Lecture 11

Web services

Web service allows you to store common processing routings on a web server, where they are available to other programmers who need to use the routines in their applications.

Introduction to Web Service

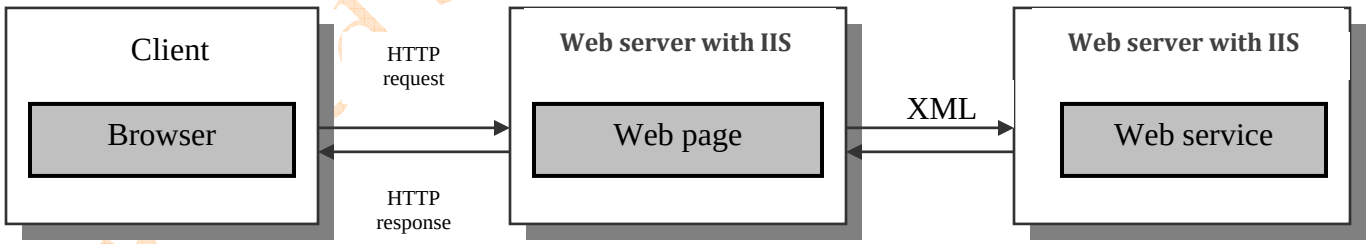
A Web Service is a class that resides on a web server and can be accessed via the Internet or an Intranet. As a result, ASP.Net application or other types of web applications can access the web service from any computer that is connected to the Internet or a local Intranet.

What web services are used for

Web services are becoming an integral part of web development. One use of web services is to develop *distributed web applications* in which portions of an application can reside on different servers. For example, you might implement the business logic and database processing code for an applications as a web service that run on one server. Then, the user interface portion of the application can run on another server and call on the web service whenever business logic or database features are needed. It is even possible to provide two interfaces to the same service: a web interface provided by an ASP.NET application that can be run from any computer that has a browser and Internet access, and a Windows interface provided by a Windows application that is installed on the user's computer.

Another use for web services is to allow applications developed by different companies or organizations to interact with one another. For example, the United States Postal Service has web services that let you calculate shipping rates, correct addresses, and track packages. Similarly, the popular search site Google offers web services that let you incorporate Google searches into your applications. As more companies create web services and publish them on the Internet, applications that use these services will become more popular.

