

I M.Sc.(CS) Distributed Technologies

Unit IV Advanced Features of ASP.NET

1. What are the Advanced Features of ASP.NET?

- ⇒ ASP.Net is an exciting server side technology used for developing web based applications
- ⇒ It uses the namespaces, classes and methods provided by the .NET Framework
- ⇒ There are the following features of the ASP.NET
 - MasterPage
 - Data Control
 - Themes and control
 - Navigation Control
 - New Server Control
- ⇒ **Master Page:**
 - is use to define common structure and interface elements for any website
 - such as a page header, footer or navigation bar, in a common location
 - can shared many different pages within the web site
- ⇒ **Data Control**
 - Data access can possible by using the new **data-bound** and **data source controls**
 - New data source controls to represent different data backend, such as SQL, business objects, and XML
 - New data-bound controls for rendering common UI for data, such as grids, details, and formview
- ⇒ **Themes and control:**
 - Theme to control the appearance of both the HTML elements and ASP.NET controls that appear in a page
 - A Theme folder can contain a variety of different types of files, including images and text files
 - Also can organize the contents of a Theme folder by adding multiple subfolders to a Theme folder
 - There are the most important types of files in a Theme:

- Skin files
- Cascading style sheet files
- A Theme can contain one or more Skin files
 - A Skin enable to modify any of the proprieties of an ASP.net control that have an effect on its appearance

⇒ **Navigation control:**

- Provide a common UI for navigating between pages in our website, such as
 - Treeview
 - Menus
 - SiteMapPath

⇒ **New server control:**

- Server controls are tags that are understood by the server
- There are three kinds of server controls:
 - HTML Server Controls - Traditional HTML tags
 - Web Server Controls – New ASP.NET tags
 - Validation Server Controls – For Input validation
- That enables powerful declarative support for data access
 - Login security
 - Wizard navigation
 - Image generation
 - Menus
 - Tree views
 - Portals
 - And more

Security in ASP.NET

2. Discuss ASP.NET Security Features.

➤ Security Feature in ASP.NET 2.0

- ⇒ Security is an important attribute of any ASP.NET application
- ⇒ *The authentication* and *authorization* of users and resistance against the malicious attacks are important tasks in web applications
- ⇒ ASP.NET 2.0 introduced a new membership and role management service that provides both authentication and authorization services and management of users who access our application without building any tables or writing any code

➤ Security Model

- ⇒ ASP.NET 2.0 provides two providers in new security model

- *Membership Provider*

- *Role Provider*

➤ Membership Provider:

- ⇒ The extensible Membership provider framework can register and authenticate new users
- ⇒ Membership provider uses Microsoft SQL Server as the back-end store
- ⇒ This abstract class derived from *ProviderBase* class
- ⇒ There are two Membership providers

- *SqlMembership Provider:*

- stores Membership information in a SQL Server database
- can also create custom Membership provider using any OLEDB DataSource or XML DataSource

- *AccessMembership Provider:*

- stores Membership information in Access database

➤ Role Provider:

- ⇒ Role Provider are used to manage user roles like creating new roles for users

➤ The SqlMembership Provider support several provider specific attributes

- **ApplicationName:**

- Need to host multiple applications on the same Web server

- Can use this property to isolate the users who are associated with the different applications
- **ConnectionStringName:**
 - The name of a database connection string defined in the **ConnectionStrings** section of the Web Configuration file
- **Description:**
 - A description of the provider definition
- **EnablePasswordReset:**
 - When true, users can reset their password to a randomly generated password
- **EnablePasswordRetrieval:**
 - When true, user passwords can be retrieved from the **Membership** provider
- **PasswordFormat:**
 - This property has three possible values:
 - Clear
 - Encrypted
 - Hashed
 - When passwords are hashed, the original passwords cannot be retrieved from the Membership provider
- **RequiresQuestionAndAnswer:**
 - When true, the user must answer a password retrieval question before the user password can be reset or retrieved
- **RequiresUniqueEmail:**
 - When true, a unique e-mail address must be associated with each user

3. Explain in detail the Different Login and Password Server Controls:

⇒ Different Login and Password server controls are :

