msdn® training

Module 9: Using XML Web Services in a C# Application

Contents

Microsoft®

Instructor Notes

Presentation: 60 minutes

The module introduces the **System.Web.Services** namespace and the process of building and consuming XML Web services in a C# application.

Lab: 60 minutes

After completing this module, students will be able to:

- Request data from an XML Web service from within a C# application.
- Build an XML Web service.

Required materials

To teach this module, you need the following materials:

- Microsoft® PowerPoint® file 2609A_09.ppt
- Module 9, "Using XML Web Services in a C# Application"

Preparation tasks

To prepare for this module:

- Read all of the materials for this module.
- Complete the practices and lab.
- Practice the guided practice.

How to Teach This Module

This section contains information that will help you to teach this module.

Important This module includes guided practices. Each practice or the lab may also include optional tasks to accommodate advanced learners.

Explain to the students that most of the complexity of XML Web services, such as handling the transport, creating and parsing XML and SOAP messages, is concealed by the Microsoft .NET Framework. This simplifies working with basic XML Web services. It is possible to modify the processes normally hidden from the developer but this is an advanced XML Web service subject.

Lesson: Consuming an XML Web Service

This section describes the instructional methods for teaching each topic in this lesson.

- The practice for this lesson is a guided practice.
- Referring to the slide for the topic, "What Is an XML Web Service," note that the diagram shows two situations: problematic client and server communication over the Internet by using a COM component, and more efficient client and server communication over the Internet by using SOAP over HTTP that supports any platform.
- The procedures for the topic, "How to Locate the URL of an XML Web Service," do not work in a disconnected network.

Lesson: Building an XML Web Service

This section describes the instructional methods for teaching each topic in this lesson.

- The practice for this lesson is a guided practice.

Practices

The practices for this module are labeled as guided practices. However, you can approach the practices in the following three ways depending on your students' needs:

- If your students are experienced with the topic and most of them successfully complete the practice in the allotted time, you do not need to intervene. This strategy is called a *hands-on practice*.
- If most of your students complete the practice but you feel they could benefit from more instruction, after about 8 minutes into a practice, you can stop them and have them watch and listen as you demonstrate how to solve the tasks on your instructor computer.
- If your students are at beginner level, you can have them perform the steps simultaneously while you demonstrate the solution on your instructor computer. This strategy is called a *guided practice*.

Review

The review questions are based mostly on conceptual understanding and procedures that are covered thoroughly in the module. You can use a discussion format to answer the questions so that everyone gets the benefit of knowing the right answers.

Lab 9.1: Using XML Web Services

Before beginning this lab, students should have completed all of the practices and answered the review questions. Students must be able to perform most of the tasks that they learned in the lessons and the practices.

Overview

Introduction

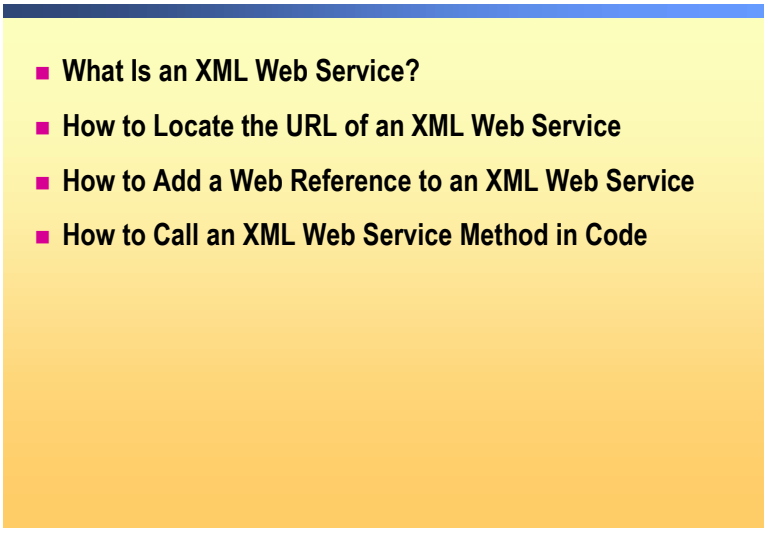
This module introduces the **System.Web.Services** namespace and the process of building and using XML Web services in a C# application.

Objectives

After completing this module, you will be able to:

- Request data from an XML Web service from within a C# application.
- Build an XML Web service.

Lesson: Consuming an XML Web Service



Introduction

This lesson presents XML Web services and explains how to request data from an XML Web service from within a C# application.