How web service work

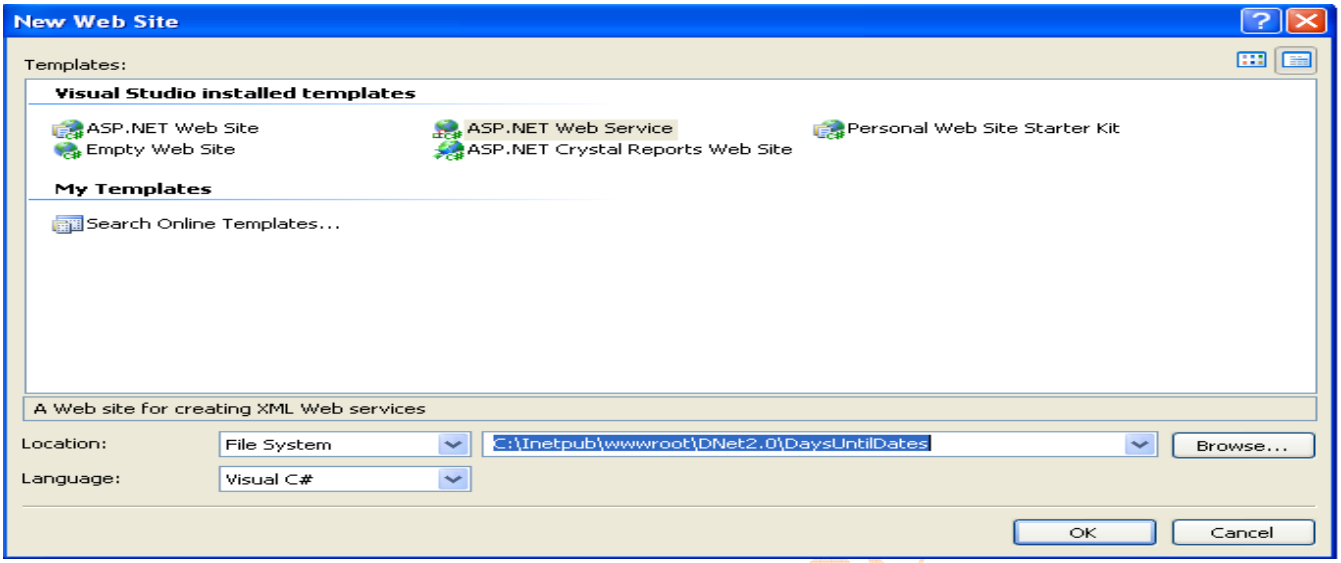


In this example, the web service and the application's web pages resides on two different web servers. Although that doesn't have to be the case, this arrangement illustrates how web services can be used to create distributed applications. The server that hosts the web service must have both IIS and the .NET Framework installed.

Notice that the web page and the web service communicate using XML. Actually, web services use a rather complicated collection of protocols that are built on XML. These protocols include ***Web Services Description Language (WSDL)***, and ***Simple Object Access Protocol (SOAP)***. As a result, you don't usually have to deal directly with XML, WSDL, or SOAP.

How to create a web service

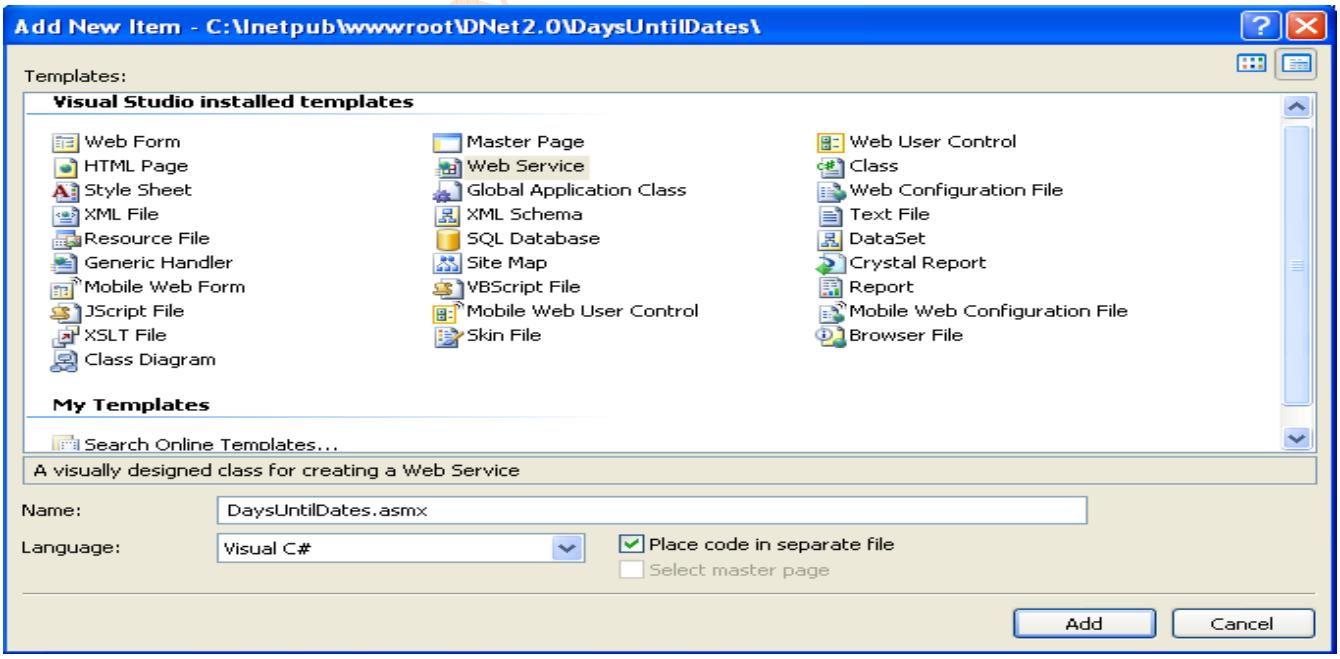
To create a web service, you start by creating a web service web site:



Then, you add each web service you want to create to this web site. Finally, you use the Code Editor to develop the public and private code for each web service.

