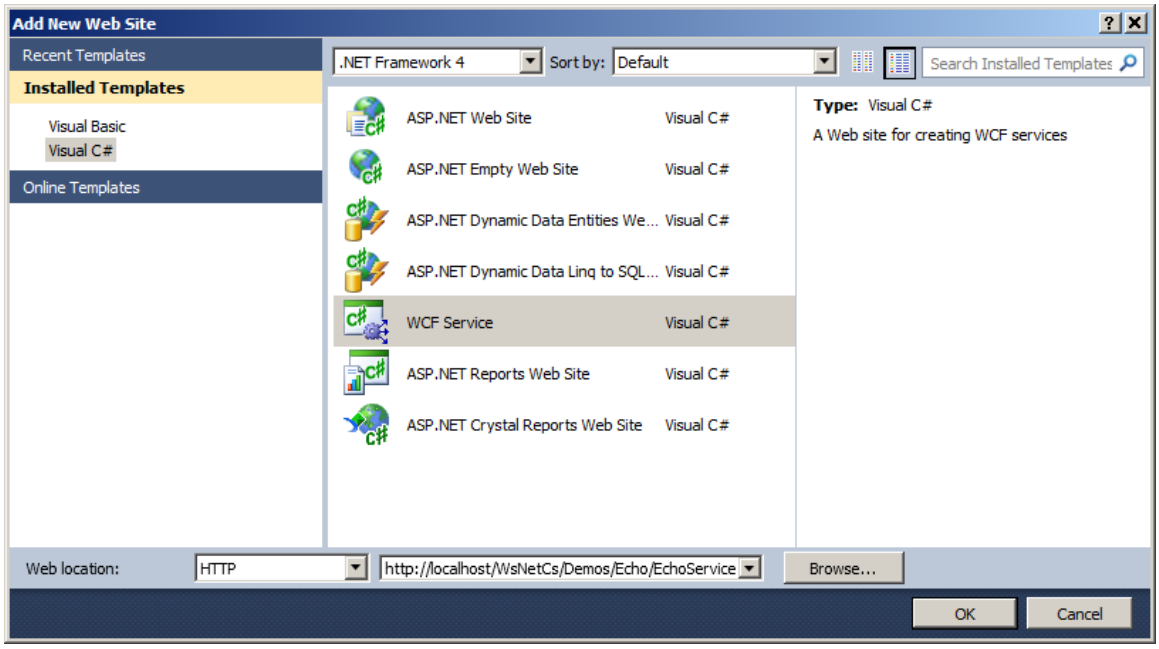
1. In Solution Explorer, right-click on the solution and choose Add | New Web Site.
2. For template choose WCF Service. For location select HTTP. Browse to the **Demos\Echo** folder, which is under the **WsNetCs** virtual directory.



3. Click Open. A message box comes up about the Web site not being marked as an application in IIS. Click Yes.

A Website for the Service (Cont'd)

4. Type in **EchoService** at the end of the path.



5. Click OK. Examine the generated website files:

   – **IService.cs** contains a sample service contract (in App_Code folder).

   – **Service.cs** contains a sample service implementation (in App_Code folder).

   – **Service.svc** provides a .svc endpoint, which helps the service model find the correct service type to host.

   – **Web.config** provides service model configuration settings for the sample service.

A Website for the Service (Cont'd)

6. Delete the supplied code in **IService.cs** and type in the following:

```
[ServiceContract]
public interface IEchoService
{
   [OperationContract]
   string GetGreeting(string value);
}
```

7. Delete the supplied code in **Service.cs** and type in the following:

```
public class EchoService : IEchoService
{
   public string GetGreeting(string value)
   {
      return "Hello, " + value;
   }
}
```

8. Edit the file **Service.svc** to specify the service type, based on the class name of the service implementation. (This file is analogous to the **.asmx** file in ASP.NET Web services.)

```
<%@ ServiceHost Language="C#" Debug="true"
Service="EchoService"
CodeBehind="~/App_Code/Service.cs" %>
```

Service Configuration

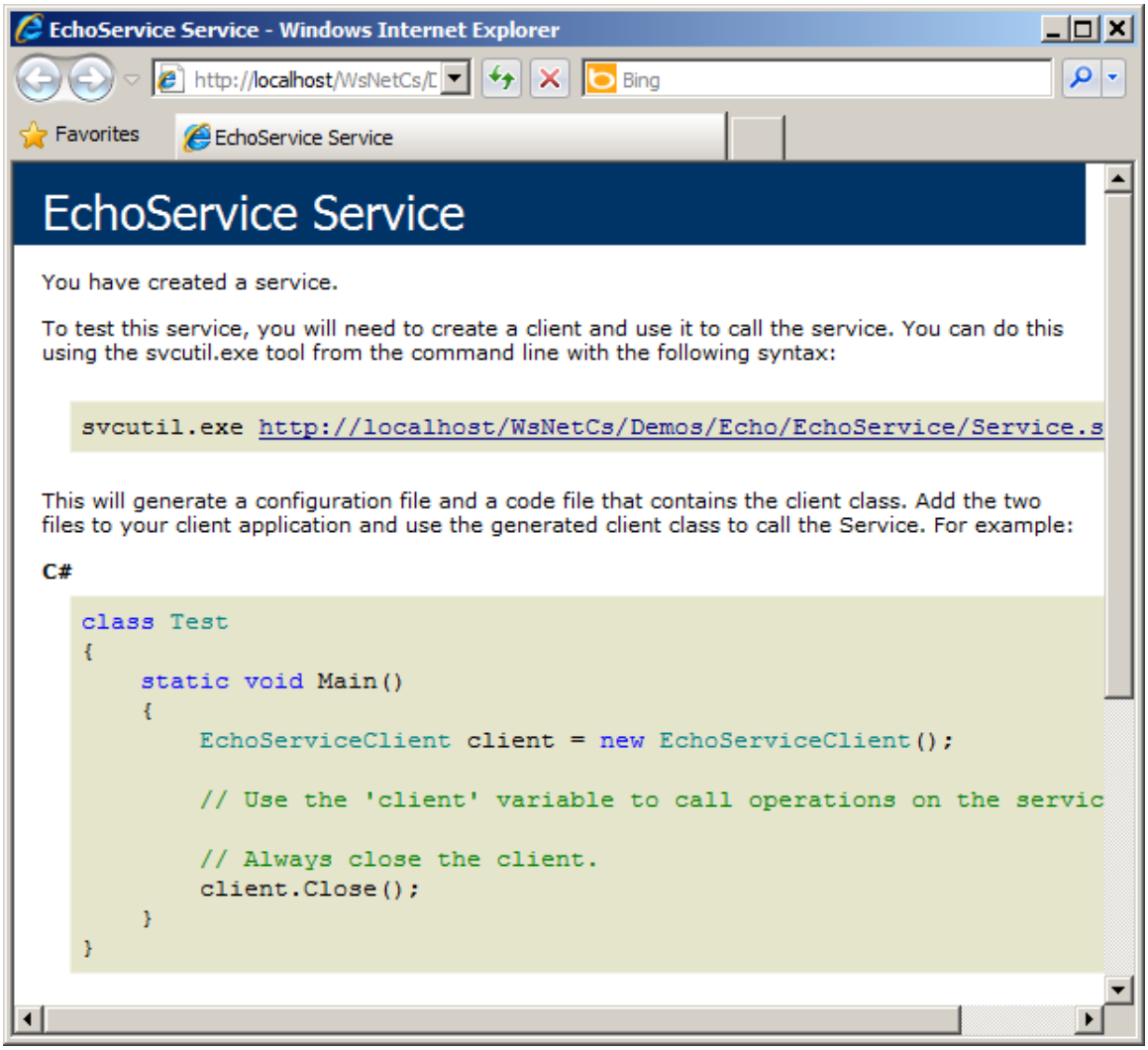
- **Services can be configured in code or using config files**
  - XML configuration files are preferred, because they make it easier to reconfigure services, and to reuse services in different contexts without needing to rebuild the service code.
- **When configuring IIS-hosted services, information on the service can be included in the *Web.config* file for the website.**
  - Data is placed in the **<system.ServiceModel>** element

```
<?xml version="1.0"?>
<configuration>
  ...
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <serviceMetadata httpGetEnabled="true"/>
          <serviceDebug
           includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <serviceHostingEnvironment
      multipleSiteBindingsEnabled="true" />
  </system.serviceModel>
  ...
```