Lesson objectives

After completing this lesson, you will be able to:

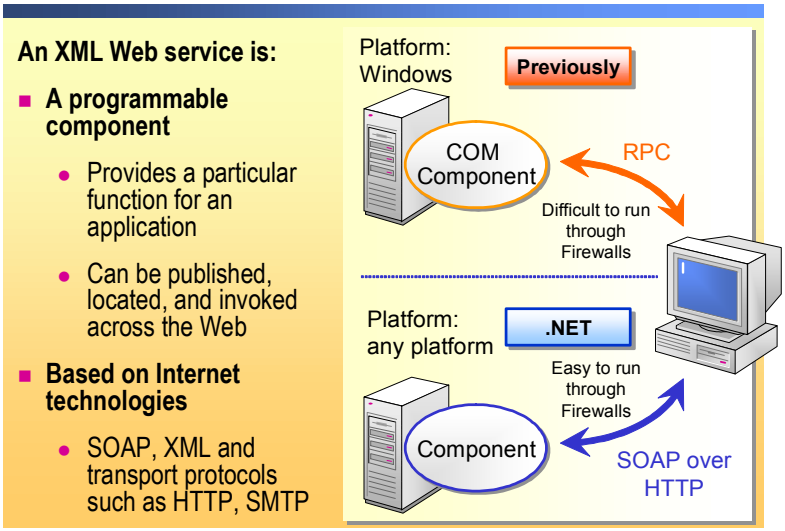
- Add an XML Web service reference to a C# project.
- Invoke methods and properties of an XML Web service.

Lesson agenda

This lesson includes the following topics and activity:

- What Is an XML Web Service?
- How to Locate the URL of an XML Web Service
- How to Add a Web Reference to an XML Web Service
- How to Call an XML Web Service Method in Code
- Practice: Using an XML Web Service from Within C#

What Is an XML Web Service?



Introduction

Applications consist of components. Like components, XML Web services are black boxes. XML Web services encapsulate their functionality and provide an interface for using this functionality. Therefore, you can use XML Web services as building blocks for applications.

Definition

An XML Web service is a programmable component that provides a particular function for an application, such as application logic. Because XML Web services communicate by using Hypertext Transport Protocol (HTTP) and XML, any network node that supports these technologies can both host and access XML Web services. This means that an XML Web service is one component that you can use to build services and applications.

Examples

Examples of XML Web services follow:

- An airline company can provide an XML Web service that accepts a flight number and returns the status of the flight.
- A raw material supply company can provide an XML Web service that allows customers to place orders.
- An online calendar service provider can provide an XML Web service that allows users to update and query their online calendars.

XML Web services protocols

The foundations for XML Web services are XML, SOAP, and commonly supported Internet transport protocols such as HTTP and Simple Mail Transport Protocol (SMTP). This kind of support makes it simple for heterogeneous systems to communicate. For example, a component written in C# and exported as an XML Web service can be used by any application capable of sending and reading XML over the applicable transport.

SOAP is a lightweight XML-based protocol that is used for information exchange. SOAP is transport protocol independent and can be exchanged between a server and a client over transport protocols such as HTTP and SMTP, although it is most commonly implemented over HTTP.

Note The development of these technologies is governed by the World Wide Web Consortium (W3C).

When is an XML Web service an appropriate solution?

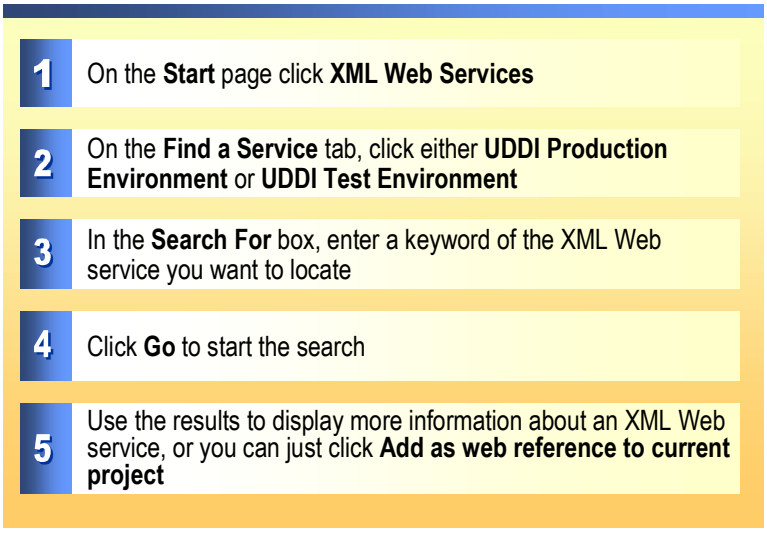
Scenarios that may require XML Web services can be categorized as follows:

- Simple services that provide a fundamental piece of functionality for your clients to use.
- Application integration services that expose the functionality and data of existing disparate applications as an XML Web service.
- Workflow services that enable applications that constitute end-to-end workflow solutions to be created. Such solutions are appropriate for long-running scenarios such as those found in business-to-business transactions.

Examples of scenarios that may require such services are described in the following table.

Service type	Example application	Details
Simple	An e-commerce application that calculates charges for an assortment of shipping options.	Such an application requires current shipping cost tables from each shipping company to use in these calculations.
Simple	An application that sends a simple XML-based message over the Internet, where the message provides the weight and dimensions of the package, ship-from and ship-to locations, and other parameters, such as class of service.	Such an application can send a simple XML-based message over the Internet, using a standard transport protocol such as HTTP, to the shipper's cost calculation XML Web service. The shipper's XML Web service can then calculate the shipping charge, using the latest cost table and return, in a simple XML-based response message, this amount to the calling application for use in calculating the total charge to the customer.
Application integration	By using XML-based messages, applications running on different operating systems and/or hardware platforms can be integrated.	Earlier systems are a good example of systems that may need to be integrated into other system. However, often the platforms on which these earlier systems run make it difficult to integrate into other systems. XML-based messages can provide a common communication mechanism.
Workflow solution	Some organizations depend on integrating business information into the systems of their partners and customers.	Microsoft BizTalk® provides the framework, and technology for business document routing, transformation, and tracking so that you can define mechanisms for identifying and addressing messages, define the message lifetime, package the message with attachments, deliver the message reliably, and secure message contents for authentication, integrity, and privacy.