In the dialog box, presented above, you identify the location of the web server and the name of the directory where the web service web site will reside just as you do for a regular web site. Then, all the files for the web site are stored in that directory on the web server.

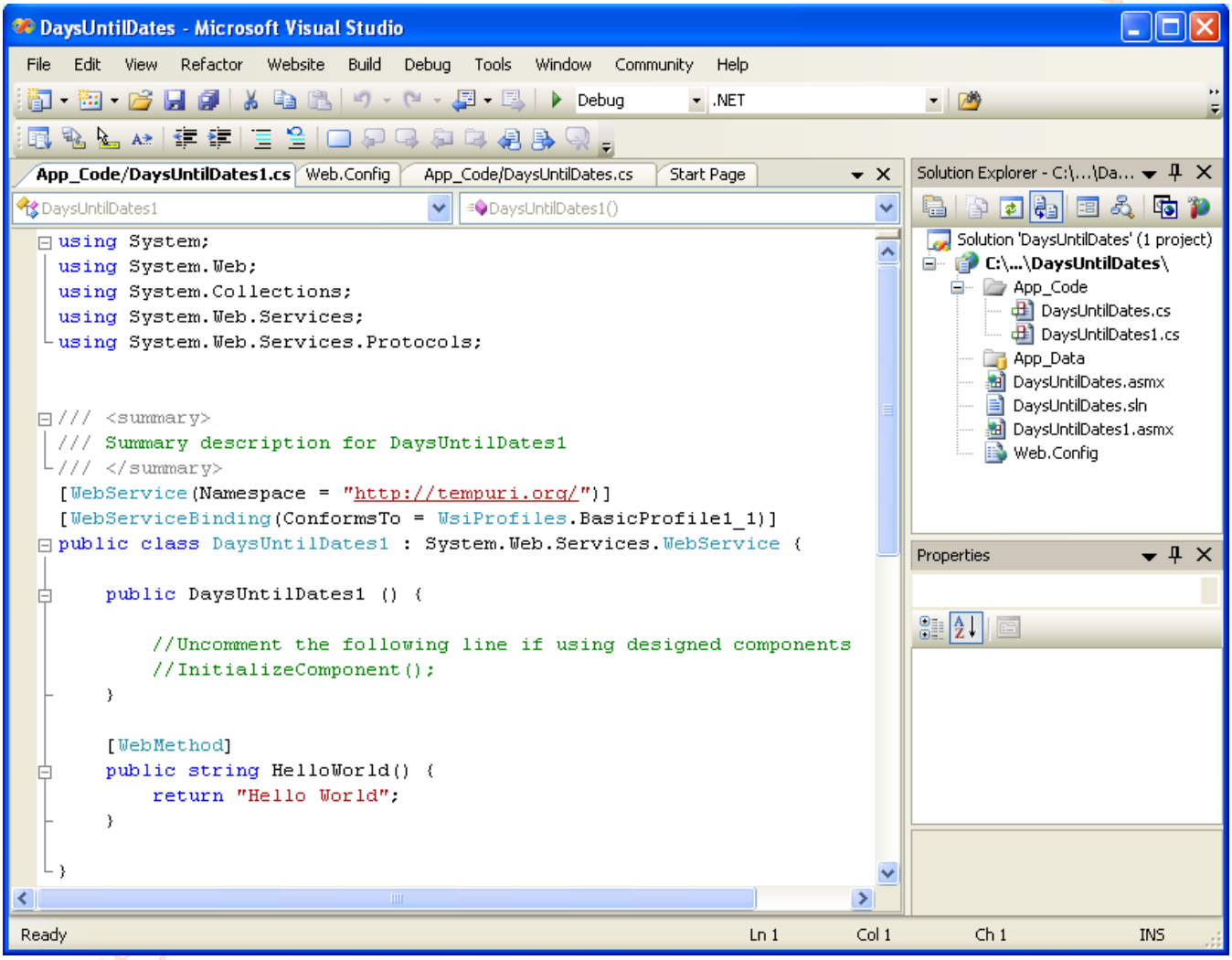
When you first start a web service web site, it contains a single web service named *Service*. In most cases, you'll want to delete this web service and add your own. Below you can see a web site with a web service named *DaysUntilDates*. To add a web service like this, you use the *Add New Item* dialog box, shown here:



Developing a programming code for the web service

When you create a web service, two files are added to the web site. The first one has an extension of *asmx* and is the file you use to call the methods that you add to the web service. Like the *aspx* extension that's used for web pages, the asmx extension tells IIS that file should be processed by ASP.NET.