  - Configuration in .NET 4.0 is simpler, and in many cases, such as this one, the configuration file generated by Visual Studio works out of the box without any changes.

Examining the Service in the Browser

- **Since our service is hosted in IIS, we can examine the *Service.svc* file in the browser.**
  - In Solution Explorer, make **Service.svc** the Start Page. Build the solution.
  - Run the solution. When Internet Explorer comes up, you should see a page similar to this:



WCF Clients

- **As in all common distribution technologies, clients interact with services through proxies.**
  - The proxy hides details of the communication mechanism being used from the client code.
  - The proxy implements the same interface as the service, so that the client can use exactly the same calls.
- **Proxies are created using metadata provided by the service.**
  - Note that the page shown in the previous page gives instructions for generating a proxy.
- **The ABC (address, binding, contract) information provided by the service can be used to construct a proxy that**
  - Knows where to contact the service.
  - Implements code to use the appropriate communication mechanism (eg. HTTP call versus TCP).
  - Implements the operations defined in the service interface.

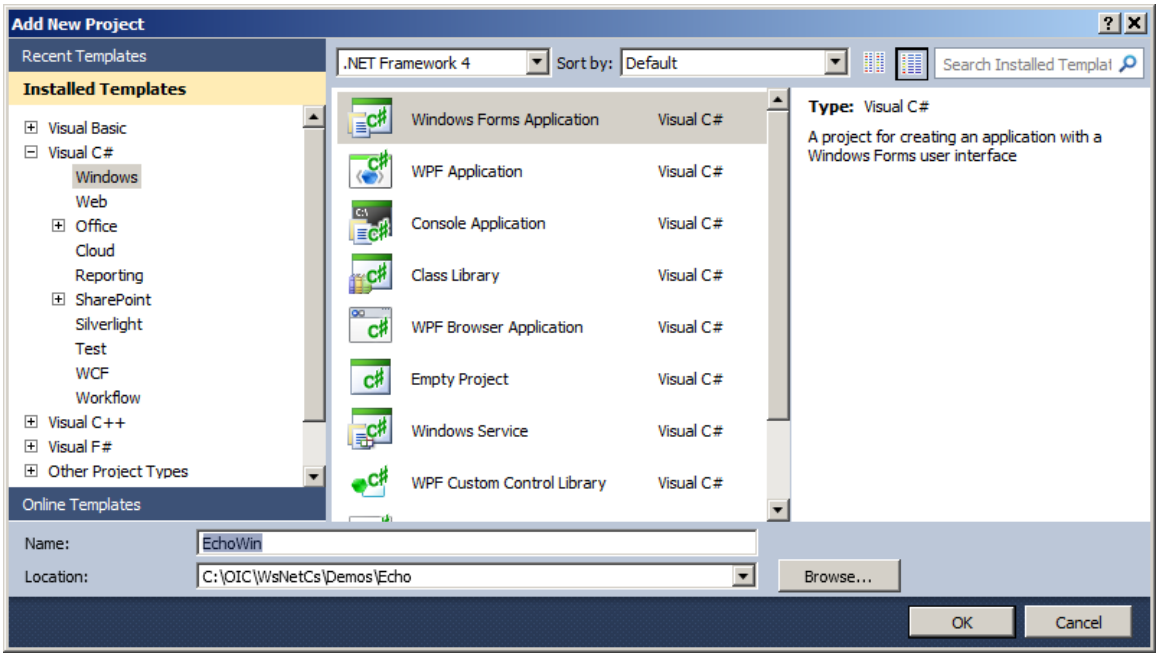
Creating WCF Clients

- **Visual Studio 2010 can create clients to WCF services in the same way as it creates them for Web services.**
  - The configuration file contains endpoint details.
- **You can use the *svcutil*.exe command line tool to generate proxies.**
  - Useful when not using Visual Studio.
  - It also provides more control over the proxy generation process.
- **The client code creates a proxy and calls methods**
  - Use exception handling, because lots can go wrong in distributed applications.

Demo – A Client for Echo Service

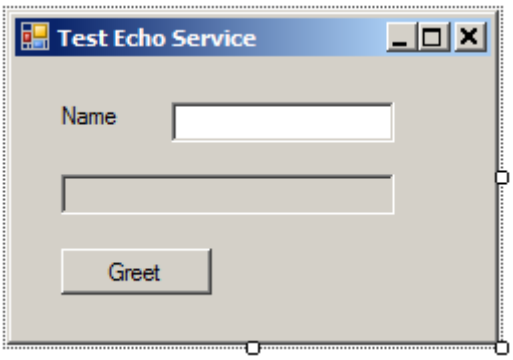
- **We'll create a Windows Forms client.**

1. Add a second project to the solution, this time a Windows Forms Application. Specify **EchoWin** as the name of your new project.



Client Demo (Cont'd)