How to Locate the URL of an XML Web Service



Introduction

If you are developing an application that uses data from a business partner, it is likely that the partner will give you the Uniform Resource Locator (URL) for the XML Web service. However, you may want your application to use other XML Web services that provide special services and data that you do not have the URL for. The advantages of XML Web services are limited if you cannot locate the service that you need. To allow you to easily locate XML Web services, Universal Description Discovery and Integration (UDDI) was created.

UDDI definition

UDDI is a comprehensive industry initiative that allows businesses to:

- Define their business.
- Discover other businesses.
- Share information about how they interact in a global registry.

UDDI is the building block that allows businesses to quickly, easily, and dynamically find and transact business with one another by using their preferred applications.

UDDI also contains standards-based specifications for service description and discovery. The UDDI standard takes advantage of W3C and Internet Engineering Task Force (IETF) standards such as XML, HTTP, and Domain Name System (DNS) protocols.

Locating XML Web service information

You can use UDDI to locate an XML Web service. Microsoft Visual Studio® .NET allows you to search for an XML Web service and then add a suitable XML Web service to your project.

To locate information about XML Web services, perform the following procedure:

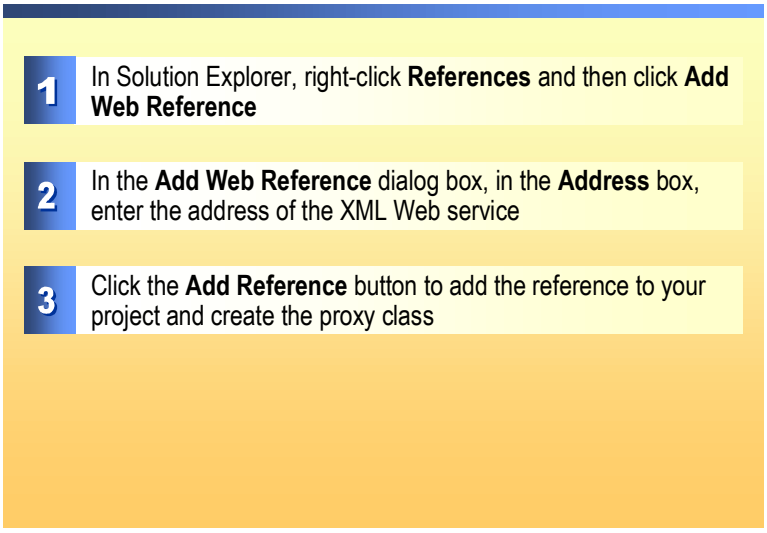
1. On the Visual Studio .NET **Start** page, click **XML Web Services**.

   If the **Start** page is not visible, click **Help**, and then click **Show Start Page**.
2. On the **Find a Service** tab, click either **UDDI Production Environment** or **UDDI Test Environment**.

   Note The **UDDI Production Environment** option is selected by default. If you are developing your application, click **UDDI Test Environment**.
3. In the **Search For** box, enter a keyword of the XML Web service that you want to locate.
4. Click **Go** to start the search.
5. The results of the search are displayed under the search criteria. You can use the results to display more information about an XML Web service, or you can just click **Add as web reference to current project**.

Note This procedure does not work in an environment that is not connected to the Internet.

How to Add a Web Reference to an XML Web Service



Introduction

The functionality necessary to use an XML Web service is integrated into the .NET Framework. In the .NET Framework, this functionality is provided by a *proxy class*.

Proxy class

To access an XML Web service from a client application, you first add a Web reference, which is a reference to an XML Web service. When you create a Web reference, Visual Studio .NET creates an XML Web service proxy class automatically and adds it to your project. This proxy class exposes the methods of the XML Web service, bundles client requests into SOAP messages that are sent on to the server, and retrieves the responses that contain the result.

You can then create an instance of the proxy class in your code so that you can use the methods of the XML Web service as if they were methods of a class that is held locally on your computer.