The second file for a web service has a *cs* extension and is stored in the *App_Code folder*. This file contains the methods and other code that define the web service. The starting code in this file is displayed by default when you create a web service:



First of all, like everything else that's built on the .NET Framework, a web service is implemented as a class. In this case, the class inherits the *WebService* class of the *System.Web.Service* namespace, which provides the web service with access to the ASP.NET objects commonly used by web services.
Secondly, notice the *WebService* attribute that precedes the class definition. An attribute is similar to a keyword (like *Public* or *Private*) that provides information as to how the code should be used. In this case, the *WebService* attribute identifies the namespace that will contain the web service. The default is *tempuri.org*, which is a temporary namespace that you can use during development of a web service. If you publish a web service so it is available to other users, however, you will want to change this name to

something unique. If a web service is made public, it must have a unique namespace to distinguish it from other web services with same name.

The *WebService* attribute can also contain a description of the web service. If you publish a web service, you will want to include this information to provide potential users of the service with information about what it does.

You use this attribute to identify a public method as a ***web method***, which is a method that's accessible from outside the service. Aside from the *WebMethod*, you code a web method just as you would any other method. In additional, you can include private variables and procedures that are used by the web method.

One key difference between a web service and regular *C#* class is that a web service ***can't have properties***. You can define public variables, but those variables aren't accessible to clients that use the web service. As a result, ***web methods are the only interface a client has to a web service***.

You can still implement the equivalent of properties using web methods, though. For example, to retrieve the value of a property, use a web method that's coded as a function that returns the value of the variable for the property.

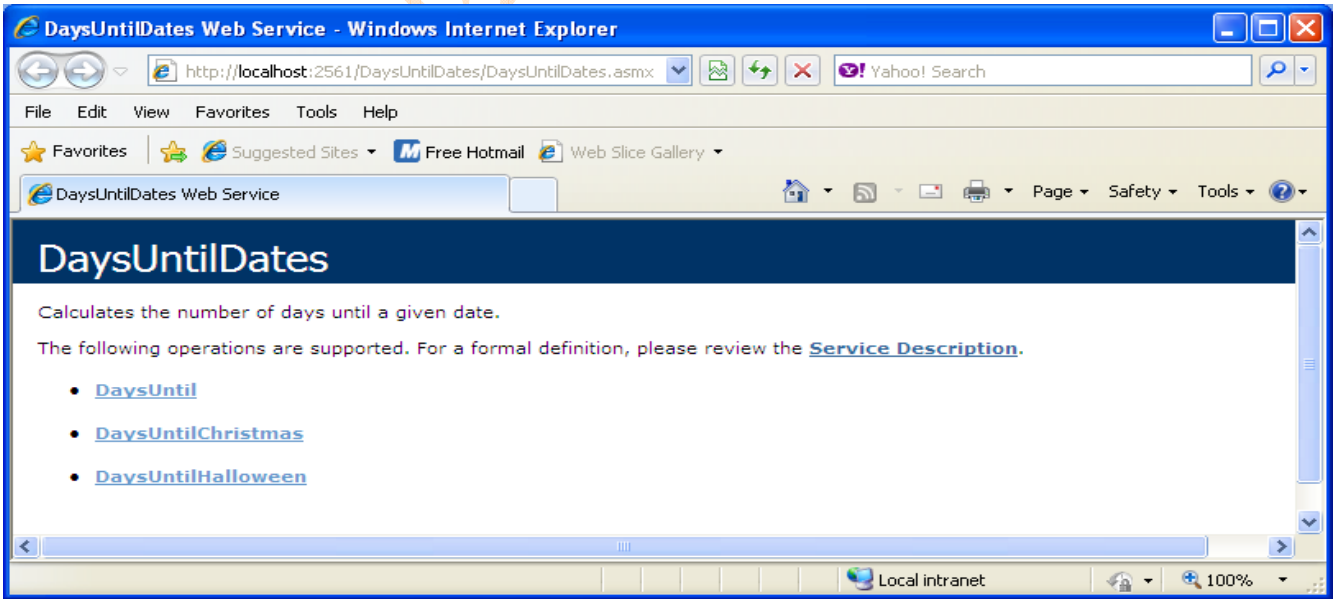
To illustrate how you can use a web service you can see below a code for a web service named *DaysUntilDates*:

```
using System;
using System.Web;
using System.Collections;
using System.Web.Services;
using System.Web.Services.Protocols;

/// <summary>
/// Summary description for DaysUntilDates
/// </summary>
[WebService(Namespace = "http://www.Utech.edu.jm/",
  Description="Calculates the number of days until a given date.")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class DaysUntilDates : System.Web.Services.WebService {

  [WebMethod]
  public int DaysUntil(DateTime date)
  {
    return DaysUntilDate(date.Month, date.Day);
  }

  [WebMethod]
  public int DaysUntilHalloween()
  {
    return DaysUntilDate(10, 31);
  }
```

```
    [WebMethod]
    public int DaysUntilChristmas()
    {
        return DaysUntilDate(12, 25);
    }

    private int DaysUntilDate(int month, int day)
    {
        DateTime targetDate;

        targetDate = DateTime.Parse(month.ToString() + "/"
                + day.ToString() + "/"
                + DateTime.Today.Year);

        if (DateTime.Today > targetDate)
        {
            targetDate = targetDate.AddYears(1);
        }

        TimeSpan timeUntil = targetDate - DateTime.Today;

        return timeUntil.Days;
    }
}
```