- Login
- LoginStatus
- LoginName
- ChangePassword
- PasswordRecovery
- LoginView
- CreateUserWizard

➤ Login Server Control:

⇒ The Login server control display standard login interface for user *authentication*

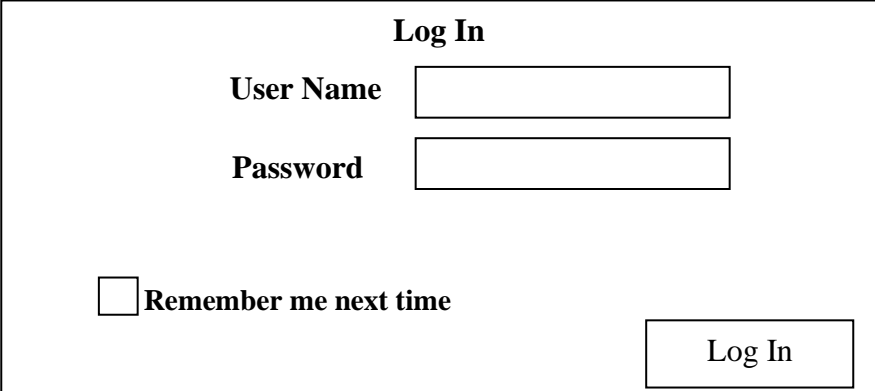
⇒ The login control can be used as a standalone control on a *main* and *home page* or can use it on a dedicated login page

Source Code (default.aspx):

```
<form id="form1" runat="server">
  <asp:Login ID="Login1" runat="server">
  </asp:Login>
</form>
```

Login control as default.asp page listing

Design Page:



The screenshot shows a design view of a login page. At the top right, the text "Log In" is displayed. Below it, there are two labels: "User Name" and "Password". Each label is followed by a corresponding input field. The "User Name" field is a standard text box, and the "Password" field is a password box with a small eye icon to toggle visibility. Below these fields, there is a checkbox with the text "Remember me next time" next to it. In the bottom right corner, there is a button labeled "Log In".

Login control properties:

- **FailureText:** used to control the content and appearance of the text that is displayed when a login attempt fails
- **CreateUserUrl:** used to create links to registration page to create user
- **PasswordRecoveryUrl:** used to create links to password recovery page
- **VisibleWhenLoggedIn:** enable to automatically hide the Login control when the user is already authenticated

- **DestinationPageUrl:** sets the name of the page that the user will be redirected to after logging in.

➤ **LoginStatus Server Control:**

- ⇒ This control enables user to click a link to *Login or Logout* of web application
- ⇒ This controls displays one of two links

Source code:

```
<form id="form1" runat="server">  
    <asp:LoginStatus ID="LoginStatus1" runat="server" />  
</form>
```

➤ **LoginName Server Control:**

- ⇒ Display the username of the authenticated user

Source code:

```
<form id="form1" runat="server">  
    <asp:LoginName ID="LoginName1" runat="server" />  
</form>
```

➤ **ChangePassword Server Control:**

- ⇒ Enables user to change their passwords
- ⇒ This control displays textboxes for entering the original password and entering a new password

Design Page (default.aspx)

Change Your Password

Password	<input type="text"/>
New Password	<input type="text"/>
Confirm New Password	<input type="text"/>

The Confirm New Password must match the New Password entry

Source code:

```
<form id="form1" runat="server">  
    <asp:ChangePassword ID="ChangePassword1" runat="server">  
    </asp:ChangePassword>  
</form>
```

➤ **PasswordRecovery Server Control**

⇒ This control can be used to *retrieve* (recovery) password for the user

Design Page for PasswordRecovery control

Forget Your Password?
Enter your UserName to receive your Password.