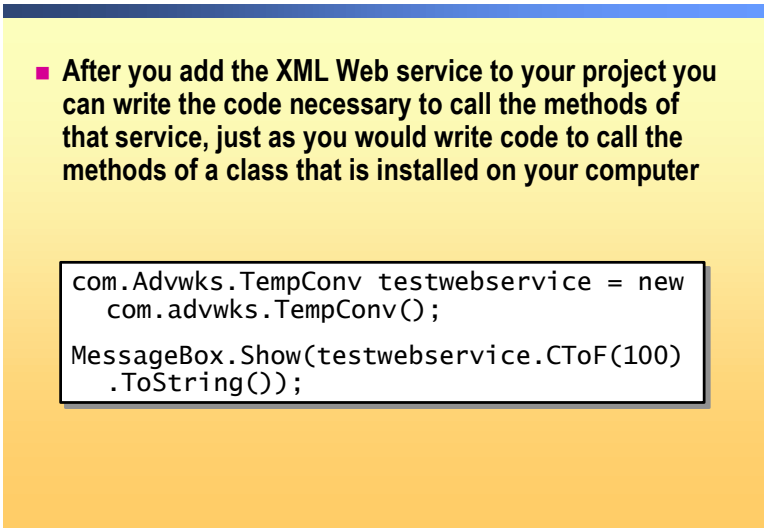
Adding a Web reference

To add a Web reference to an XML Web service, perform the following procedure:

1. In Solution Explorer, right-click **References**, and then click **Add Web Reference**.
2. In the **Add Web Reference** dialog box, in the **Address** box, enter the address of the XML Web service, such as http://advworks.msft/webservices/testservice.asmx.
3. Click the **Add Reference** button to add the reference to your project and create the proxy class.

Note This procedure requires that a connection using the transport protocol indicated is possible.

How to Call an XML Web Service Method in Code



Introduction

After you add the XML Web service to your project, you can write the code necessary to call the methods of that service, just as you would write code to call the methods of a class that is installed on your computer.

Example

For example, if you add the XML Web service that is defined at http://advwks.com/TempConv, the full name of the class created is:

```
<application name space>.com.advwks.TempConv
```

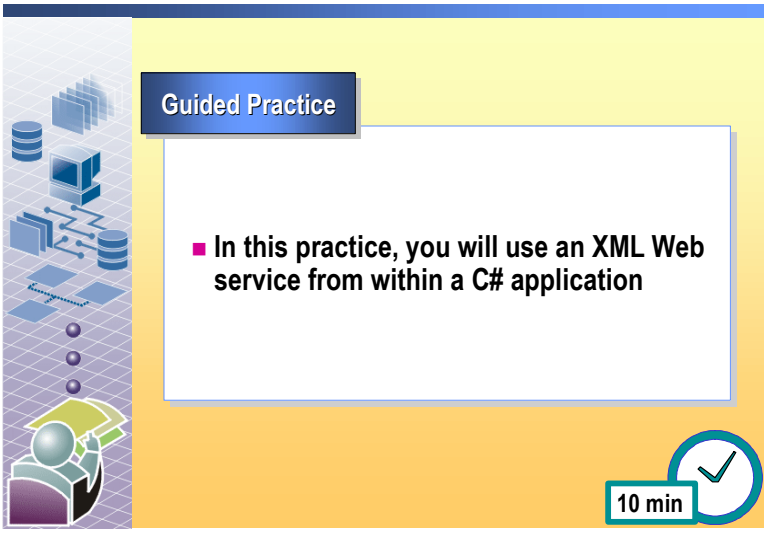
An example of calling the **CToF** method of this XML Web service is:

```
com.Advwks.TempConv testwebservice = new
   com.advwks.TempConv();
MessageBox.Show(testwebservice.CToF(100).ToString());
```

The first line of the preceding code creates an instance of the **TempConv** class. The last line uses the static method of the **MessageBox** class **Show** to display a message box that shows the result of **TempConv** class method **CToF**.

Note After the reference to the XML Web service is made, the class is treated in code as a local class, and the details of SOAP and HTTP are abstracted.

Practice: Using an XML Web Service from Within C#



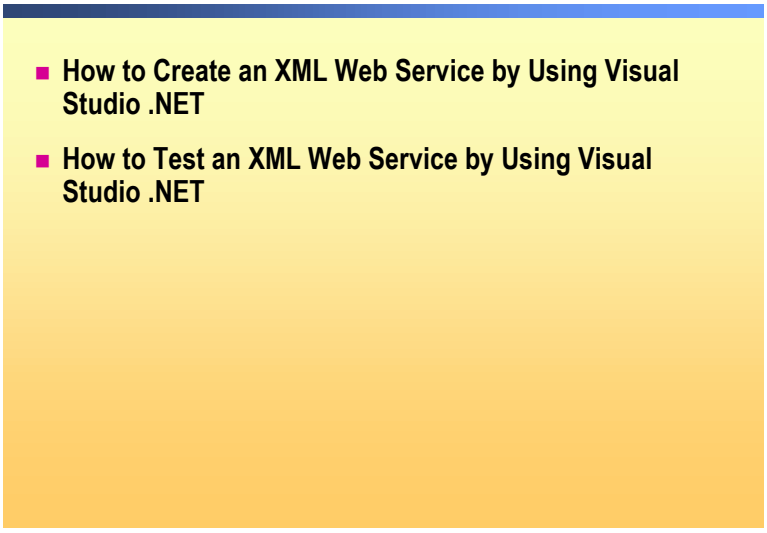
In this practice, you will use an XML Web service from within a C# application. The provided XML Web service will run on your computer. The XML Web service provides four basic math functions: add, subtract, divide, and multiply.

The solution for this practice is in *install_folder*\Practices\Mod09\Consume_Solution. Start a new instance of Visual Studio .NET before opening the solution.