2. Provide the following user interface.

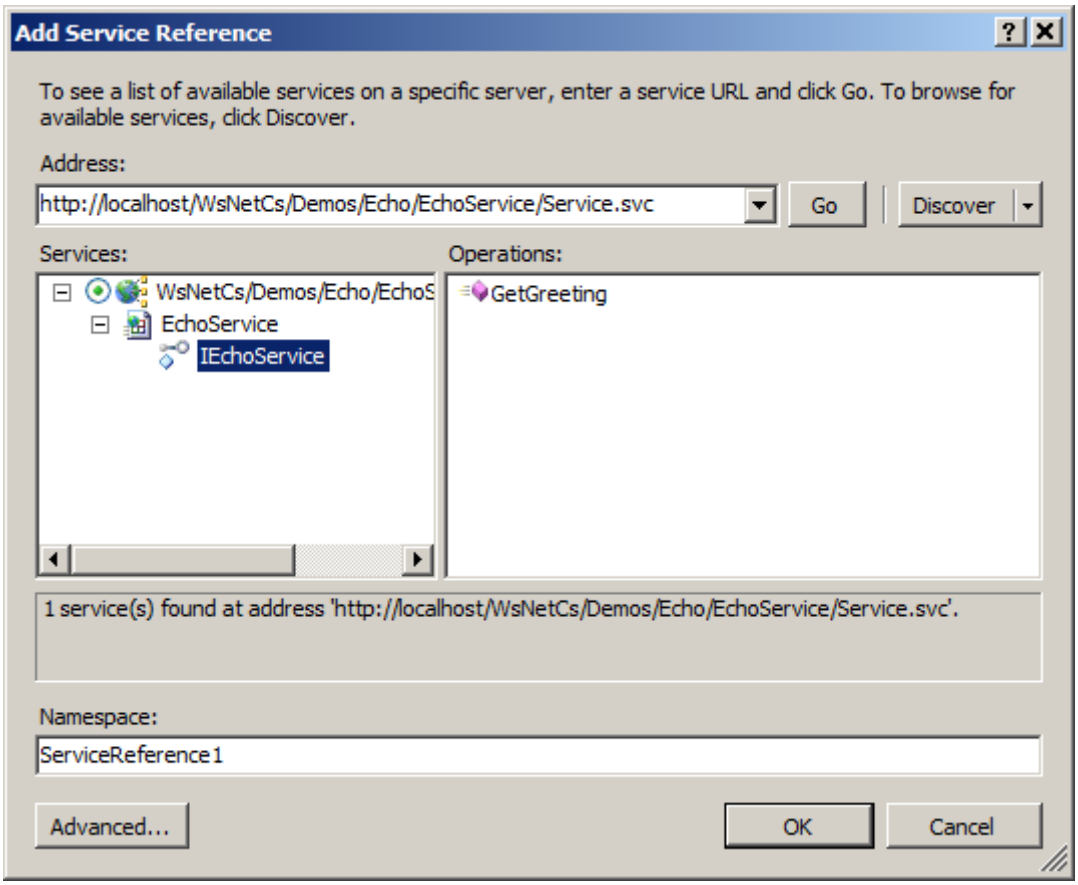


3. In Solution Explorer, right-click over the **EchoWin** project and select Add Service Reference from the context menu.

4. Click the Discover button to find the services in the solution.

Add Service Reference Dialog

5. In the tree view expand the services and interfaces.



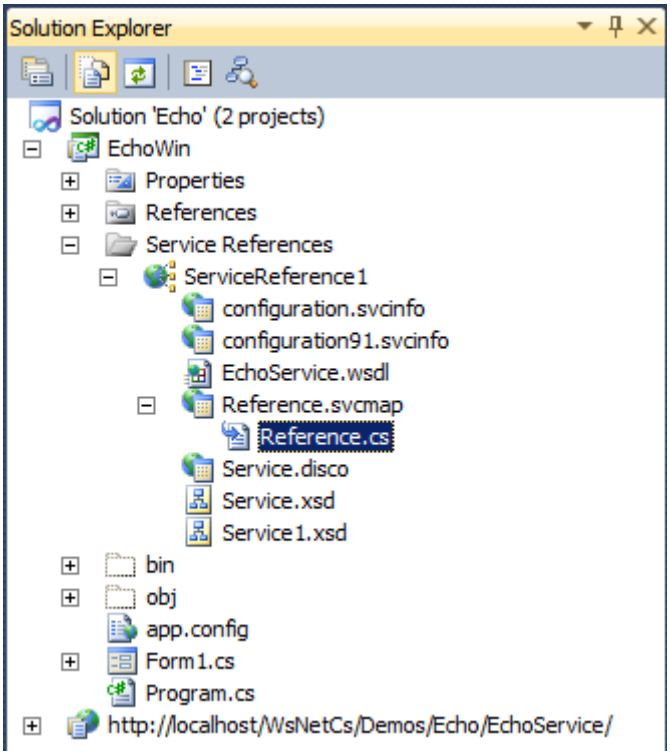
6. Click OK.

Client Demo (Cont'd)

7. In **EchoWin** verify that there is a reference to **System.ServiceModel**. (Added automatically when the service reference is added.)

8. Build the solution to make sure you get a clean compile for both projects.

9. In Solution Explorer, click the Show All Files button. Observe there is now a file **Reference.cs** under Service References.



10. Examine the file **Reference.cs**. This contains the code for the proxy. Note that the proxy class is **EchoServiceClient**, in the namespace **EchoWin.ServiceReference1**.

Client Demo (Cont'd)

11. In **Form1.cs** import the namespace **EchoWin.ServiceReference1**.

```
using EchoWin.ServiceReference1;
```

12. Add a handler for the Greet button and provide the following code to invoke the **GetGreeting()** method on the proxy.

```
private void btnGreet_Click(object sender,
EventArgs e)
{
   EchoServiceClient client =
      new EchoServiceClient();
   txtGreeting.Text =
      client.GetGreeting(txtName.Text);
   client.Close();
}
```

13. Make **EchoWin** the startup project.

14. Build and run. You should be able to enter any name and get back a greeting from the service.

15. A copy of this completed demo is provided in the **Echo** folder in the chapter directory. It has been modified to run in its new location. (Again, in Windows 7 you must run Visual Studio as Administrator.)

Lab 10A

Creating a Simple Service and Client

In this lab, you will use Visual Studio 2010 to create a simple WCF service that is hosted in IIS. You will then create a client Console application, use Add Service Reference to create a proxy for the service, and demonstrate that the client can contact the service.

Detailed instructions are contained in the Lab 10A write-up at the end of the chapter.

Suggested time: 45 minutes

Interop with ASMX Web Services

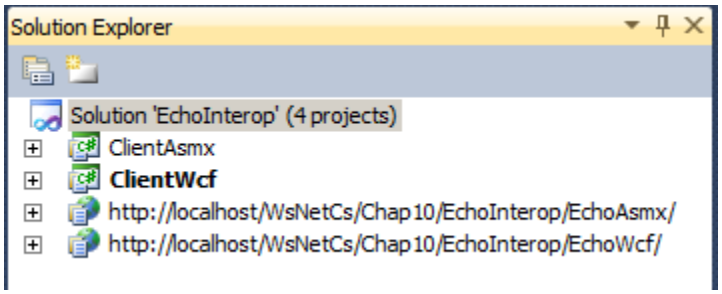
- **In this course we have learned two technologies for developing Web services:**
  - ASP.NET, sometimes called ASMX Web services after the standard file extension **.asmx**
  - WCF, where the standard file extension is **.svc**.
- **With .NET 2.0 you can only create ASMX Web services and clients.**
- **With .NET 4.0 you can create both kinds of services and clients.**
- **To ensure interoperability between the two kinds of Web services, they should conform to Basic Profile 1.1.**
  - With ASP.NET 2.0 we specify conformance to Basic Profile 1.1 by use of the **WebServiceBinding** attribute.

```
[WebServiceBinding(ConformsTo =
WsiProfiles.BasicProfile1_1)]
```

  - With WCF we can specify conformance by using the binding **basicHttpBinding** in place of **wsHttpBinding**.
  - The good news with .NET 4.0 is that conformance to Basic Profile 1.1 occurs out of the box with IIS hosting.

Interop Demonstration[2]

- **The solution *EchoInterop* in the chapter directory contains four projects.**



- **There are two different versions of an echo service:**
  - **EchoAsmx** is implemented using ASP.NET.
  - **EchoWcf** is implemented using WCF.
- **There are two client projects.**
  - **ClientAsmx** has Web References to both services.
  - **ClientWcf** has Service References to both services.

[2] To build and run this example in Windows 7 you must, as usual, start Visual Studio as Administrator.