User Name

Source code:

```
<form id="form1" runat="server">  
    <asp:PasswordRecovery ID="PasswordRecovery1" runat="server">  
    </asp:PasswordRecovery>  
</form>
```

➤ LoginView Server Control:

⇒ Can be used to display different content depending on the role of the current user

Source code:

```
<form id="form1" runat="server">  
    <asp:LoginView ID="LoginView1" runat="server">  
    </asp:LoginView>  
</form>
```

➤ CreateUserWizard Server Control:

⇒ Enables to create a *Standard User Registration Page*

⇒ Used to allow users to create a new user entry in the membership system

Design Page for the CreateUserWizard control

Sign Up For Your New Account

User Name

Password

Confirm Password

E-mail

Security Question

Security Answer

The Password and Confirmation Password must match.

Source code:

```
<form id="form1" runat="server">
  <asp:CreateUserWizard ID="CreateUserWizard1" runat="server">
    <WizardSteps>
      <asp:CreateUserWizardStep ID="CreateUserWizardStep1" runat="server">
      </asp:CreateUserWizardStep>
      <asp:CompleteWizardStep ID="CompleteWizardStep1" runat="server">
      </asp:CompleteWizardStep>
    </WizardSteps>
  </asp:CreateUserWizard>
</form>
```

Example Program for the Login Control:

Source Code:

```
<% @ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Untitled Page</title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:Login ID="Login1" runat="server" Height="239px" Width="595px">
    </asp:Login>
  </form>
</body>
</html>
```

View Code:

```
Imports System.Data
Imports System.Data.SqlClient
Partial Class _Default
  Inherits System.Web.UI.Page
  Dim con As New SqlConnection
  Dim ad As New SqlDataAdapter
  Dim ds As New DataSet
  Protected Sub Login1_Authenticate(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.AuthenticateEventArgs) Handles Login1.Authenticate
    con = New SqlConnection
    con.ConnectionString = "Datasource"
    con.Open()
    ad = New SqlDataAdapter("select * from test Name='" & Login1.UserName & "' and No=" &
Login1.Password, con)
    ds = New DataSet
    ad.Fill(ds)
    If ds.Tables(0).Rows.Count > 0 Then
      Response.Write("You are authenticate user")
    Else
```



```
    Response.Write("Incorrect user")  
End If  
End Sub  
End Class
```

State Management in ASP.NET

4. What purposes of the State Management?

⇒ Statement is the process by which we maintain state and page information over multiple requests for the same or different pages

➤ Types of State Management:

⇒ There are Two types of State Management

1. Client-Side State Management

2. Server-Side State Management

1. Client-Side State Management:

⇒ This stores information on the client's computer by embedding the information into a web page, a Uniform Resource Locator (URL) or a cookie

⇒ This techniques available to store the state information at the client end are listed below:

- a. View State
- b. Control State
- c. Hidden Fields
- d. Cookies
- e. Query Strings

a. View State

- Uses view state to track the values in the controls
- This property provides dictionary object for retaining values between multiple request for the same page
- When the page is processed, the current state of the page and controls is hashed into a string and saved in the page as a hidden field
- It is used by the Asp.net page framework to automatically save the values of the page and of each control just prior to rendering to the page
- We can store values in view state as well. The following example shows how to store a value in the view state

ViewState("color")="red"

b. Control State

- To store control-state data in order for a control to work properly the control state is used
- That requires ViewState to work properly we should use control-state to ensure other developers don't break our control by disabling view state
- The ControlState property allows we to persist property information that is specific to a control and cannot be turned off like the ViewState property

c. Hidden Fields

- Like View State
- Hidden fields store data in an HTML form without displaying it in the user's browser
- The data is available only when the form is processed

d. Cookies

- A cookie is a small amount of data that is store a value in the user's browser that the browser sends with every page request to the same server
- The most common use of cookies is to identify a single user as he or she visits multiple web pages

e. Query Strings

- A query string is information that is appended to the end of a page URL
- A typical query string might look like the following examples

`http://www.contoso.com/listwidgets.aspx?category=basic&price=100`

2. Server-Side State Management:

⇒ To maintain state information on the server

a. Application State

- Is used to store and retrieve information that can be shared among all user of an application
- **Ex: Application("MS")="welcome message"**
- Application state information is available to all pages, regardless of which user requests a page

b. Session State

- Used to store and retrieve information about particular sessions
- Session state is only maintained for browsers that support cookies
- Both Application state and Session state information is lost when the application restart
- To persist user data between application restart

c. Profile Properties

- To store user-specific data
- Is similar to session state
- Stored in a persistent format and associated with an individual user
- **Profile.postalcode=textbox1.text**