Tasks	Detailed steps
1. Install the Math Web Service by using the Setup.exe in *install_folder*\Practices \Mod09\MathWebService.	**a.** Click **Start**, and then click **Run**. **b.** In the **Run** dialog box, in the **Open** box, type *install_folder***Practices\Mod09\MathWebService \Setup.exe** and then click **OK**. **c.** In the **Math Web Service** setup wizard, on the **Welcome to the Math Web Services Setup Wizard** page, click **Next**. **d.** On the **Select Installation Address** page, click **Next**. **e.** On the **Confirm Installation** page, click **Next**. **f.** On the **Installation Complete** page, click **Close**.
2. Start Visual Studio .NET, and create a new project. Name: **Consume** Project Type: **Visual C# projects** Template: **Windows Application** Location: *install_folder***Practices\Mod09**	**a.** Start Visual Studio .NET. **b.** On the **File** menu, point to **New**, and then click **Project**. **c.** In the **New Project** dialog box, under **Project Types**, click **Visual C# Projects**. **d.** Under **Templates**, click **Windows Application**. **e.** In the **Name** box, type **Consume** **f.** In the **Location** box, type *install_folder***Practices\Mod09** and then click **OK**.

Tasks	Detailed steps
3. Add a Web reference to the project URL: http://localhost/MathWebService/Math.asmx	**a.** In Solution Explorer, right-click **References**, and then click **Add Web Reference**. **b.** In the **Add Web Reference** dialog box, in the **Address** box, type **http://localhost/MathWebService/Math.asmx** and then click the **Go** (→) button. **c.** When the download is complete, click **Add Reference**.
4. Add two text boxes to the form.	▪ Add two text boxes to the form.
5. Add a button with a caption **Add**.	▪ Add a button with a caption **Add**.
6. Add a button with a caption **Subtract**.	▪ Add a button with a caption **Subtract**.
7. Add a button with a caption **Multiply**.	▪ Add a button with a caption **Multiply**.
8. Add a button with a caption **Divide**.	▪ Add a button with a caption **Divide**.
9. Write code into each button click event that uses the appropriate Web method of the Math Web service. Use the data held in textbox1 and textbox2 as parameters for the method. Use the **MessageBox** class to display the result.	▪ Refer to the code example earlier in this module to help you write the code that satisfies the task on the left.
10. Run, debug if necessary, and then test your application.	▪ Run, debug if necessary, and then test your application.

Lesson: Building an XML Web Service



Introduction

This lesson explains how to build and test an XML Web service.

Lesson objectives

After completing this lesson, you will be able to:

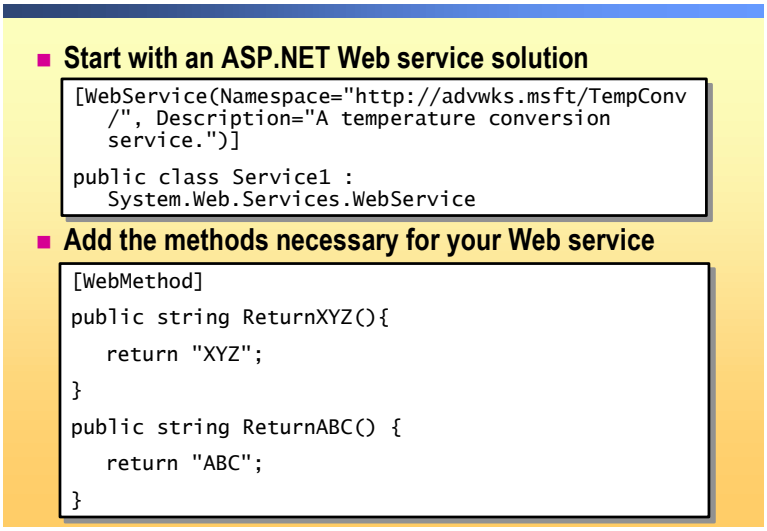
- Create XML Web services by using Visual Studio .NET.
- Test an XML Web service.

Lesson agenda

This lesson includes the following topics and activity:

- How to Create an XML Web Service by Using Visual Studio .NET
- How to Test an XML Web Service by Using Visual Studio .NET
- Practice: Creating an XML Web Service

How to Create an XML Web Service by Using Visual Studio .NET



Introduction

The .NET Framework and Visual Studio NET provide all the functionality that is necessary to implement an XML Web service. Visual Studio .NET provides a project template for developing an XML Web service.

Creating an XML Web service

Visual Studio .NET provides an ASP.NET Web Service project template to help you create XML Web services in Microsoft Visual C#®.

To create an XML Web service by using Visual Studio .NET, perform the following procedure:

1. On the **File** menu, point to **New**, and then click **Project**.
2. In the **New Project** dialog box, select the Visual C# Projects folder.
3. Click the **ASP.NET Web Service** icon.
4. Enter the location of the XML Web service, for example http://localhost/webservice.
5. Click **OK** to create the project.
6. In Solution Explorer, right-click **Service1.asmx**, and then click **Rename**. Enter a suitable name for the XML Web service, for example **TempConv**.
7. In Solution Explorer, right-click the service, for example **TempConv**, and then click **View Code**.