How to test a web service

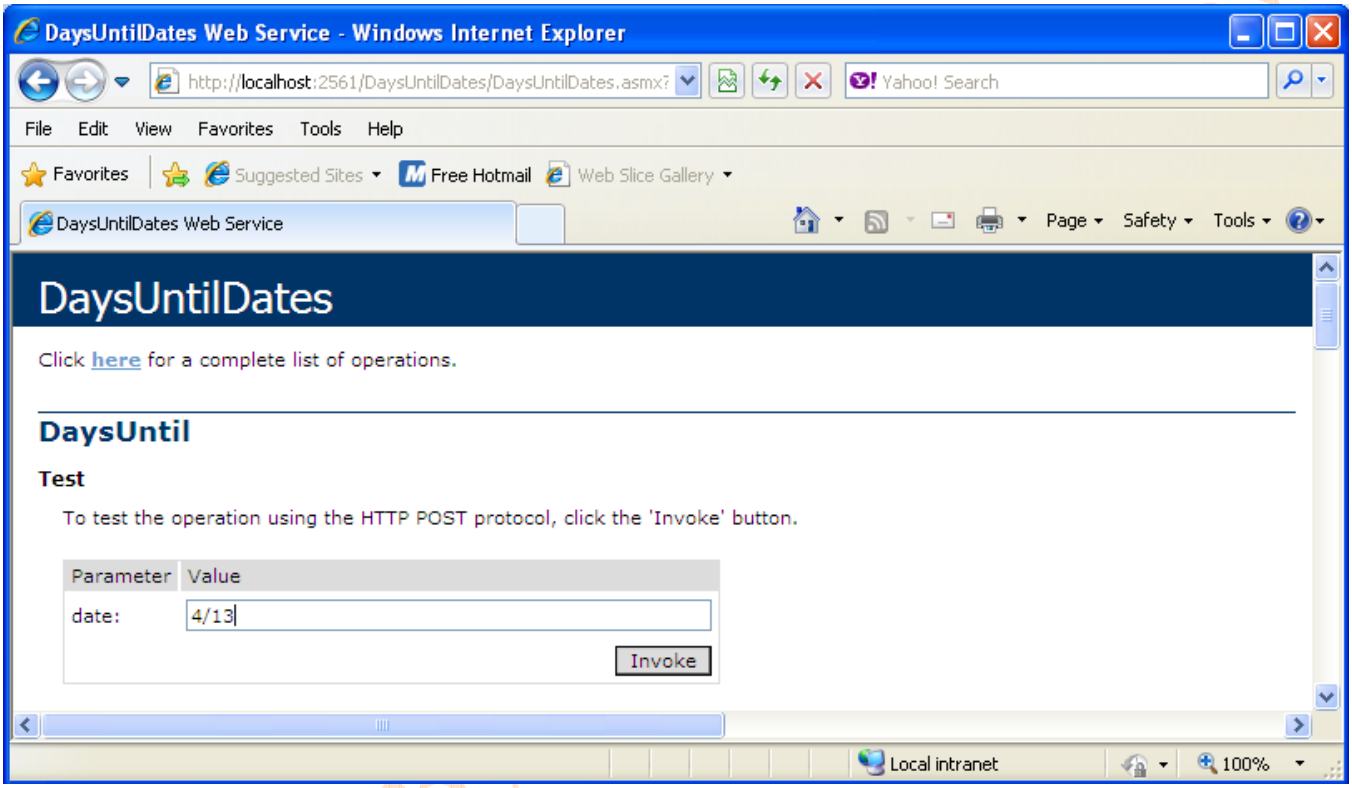
To do that, you simply build and run the web service in your default browser or in a Browser window just as you would a web application. When you do that, a ***Service help page*** like shown below is displayed:



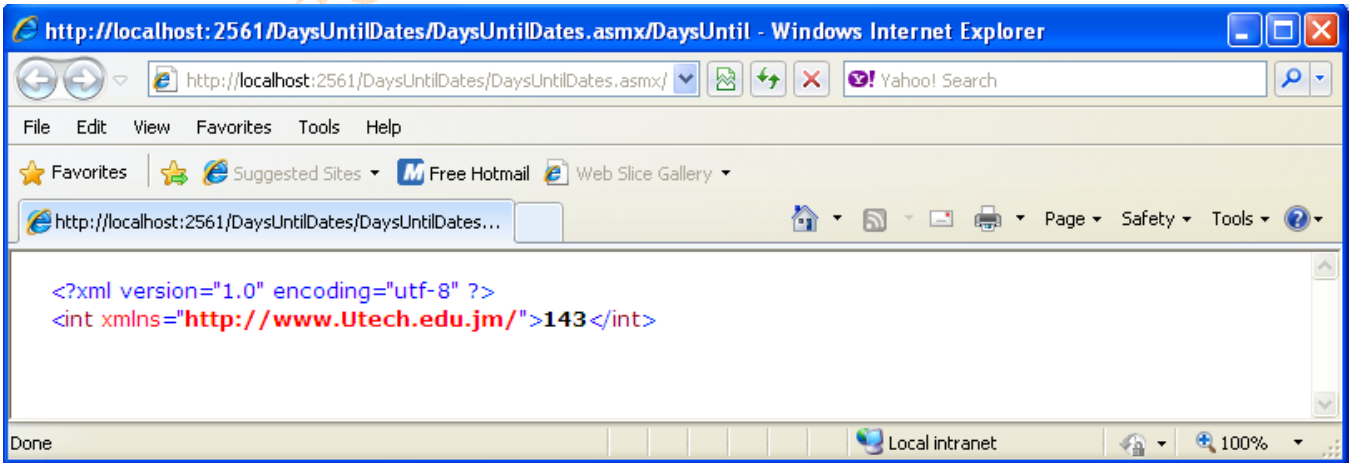
The *Service* help page identifies the web service and displays the description you specified in the *WebService* attribute. It also includes a link to the service description for the web service.

Finally, the *Service* page lists the web methods that are available from the web service. If you click on the link for one of these methods, a *Service method help page* for that method is displayed. The screen below, for example, is the help page for the *DaysUntil* method.