EchoAsmx

- ***EchoAxmx* is implemented using .NET 2.0 ASP.NET technology.**

```
[WebService(Namespace = "http://oi.com/")]
[WebServiceBinding(ConformsTo =
WsiProfiles.BasicProfile1_1)]
public class EchoAsmx :
System.Web.Services.WebService
{
   [WebMethod]
   public string GetGreeting(string name)
   {
      return "Hello, " + name;
   }
}
```

- **It conforms to Basic Profile 1.1.**

```
[WebServiceBinding(ConformsTo =
WsiProfiles.BasicProfile1_1)]
```

EchoWcf

- **EchoWcf is implemented using WCF.**
  - Here is the contract:

```
[ServiceContract]
public interface IEchoWcf
{
   [OperationContract]
   string GetGreeting(string name);
}
```

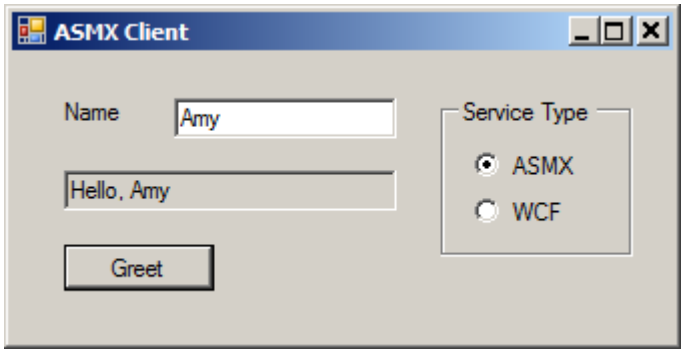
  - Here is the implementation:

```
public class EchoWcf : IEchoWcf
{
   public string GetGreeting(string name)
   {
      return "Hello, " + name;
   }
}
```

  - The simplified configuration of .NET 4.0 specifies the binding **basicHttpBinding**, so the service conforms to Basic Profile 1.1.

ASMX Client

- ***ClientAsmx* connects to the services using Web References.**
  - Radio buttons allow the user to choose between **EchoAsmx** and **EchoWcf**.

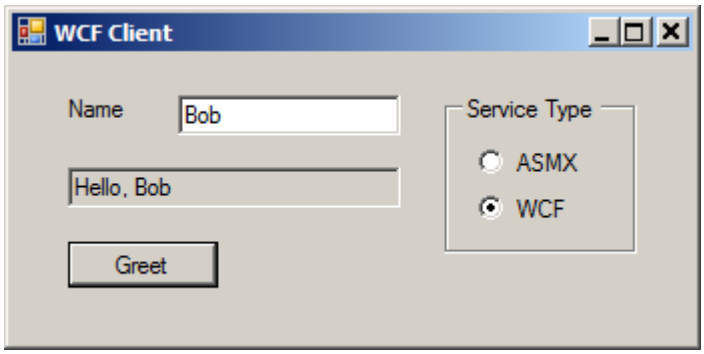


  - Code:

```
using ClientAsmx.ASMXlhost;
using ClientAsmx.WCFhost;
...
private void btnGreet_Click(object sender,
EventArgs e)
{
   if (radASMX.Checked)
   {
      EchoAsmx client = new EchoAsmx();
      txtGreeting.Text =
         client.GetGreeting(txtName.Text);
   }
   else if (radWCF.Checked)
   {
      EchoWcf client = new EchoWcf();
      txtGreeting.Text =
         client.GetGreeting(txtName.Text);
   }
}
```

WCF Client

- ***ClientWcf* connects to the services using Service References.**



  - Code:

```
using ClientWcf.ServiceAsmx;
using ClientWcf.ServiceWcf;
...
private void btnGreet_Click(object sender,
EventArgs e)
{
   if (radASMX.Checked)
   {
      EchoAsmxSoapClient client =
         new EchoAsmxSoapClient();
      txtGreeting.Text =
         client.GetGreeting(txtName.Text);
   }
   else if (radWCF.Checked)
   {
      EchoWcfClient client = new EchoWcfClient();
      txtGreeting.Text =
         client.GetGreeting(txtName.Text);
   }
}
```

Data Contracts

- **A data contract abstractly describes the data to be exchanged between a service and a client.**
- **A service is not tied to a specific platform, so you cannot pass data as CLR types across a service boundary.**
  - What is needed is a neutral format that can be converted to and from CLR types.
  - The neutral format is an XML schema definition.
  - To be exchanged, data must be converted to XML according to the schema, or **serialized**.
- **There are several serializers in the .NET Framework.**
  - By default, WCF uses the Data Contract Serializer to serialize and deserialize data.
- **All primitive .NET types, and many other Framework types as well, have default data contracts and can be serialized without any other preparation.**
- **Data types defined by the programmer must have the *DataContract* attribute applied to the class to be serializable.**
  - Furthermore, the **DataMember** attribute must be applied to each member that is to be serialized.

Data Contract Example

- **See *Courses\Step1* in the chapter directory.**
  - Examine the file **Course.cs**.

```
using System.Runtime.Serialization;

[DataContract]
public class Course
{
   [DataMember]
   public string courseNumber;
   [DataMember]
   public string title;
   [DataMember]
   public decimal price;
   public Course(string id, string name, decimal p)
   {
      courseNumber = id;
      title = name;
      price = p;
   }
}
```

  - The **DataContractAttribute** and **DataMemberAttribute** classes are in the **System.Runtime.Serialization** namespace.
  - There is a public constructor in the class as well as the data members. This constructor does not participate in any contract, but can be used locally by code within the service.

Operation Contract

- **The operation contract is specified in the file *IService.cs*.**

```
[ServiceContract]
public interface ICourseManager
{
   [OperationContract]
   Course[] GetCourses();
}
```

- **The contract is implemented in the file *Service.cs*.**

```
public class CourseService : ICourseManager
{
   public Course[] GetCourses()
   {
      List<Course> courses = new List<Course>();
      courses.Add(new Course(
         "411", "C# Essentials", 100m));
      courses.Add(new Course(
         "412", ".NET Framework Using C#", 150m));
      return courses.ToArray();
   }
}
```

Client Program

- **The Console project *CourseClient* implements the client program.**
  - **See Program.cs**.

```
static void Main(string[] args)
{
   CourseManagerClient client =
      new CourseManagerClient();
   Course[] courses = client.GetCourses();
   ShowCourses(courses);
   client.Close();
}
private static void ShowCourses(Course[] courses)
{
   foreach (Course c in courses)
      Console.WriteLine("{0}  {1, -25}  {2, 7:C}",
         c.courseNumber, c.title, c.price);
   Console.WriteLine("--------------------------");
}
```