8. It is important to provide a unique namespace for your XML Web service. To do this, add a **WebService** attribute on the line above the class definition as shown in the following example:

```
[WebService(Namespace="http://advwks.msft/TempConv/",
Description="A temperature conversion service.")]
public class Service1 : System.Web.Services.WebService
```

Important If you do not provide a namespace for your XML Web service, when you test your XML Web service, you will see a warning message stating that the service is using the namespace http://tempuri.org as its namespace.

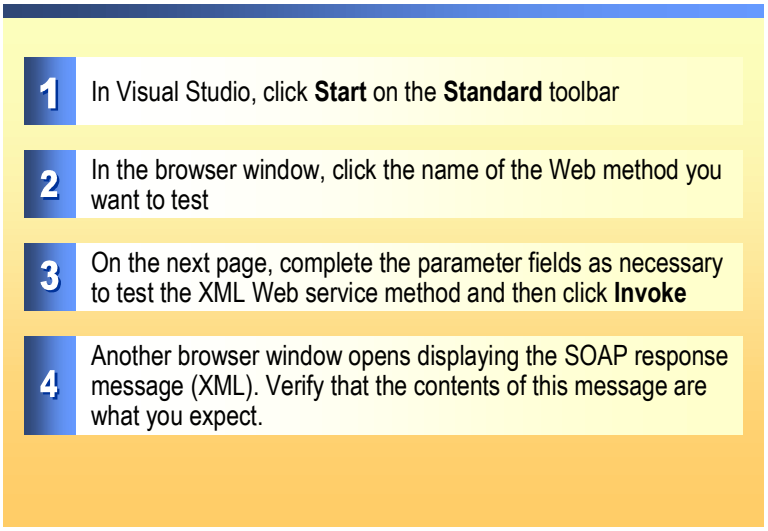
9. After adding the XML Web service attribute, you can scroll through the code to the EXAMPLE WEB METHOD section. Here, you add the methods that are necessary for your XML Web service. Each Web method must be defined as public and marked with a special attribute, **WebMethod**, or the method will not be available in the XML Web service. For example, only the **ReturnXYZ** method shown in the following example will be available as an XML Web service method, because the second method is not marked with the **WebMethod** attribute.

```
[WebMethod]
public string ReturnXYZ(){
   return "XYZ";
}

public string ReturnABC() {
   return "ABC";
}
```

10. Compile the project.

How to Test an XML Web Service by Using Visual Studio .NET



Introduction

Because messages are protocol independent, a message can be used with HTTP-GET/POST, SOAP, or any other protocol that an XML Web service provider supports. If you use SOAP, the message element corresponds to the payload of the SOAP request or response. An XML Web service created by using the .NET Framework and Visual Studio .NET can receive requests in three different formats:

- HTTP GET
- HTTP POST
- SOAP (over HTTP)

The first way to test a new XML Web service is by using the HTTP GET format. Visual Studio .NET provides access to this method.

Testing an XML Web service

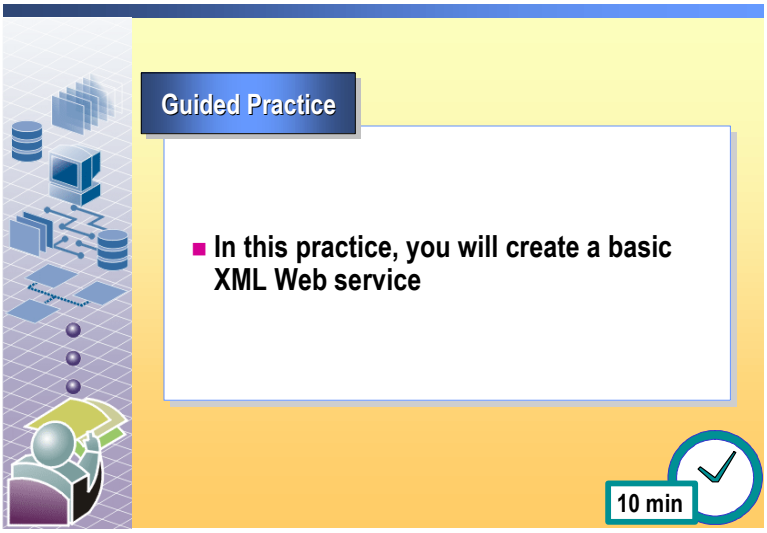
To test your XML Web service, perform the following procedure:

1. In Visual Studio .NET, click **Start** on the **Standard** toolbar.

   The project compiles and then launches a browser window to the XML Web service address. This browser window allows you to test the XML Web service by completing a Web form.
2. In the browser window, click the name of the Web method that you want to test.
3. On the next page, complete the parameter fields as necessary to test the XML Web service method, and then click **Invoke**.
4. Another browser window opens displaying the SOAP response message in XML. Verify that the contents of this message are what you expect.

After the responses to the HTTP GET tests are correct, you should plan to test the Web service from a test application using SOAP over HTTP.