If a method required arguments, the *Service Method* help page lists them by name and let you enter values for them. Then, when you click the *Invoke* button, the method is executed and the XML that's generated as a result is displayed:



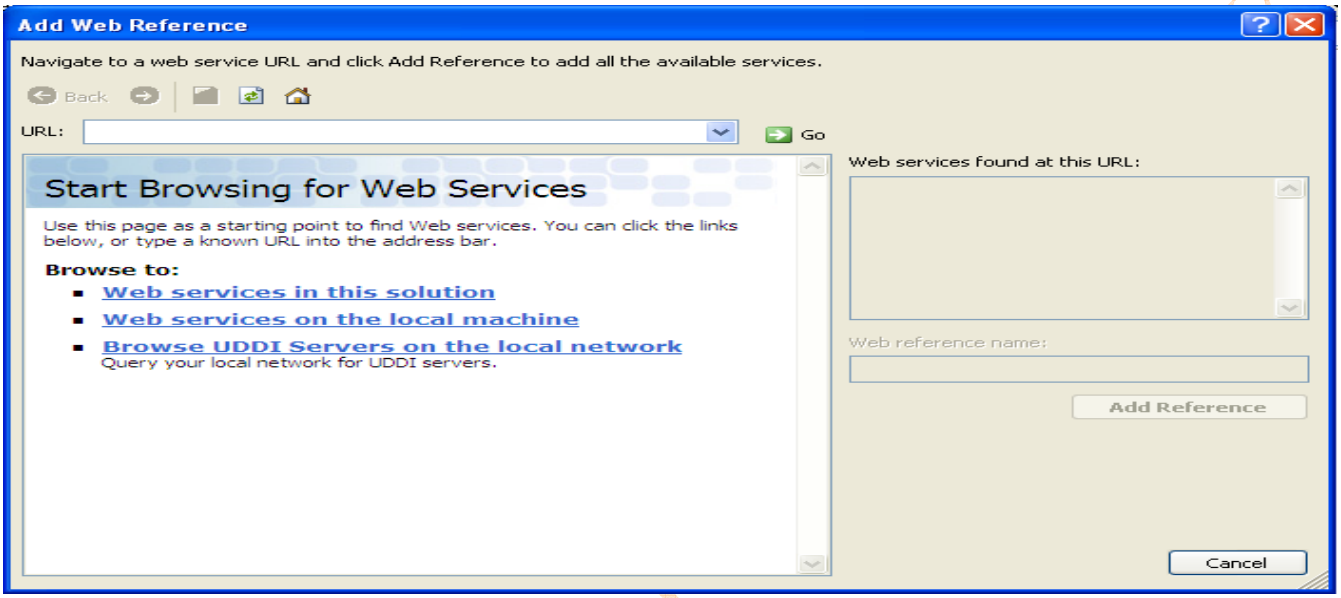
Then, when you click the *Invoke* button, the method is executed and the XML that's generated as a result is displayed:



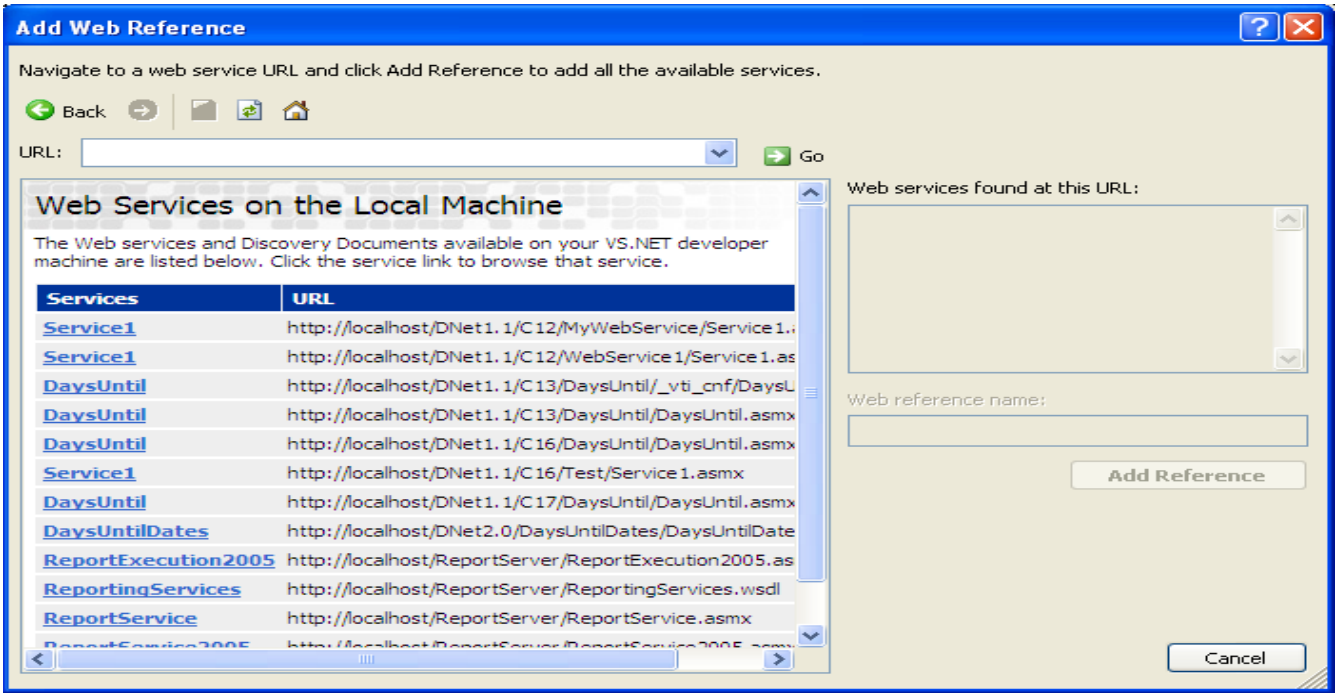
How to consume and use a web service

After you build and test a web service, you can use it from any projects that required the service it provides. To do that, you first have to add a reference to the web service. Then, you can create an instance of it and use it like any other class.

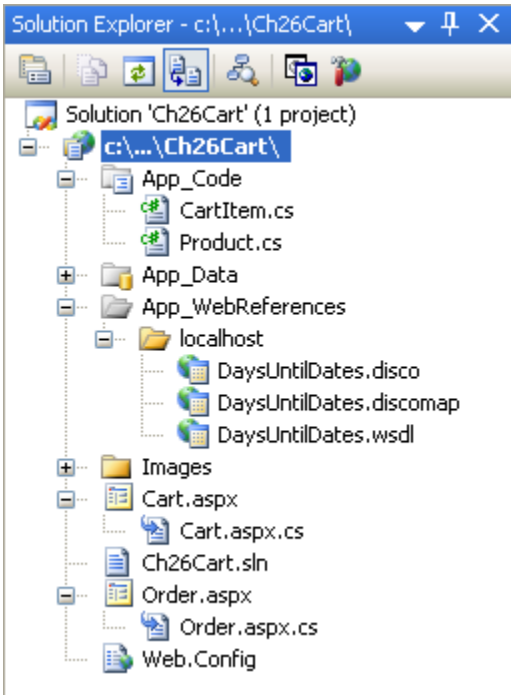
To add a reference to a web service to a project, right-click the project file in the Solution Explorer and select the *Add Web Reference* command from the shortcut menu to display the *Add Web* reference dialog box.



If the service is on your local server, for example, you can click the *Web Services* on *local Machine* link and a list of all the web services on the local server will be listed. Then you can click the link for the web services you want to add.



After you add a reference to a web service, it appears in the folder that has the same name as the server that contains the web service. This folder is subordinate to the *App_WebReferences* folder, which is added automatically if it doesn't already exist:



The first two files for the web service, *DaysUntilDates.disco* and *DaysUntilDates.discomap*, contain information about the location of the web service that let other users "discover" the web service. The third file, *DaysUntilDates.wsdl*, is an XML document that describes the web service and contains information on how a client can interact with it. This file can be helpful for figuring out how to interact with a web service if you find that the documentation for the service is incomplete or confusing.

To create a web service, you create an instance of it and assign it to an object variable as shown here:

```
localhost.DaysUntilDates daysUntilDates = new localhost.DaysUntilDates();
```

After you assign an instance of the web service to an object variable, you can refer to any web method defined by the service using standard techniques:

```
lblDaysUntilHalloween.Text = daysUntilDates.DaysUntilHalloween().ToString();
```