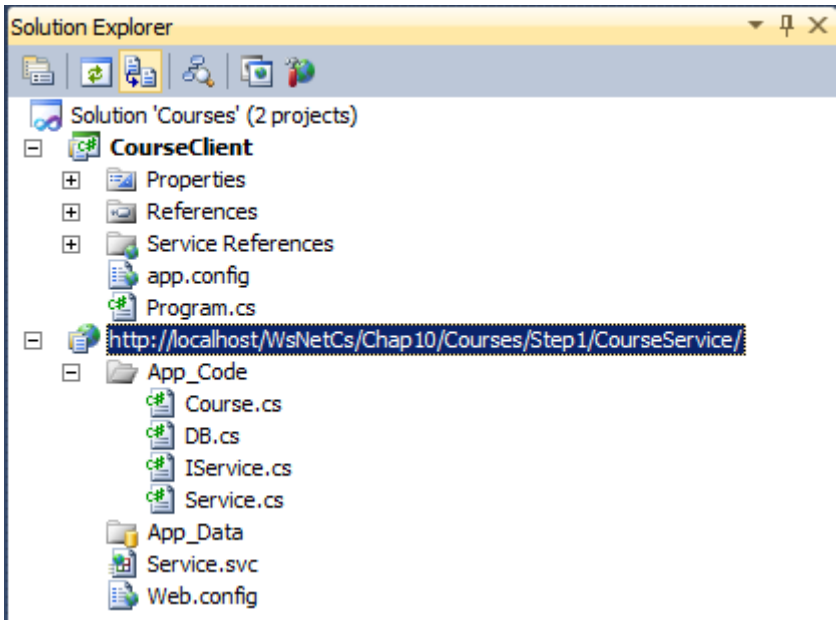
  - Here is the output:

```
411  C# Essentials              $100.00
412  .NET Framework Using C#    $150.00
--------------------------------------
```

Deploying a WCF Web Service

- **You can deploy a WCF Web service in the same manner that you deploy ASP.NET Web services.**
  - Use the Copy Web Site facility in Visual Studio.
- **As a demo, let's create a copy of the Step1 version of the *Courses* example in the *Demos* folder.**

1. In **Demos** create a new folder **Courses**.
2. Copy the contents of **Chap10\Courses\Step1** to **Demos\Courses**.
3. Open the solution **Courses.sln** in **Demos\Courses**.
4. Observe that the Web Site at this new location is still the original Web Site of the Step1 version.

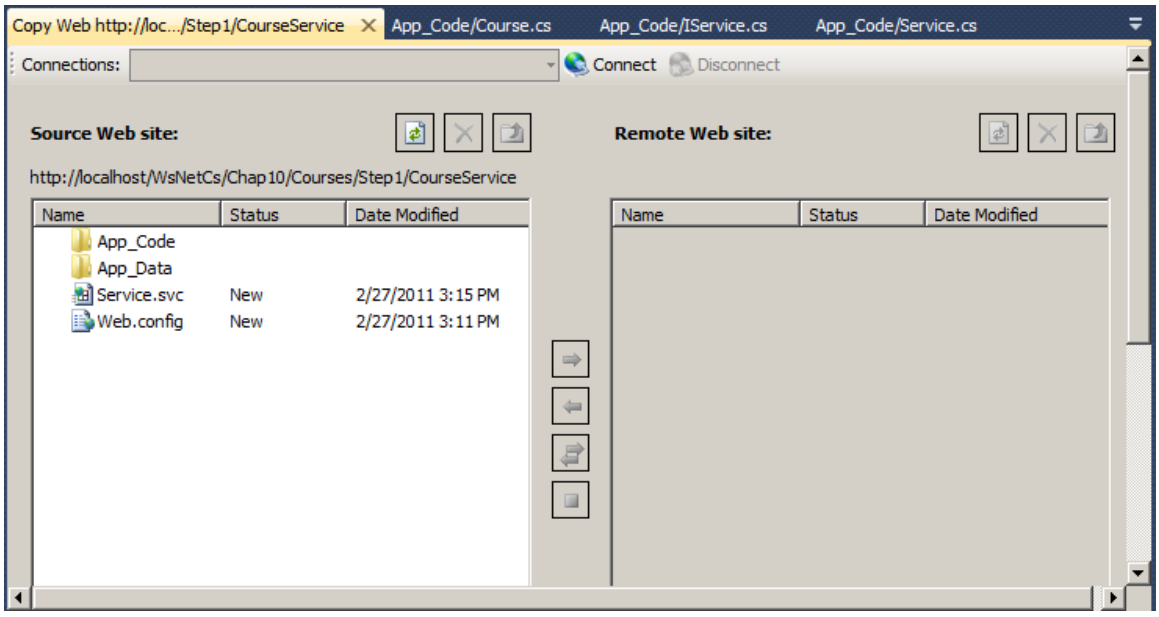


5. Remove this Web Site in Solution Explorer and delete the contents of the folder **Demos\Courses\CourseService**.

Deployment Demo

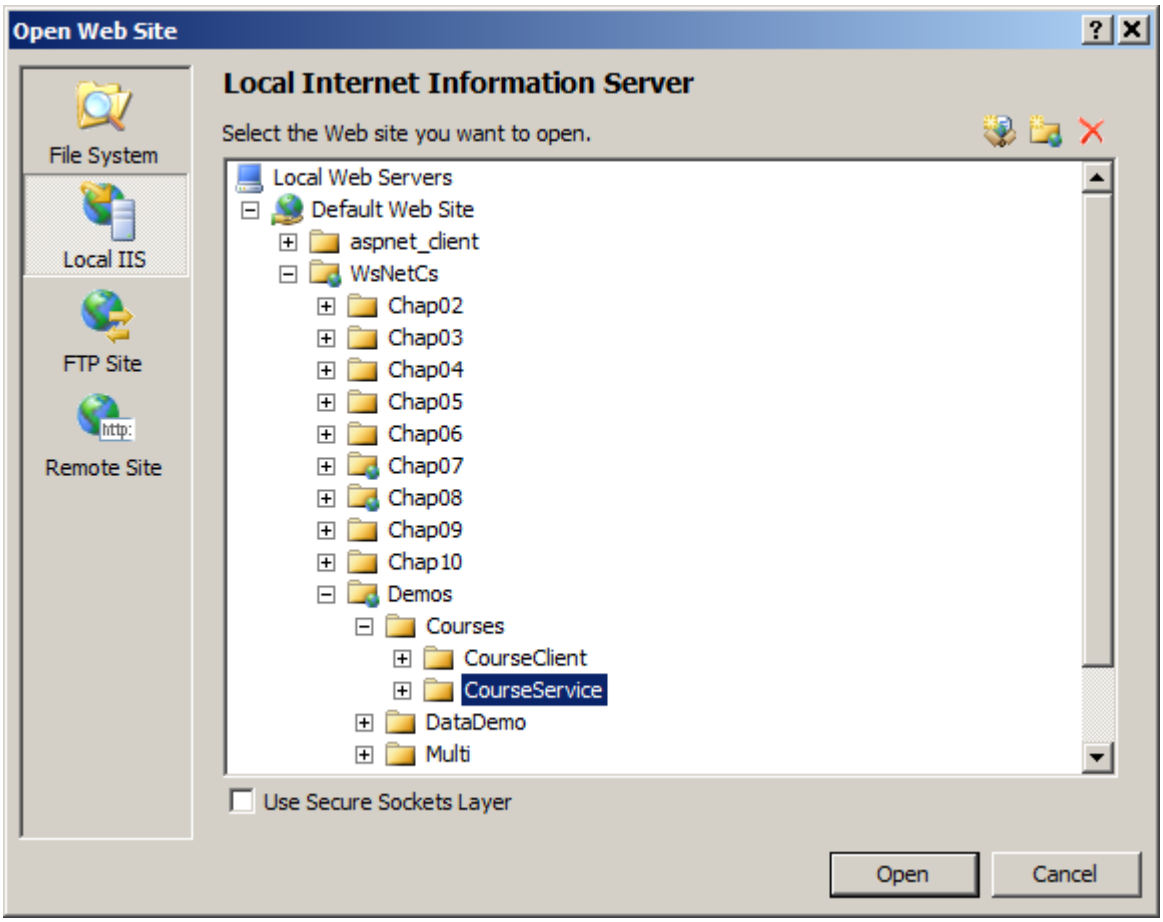
6. Open up the solution in **Chap10\Courses\Step1**.