Mobile Application Development in ASP.NET

5. Describe the mobile application design concepts in ASP.NET.

- ⇒ ASP.NET provides a mobile component which allows we to build applications for mobile devices to access web pages
- ⇒ Like **cell phones, digital phones & personal digital assistants**
- ⇒ It is very easy for ASP.NET developer to develop mobile applications

Mobile ASP.NET page structure

- ⇒ Mobile ASP.Net pages are based on the **MobilePage** class which exists in **System.Web.UI.MobileControls** namespace

```
<% page Language="VB" Inherits="System.Web.UI.MobileControls.MobilePage"%>
```

- ⇒ Uses on a mobile ASP.NET page come from their own namespace, we need to include that namespace on each of our mobile ASP.Net pages

```
<% @ Register TagPrefix="Mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
```

Example :

```
<mobile:Form id="Page1" runat="Server">
  <mobile:label id="Label1" runat="Server" Text="The text on the first page"/>
  <mobile:Link id="Link1" runat="Server" Text="View Next Page" NavigateUrl="Page2"/>
</mobile:Form1>
```

```
<mobile:Form id="Page2" runat="Server">
  <mobile:label id="Label2" runat="Server" Text="The text on the second page"/>
  <mobile:Link id="Link2" runat="Server" Text="View Previous Page" NavigateUrl="Page1"/>
```

</mobile:Form>

Creating Web Application in ASP.NET

1. Click to open Microsoft Visual Studio 2008
2. On the **File Menu**, choose New, and then choose Web Site. The New Web Site dialog box appears
3. Under Visual Studio installed templates, select ASP.NET Web Sites
4. Click **Browse**. The choose Location dialog box appears
5. Location **File System** and **LRC**
6. Language **Visual C#**
7. Click **OK** button
 - ⇒ A Default.aspx is added in our solution

Creating Mobile Web page in Application

1. Right-click the **Default.aspx** page in **Solution Explorer** and choose **Delete**
2. Click **OK** in the dialog box
3. Right-click the application in **Solution Explorer** and choose **Add New Item**
4. Choose **Mobile Web Form** under **Visual Studio Installed Templates**
5. Name Calculator.aspx
6. Choose Language **Visual C#**
7. Check place code in separate file
8. Click **Add** in the dialog box

Mobile Controls

1. Label Control

⇒ To display text on an ASP.NET page

```
<mobile:Form id="Form1" runat="server" >  
  <mobile:Label id="Label1" runat="server" StyleReference="title" Text="Title Text style"/>  
  <mobile:Label id="Label2" runat="server" Alignment="Center" Text="The word"/>  
</mobile:Form>
```

2. Link Control

⇒ Provides, to allow the visitors to navigate to another page or another form within the current page

```
<mobile:Form id="Form1" runat="server" >  
  <mobile:Label id="Label1" runat="server" Text="Link Test Page"/>  
  <mobile:Link id="Link1" runat="server" Text="click here!"  
NavigateUrl="http://www.google.com" />  
  <mobile:Link id="Link2" runat="server" Text="or here!"  
NavigateUrl="http://www.gmail.com" Alignment="Right"/>  
</mobile:Form>
```

3. Call Control

⇒ Use the Call Control to make it easier for the visitor to call the contact

```
<mobile:Form id="Form1" runat="server" >  
  <mobile:Label id="Label1" runat="server" Text="Call Test Page"/>  
  <mobile:Call id="call1" runat="server" Text="Call Home!" phoneNumber="999999"  
AlternateUrl="http://www.google.com" AlternateFormat="call home(0) with the number(1)"/>  
</mobile:Form>
```

⇒ That supports dialing phone numbers. If the visitor were to click the call control, they would e asked to confirm that they wanted to make the call

4. Image Control

⇒ to display a graphic on our mobile ASP.NET page

```
<mobile:Image id="Image1" runat="server" AlternateText="Your device does not support  
graphics">  
</mobile:Image>
```