Practice: Creating an XML Web Service



In this practice, you will create a basic XML Web service. This XML Web service will accept an array of decimal values and will calculate the average value from the array of decimal values.

The solution for this practice is in *install_folder*\Practices\Mod09\Create_Solution. To open the solution, follow the instructions in the Readme.txt file located in the folder.

Tasks	Detailed steps
1. Start Visual Studio .NET and create a new project. Name: **Create** Project Type: **Visual C# projects** Template: **ASP.NET Web Service** Location: **http://localhost/Create**	**a.** Start Visual Studio .NET. **b.** On the **File** menu, point to **New**, and then click **Project**. **c.** In the **New Project** dialog box, under **Project Types**, click **Visual C# Projects**. **d.** Under **Templates**, click **ASP.NET Web Service**. **e.** In the **Location** box, type **http://localhost/Create** and then click **OK**.
2. Rename Service1.asmx to **Stats.asmx**	**a.** In Solution Explorer, right-click **Service1.asmx**, and then click **Rename**. **b.** Type **Stats.asmx**
3. Display the code view of **Stats.asmx**.	▪ In Solution Explorer, right-click **Stats.asmx**, and then click **View Code**.
4. Find and replace all occurrences of **Service1** with **Stats**.	**a.** On the **Edit** menu, point to **Find and Replace**, and then click **Replace**. **b.** In the **Replace** dialog box, in the **Find What** box, type **Service1** **c.** In the **Replace** with box, type **Stats** and then click **Replace All**. Three replacements are made.

Tasks	Detailed steps
5. Add a public method to the **Stats** class named **Analyze**. This method should accept an array of decimal types and return the average as type **decimal**.	**a.** Add a public method to the **Stats** class named **Analyze.** This method should accept an array of decimal values and return a decimal (the average of the values passed into the procedure). **b.** Write code to loop through the array, summing the values contained in the array. Finally, divide the total by the number of values contained in the array and return this value as the result of the Web method.

6. Add the **WebService** attribute to the **Stats** class.
Set the Namespace for the Web service to **http://advworks.msft /webservices/**
	▪ Above the line
	`public class stats : System.Web.Services.WebService`
	Add the following line:
	`[WebService(Namespace="http://advworks.msft /webservices/")]`

7. Add the **WebMethod** attribute to the **Analyze** method.	▪ Above the line:
	`public decimal Analyze(decimal[] values)`
	Add the following line:
	`[WebMethod]`

8. Compile the Web service.	▪ In the Solution Explorer window, right-click **Create**, and then click **Build**.
Note: Because this Web service takes an array as a parameter it is not possible to test this Web service using the browser interface. You must create a test application to call the Web service with an array of values.	
9. Start another instance of Visual Studio .NET. Create a C# Windows application to test the Web service created above.	**a.** Start a new instance of Visual Studio .NET. **b.** Create a new C# Windows Application project. **c.** Add a Web reference to **http://localhost/Create/stats.asmx** **d.** Add a button to the form. **e.** In the button's click event, create an array of decimal values **100**, **20.2**, **34.5**, **42.3**, **103**. **f.** Create an instance of the Web service proxy class (**localhost.Stats**). **g.** Call the **Analyze** method of the proxy class passing the array of decimal values. **h.** Use a **MessageBox** to display the average returned. The average of the decimal values is **60**.

Review

1. How do you declare a method as a Web method?

   A Web method is declared by adding the [WebMethod] attribute above the method definition.

2. How do you define a class as an XML Web service?

   A Class is defined as an XML Web service by adding the [WebService…] attribute to the class definition.

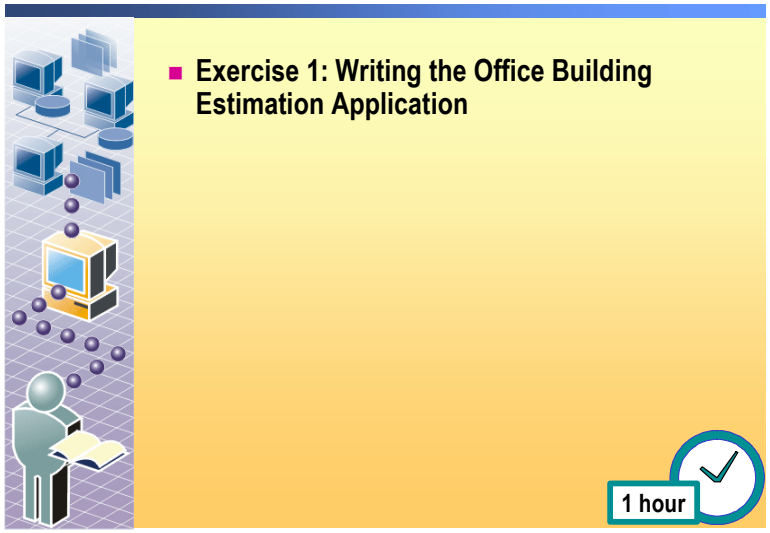
3. When would you use an XML Web service?

   An XML Web service is appropriate for applications that may need to send XML-based messages over the Internet that provide specific functionality for a customer, applications that require integration where the XML Web service exposes the functionality and data of each application as an XML Web service, or workflow applications where the XML Web service allows end-to-end workflow solutions to be created.