7. In Solution Explorer right-click on the Web Site. Select Copy Web Site from the context menu.



Deployment Demo (Cont'd)

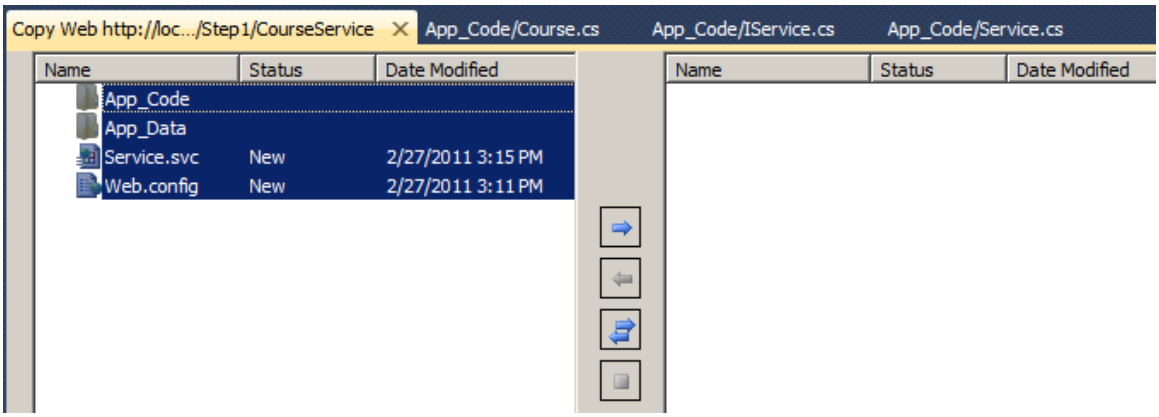
8. Click the Connect button and navigate to **Demos\Courses\CourseService**.



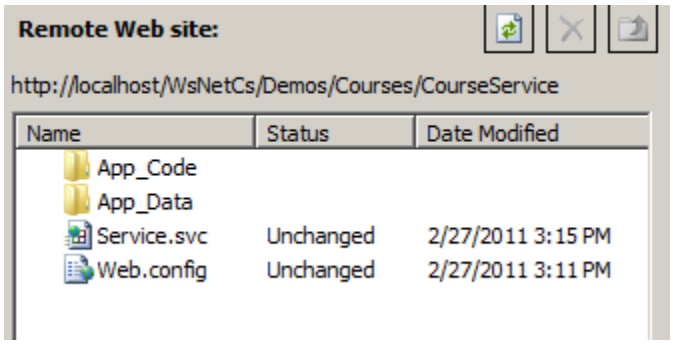
9. Click Open. A message box will inform you that the Web site is not marked as an application in IIS. Click Yes to open it anyway.

Deployment Demo (Cont'd)

10. Select all the files in the Source Web Site and click the blue arrow that points to the right.



11. The files will now be copied to the Remote Web Site.

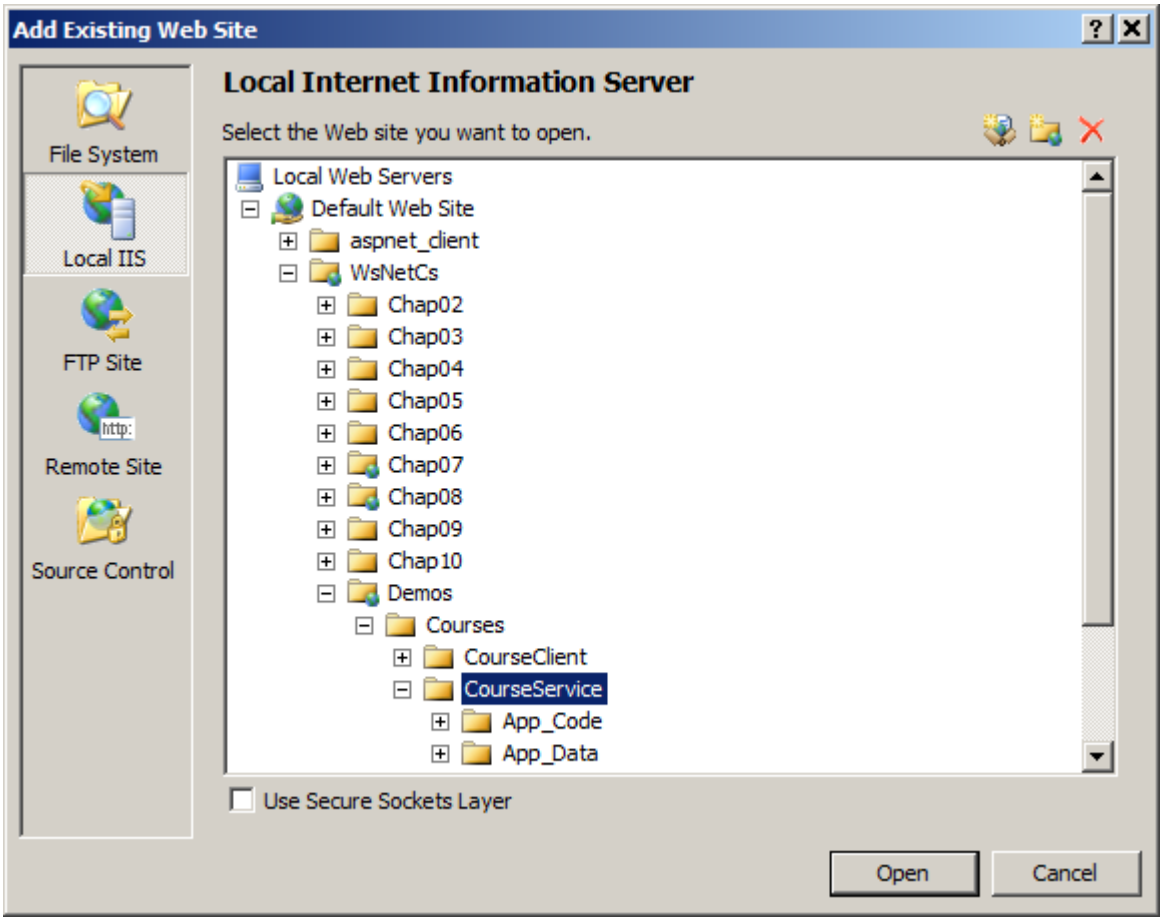


12. Use Visual Studio to open the solution in **Demos\Courses**.

13. In Solution Explorer right-click over the solution and choose Add | Existing Web Site from the context menu.

Deployment Demo (Cont'd)

14. Navigate to **Demos\Courses\CourseService**.



15. Click Open. Again you get a message box reminding you that the Web site CourseService is not marked as an application in IIS. Click Yes to open it anyway.

16. Observe that the Web Site in the solution now has this URL:

```
http://localhost/WsNetCs/Demos/Courses/CourseService/
```

Deployment Demo (Cont'd)

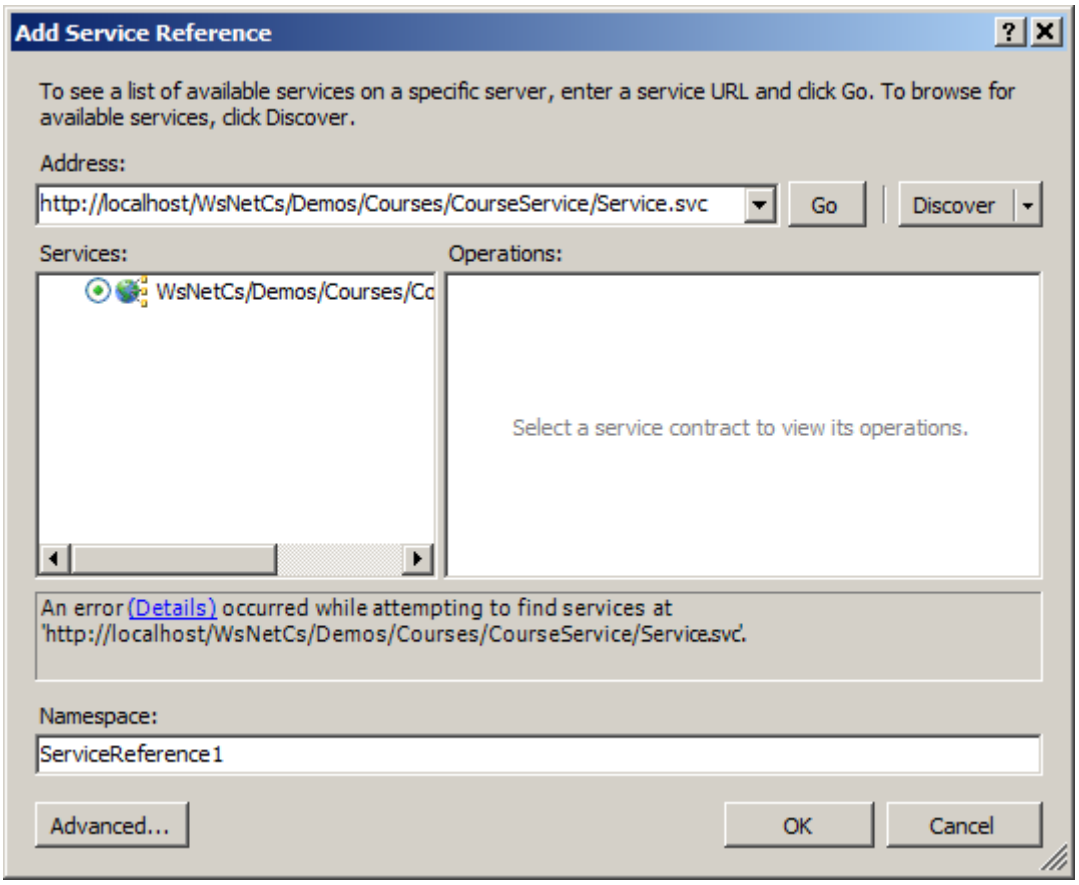
17. To test that we are now working with a copy of the Web service, edit the file **Service.cs** to specify an additional course to be returned by **GetCourses()**.

```
public Course[] GetCourses()
{
   List<Course> courses = new List<Course>();
   courses.Add(new Course(
      "411", "C# Essentials", 100m));
   courses.Add(new Course(
      "412", ".NET Framework Using C#", 150m));
   courses.Add(new Course(
      "444", "Introduction to HTML", 50m));
   return courses.ToArray();
}
```

18. Rebuild the solution and try to run it. You still get the original output! Why?

Deployment Demo (Cont'd)

19. The client is still connected to the Step1 service. In the client project remove the service reference and then try to add a service reference to service in Demos.



20. Click on the "Details" link. You'll find this helpful message: This error can be caused by a virtual directory not being configured as an application in IIS. Use IIS to configure the directory **WsNetCs/Demos/Courses/CourseService** as an application.

Deployment Demo (Cont'd)

21. Back in the Add Service Reference dialog click Go to attempt to find the service again. This time it will work and you can retrieve the metadata.

22. Click OK. Build and run again. This time you'll connect to the modified service.

```
411  C# Essentials                $100.00
412  .NET Framework Using C#      $150.00
444  Introduction to HTML          $50.00
```

Lab 10B

Implementing a Data Contract

In this lab, you implement a data contract for courses. A service contract specifies operations to obtain a list of courses, to add a course, and to delete a course. The course data is stored in an Access database.

Detailed instructions are contained in the Lab 10B write-up at the end of the chapter.

Suggested time: 45 minutes

Summary

- **WCF unifies a number of existing technologies for creating distributed applications.**
- **WCF services can be hosted in IIS, Windows or Console applications, or WAS.**
- **WCF services and clients exchange SOAP messages.**
- **WCF services and clients can be configured in code, or via XML configuration files.**
- **WCF services are defined by addresses, bindings and contracts.**
- **Web services created using ASP.NET are interoperable with those created using WCF as long as both conform to Basic Profile 1.1.**
- **A data contract describes the data to be exchanged between a service and a client.**

Lab 10A

Creating a Simple WFC Service and Client

Introduction

In this lab, you will use Visual Studio 2010 to create a simple WCF service that is hosted in IIS. You will then create a client Console application, use Add Service Reference to create a proxy for the service, and demonstrate that the client can contact the service.

Suggested Time: 45 minutes

Root Directory: **OIC\WsNetCs**

Directories: **Labs\Lab10A** (create solution here)
Chap10\SimpleMath (contains lab solution)

Part 1: Creating the Service Website

1. Use Visual Studio 2010 to create an empty solution **SimpleMath** in the working directory.

2. In Solution Explorer, right-click on the solution and choose Add | New Web Site.

3. For template choose WCF Service. Use HTTP and the following URL:

```
http://localhost/WsNetCs/Labs/Lab10A/SimpleMath/MathService
```

4. In the file **IService.cs**, provide the following simple service contract **ICalc**.

```
[ServiceContract]
public interface ICalc
{
   [OperationContract]
   int Add(int x, int y);
   [OperationContract]
   int Multiply(int x, int y);
}
```

5. In the file **Service.cs**, provide the following implementation:

```
public class Calc : ICalc
{
   public int Add(int x, int y)
   {
      return x + y;
   }

   public int Multiply(int x, int y)
```

```
    {
        return x * y;
    }
}
```

6. Edit the file **Service.svc** to specify **Calc** as the service.

```
<%@ ServiceHost Language="C#" Debug="true" Service="Calc"
CodeBehind="~/App_Code/Service.cs" %>
```

7. In Solution Explorer, make **Service.svc** the Start Page.

8. Build and run the solution. When Internet Explorer comes up, it should display a page with information about the service. In particular, notice the sample client code.