4. Why would you use an XML Web service in an internal network?

   An XML Web service can be used in an internal network to support earlier applications by exposing the functionality of the applications to programmers in an organization.

Lab 9.1: Using XML Web Services



Objectives

After completing this lab, you will be able to:

- Use an XML Web Service.

Prerequisites

Before working on this lab, you must have:

- Completed Module 9.

Scenario

This lab demonstrates how an application can use multiple XML Web Services to provide a solution. You will write a basic Windows application that prompts the user to enter the quantities for different components that make up an office building and then responds by providing the cost estimates from different suppliers for the raw materials. Each materials supplier has implemented an XML Web service to allow you to convert quantities into a cost.

The solution for this lab is provided in *install_folder*\Labfiles\Lab09_1\MaterialApplication_Solution. To open the solution, follow the instructions in the Readme.txt file located in the folder.

Complete this lab:
60 minutes

Exercise 0
Lab Setup

The Lab Setup section lists the tasks that you must perform before you begin the lab.

Tasks	Detailed steps
1. Log on to Windows as **Student** with a password of **P@ssw0rd**.	▪ Log on to Windows with the following account: • User name: **Student** • Password: **P@ssw0rd**
2. Install the Material Web Services using the Setup.exe found in *install_folder***Lab09_1\WebService.**	**a.** Click **Start**, and then click **Run**. **b.** In the **Run** dialog box, in the **Open** box, type *install_folder***Lab09_1\WebService\Setup.exe** and then click **OK**. **c.** In the **Material Web Services** setup wizard, on the **Welcome to the Material Web Services Setup Wizard** page, click **Next**. **d.** On the **Select Installation Address** page, click **Next**. **e.** On the **Confirm Installation** page click **Next**. **f.** On the **Installation Complete** page, click **Close**.

Exercise 1
Writing the Office Building Estimation Application

In this exercise, you will create a Windows Application by using C#.

Your instructor will have on display a list of the names of the servers in the room. Each server in the room has a Web service with methods to calculate the cost and shipping cost for each of six raw materials.

Select a server (not your own). The URL you need for the Web service on that server is provided below.

http://*servername*/**materialwebservices/estimate.asmx**

Tasks	Detailed steps
1. Start Visual Studio .NET, and create a new Visual C# Project based on the Windows Application template.	**a.** Start a new instance of Visual Studio .NET. **b.** On the **Start Page**, click **New Project**. **c.** In the **New Project** dialog box, under **Project Types**, click **Visual C# Projects**, under **Templates**, click **Windows Application**. **d.** In the **Name** box, type **MaterialApplication** **e.** In the **Location** box, type *install_folder***Lab09_1\MaterialApplication** and then click **OK**.
2. Add a Web Reference to the Web service specified by the URL above this table.	**a.** In the Solution Explorer window, right-click **References**, and then click **Add Web Reference**. **b.** In the **Add Web Reference** dialog box, in the **Address** box, type **http**://*servername*/**materialwebservices/estimate.asmx** and then press Enter. **c.** Click **Add Reference**.

Note: You must now select 3 of the following raw materials. You will implement controls on the form to accept the quantities for these raw materials.

- Concrete (cubic meters)
- Steel Bars (meters)
- Windows
- Doors
- Carpet (square meters)
- Paint (cubic meters)

Tasks	Detailed steps
3. Add to the form a label and text box pair for each of the three selected raw materials.	**a.** Add a label to the form and change **Text** property of the label to match the name of the first raw material. **b.** Add a text box to the form, next to the first label. You may wish to change the name of the text box to the name of your first raw material. **c.** Add a label to the form and change **Text** property of the label to match the name of the second raw material. **d.** Add a text box to the form, next to the second label. You may wish to change the name of the textbox to the name of your second raw material. **e.** Add a label to the form and change **Text** property of the label to match the name of the third raw material. **f.** Add a text box to the form, next to the third label. You may wish to change the name of the textbox to the name of your third raw material.
Note: If you select Windows as a raw material, you must add a check box control to your form, under the label and window quantity text box, that allows you to specify double-glazed windows. In Step 5, add checkBox1.Checked as the second parameter of the call to the Web service.	
4. Add a button to the form and change the caption of the button to **Calculate**.	▪ Add a button to the form and change the **Text** property of the button to **Calculate**.
5. Add code to the click event of the **Calculate** button to call the appropriate Web service method for each raw material passing the quantities from the related text box on the form.	▪ Add code to the click event of the **Calculate** button to call the appropriate Web service method for the three raw materials your application is using. The code in the note below can be modified and used for each raw material. You need to add additional code to calculate the total cost and total delivery charges and display this information in a message box.

Note: Example code for calling the Concrete raw material Web service method.
`Servername.Estimate EstimateWebService = new`
`Servername.Estimate();`
`Servername.Result result = EstimateWebService.Concrete(`
`System.Convert.ToDecimal(Concrete.Text));`
`\\result.Cost now contains the cost of the concrete`
`\\result.Delivery now contains the delivery charge`	

6. Run and test the application.	▪ Run and test the application.