C#

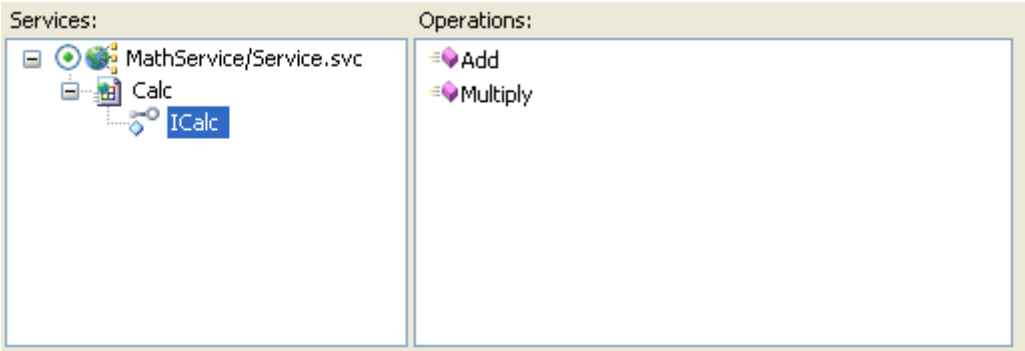
```
class Test
{
    static void Main()
    {
        CalcClient client = new CalcClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

Part 2: Creating a Console Client

1. Add a second project to your solution, a Console application named **MathCon**.

2. In Solution Explorer, right-click over the **MathCon** project and select Add Service Reference from the context menu.

3. In the Add Service Reference dialog, click on the Discover button to find the services in the solution. Expand the services and the interfaces.



4. Note that the suggested namespace of the proxy is ServiceReference1. Click OK.

5. In **Program.cs** import the namespace **MathCon.ServiceReference1**.

6. Provide the following code for **Main()** to instantiate a client proxy and exercise the two methods..

```
static void Main(string[] args)
{
   CalcClient proxy = new CalcClient();
   int sum = proxy.Add(5, 7);
   Console.WriteLine("sum = {0}", sum);

   int prod = proxy.Multiply(5, 7);
   Console.WriteLine("product = {0}", prod);

   proxy.Close();
   Console.WriteLine("Press ENTER to exit");
   Console.ReadLine();
}
```

7. Make **MathCon** the startup project.

8. Build and run.

Lab 10B

Implementing a Data Contract

Introduction

In this lab, you implement a data contract for courses. A service contract specifies operations to obtain a list of courses, to add a course, and to delete a course. The course data is stored in an Access database.

Suggested Time: 45 minutes

Root Directory: **OIC\WsNetCs**

Directories:	**Labs\Lab10B**	(work area)
	Chap10\Courses\Step1	(starter code to deploy)
	Chap10\Courses\Step2	(answer)

Part 1: Deploy the Starter Service Website to Lab Area

1. Copy the contents of **Chap10\Courses\Step1** to the work area.

2. Open the solution **Courses.sln** in the work area.

3. Observe that the Web Site at this new location is still the original Web Site of the Step1 version. Remove this Web Site in Solution Explorer and delete the contents of the folder **Labs\Lab10B\CourseService**.

4. Close the solution in the lab area, saving your modifications. Open the Step1 solution in Visual Studio.

5. In Solution Explorer right-click on the Web Site. Select Copy Web Site from the context menu.

6. Click the Connect button and navigate to **Labs\Lab10B\CourseService**.

7. Click Open. A message box will inform you that the Web site is not marked as an application in IIS. Click Yes to open it anyway. Do this again when it happens later.

8. Select all the files in the Source Web Site and click the blue arrow that points to the right. The files will now be copied to the Remote Web Site.

9. In Visual Studio go back to the solution in the work area. In Solution Explorer right-click over the solution and choose Add | Existing Web Site from the context menu.

10. Navigate to **Labs\Lab10B\CourseService**. Click Open. Observe that the Web Site in the solution now has this URL:

```
http://localhost/WsNetCs/Labs/Lab10B/CourseService/
```

11. As a final step go to the client project and delete the existing service reference. Add a service reference to the service that is now in the solution. You won't be able to find it yet because the Web site is not configured as an application in IIS. Perform this configuration and then finish adding the service reference. You have now completed deployment of the solution with its web site and client program to the lab work area. Congratulations!

Part 2: Obtain Data from an Access Database

1. Examine the file **DB.cs** in the App_Code folder of the Web site project. This contains a class **DBData** that encapsulates access to the Course table of the **Zenith.mdb** database.

2. Open the file **Service.cs** in the Web site project. Replace the stub code that obtains a list of some hard-coded courses by code that obtains the courses from the **GetCourseList()** method of the **DBData** class.

```
public class CourseService : ICourseManager
{
   public Course[] GetCourses()
   {
      DBData DB = new DBData();
      List<Course> courses = DB.GetCourseList();
      return courses.ToArray();
   }
}
```

3. Build and run. You should not have to replace the service reference because you have not modified the contract, only the implementation. Now your client program should show the courses from the database.

4. Add operations to the service contract for adding and deleting a course.

```
[ServiceContract]
public interface ICourseManager
{
   [OperationContract]
   Course[] GetCourses();
   [OperationContract]
   int AddCourse(string courseNumber, string title, decimal price);
   [OperationContract]
   int DeleteCourse(string courseNumber);
}
```

5. Implement these operations by calling the appropriate methods of the **DBData** class.

```
public int AddCourse(string courseNumber, string title, decimal price)
{
   DBData DB = new DBData();
   return DB.AddCourse(courseNumber, title, price);
```

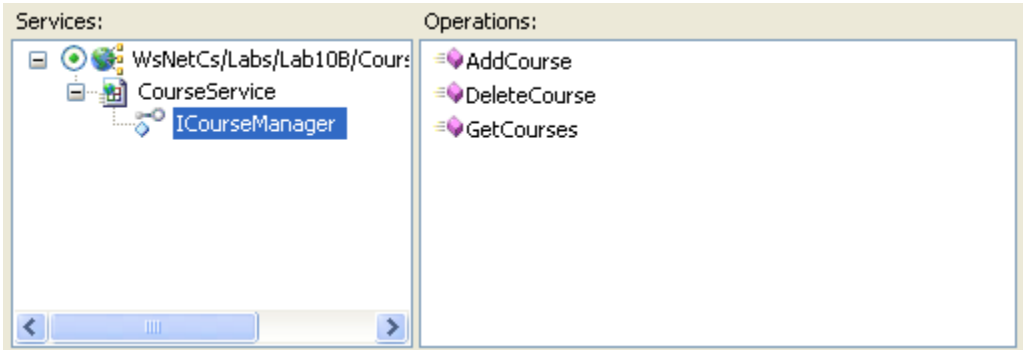
```
}
public int DeleteCourse(string courseNumber)
{
   DBData DB = new DBData();
   return DB.DeleteCourse(courseNumber);
}
```

6. Add code to **Program.cs** in the client project to exercise the new operations. Simply add a course, show the list of courses, delete the new course, and show the list again.

```
static void Main(string[] args)
{
   CourseManagerClient client = new CourseManagerClient();
   Course[] courses = client.GetCourses();
   ShowCourses(courses);
   client.AddCourse("444", "Introduction to WCF", 175m);
   ShowCourses(client.GetCourses());
   client.DeleteCourse("444");
   ShowCourses(client.GetCourses());
   client.Close();
}
```

7. Rebuild the solution. You will get compile errors because the old proxy does not about the new methods that you are attempting to exercise in the test program.

8. Delete the service reference and add a service reference. The proxy will now know about the new operations.



9. Build and run. Your solution should now be fully operational.