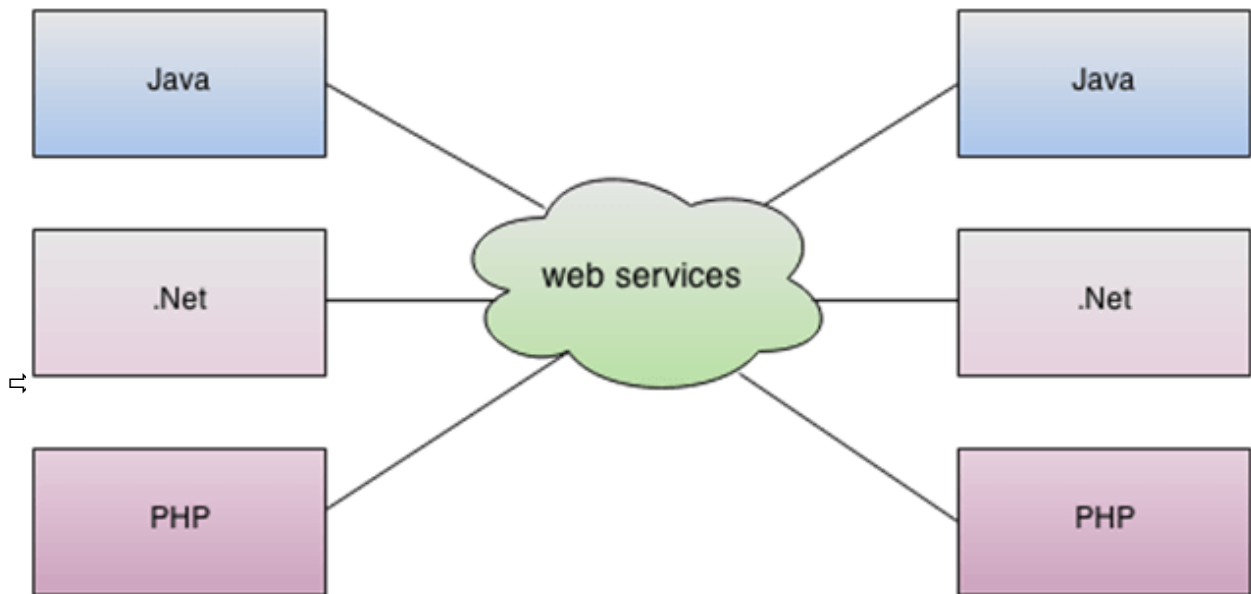
Unit V

Web Services

What is Web Services

A **Web Service** is can be defined by following ways:

- It is a client-server application or application component for communication.
- The method of communication between two devices over the network.
- It is a software system for the interoperable machine to machine communication.
- It is a collection of standards or protocols for exchanging information between two devices or application.



an see in the figure, Java, .net, and PHP applications can communicate with other applications through web service over the network.

⇒ For example, the Java application can interact with Java, .Net, and PHP applications. So web service is a language independent way of communication.

Types of Web Services

There are mainly two types of web services.

1. SOAP web services.
2. RESTful web services.

Web Service Components

There are three major web service components.

1. SOAP
2. WSDL
3. UDDI

SOAP

- SOAP is an acronym for Simple Object Access Protocol.
- SOAP is a XML-based protocol for exchanging information between applications within a distributed environment
- SOAP is a W3C recommendation for communication between applications.
- SOAP is XML based, so it is platform independent and language independent. In other words, it can be used with Java, .Net or PHP language on any platform.

WSDL

- WSDL is an acronym for Web Services Description Language.
- WSDL is a xml-based language for describing web service
- WSDL is XML document containing information about web services such as method name, method parameter and how to access it.
- WSDL is a part of UDDI. It acts as a interface between web service applications.
- WSDL is pronounced as wiz-dull.

UDDI

- UDDI is an acronym for Universal Description, Discovery and Integration is specification for a registry of information for web services
- UDDI is a XML based framework for describing, discovering and integrating web services.
- UDDI is a directory of web service interfaces described by WSDL, containing information about web services.

The Role of Web Services in Distributed Computing

- ⇒ We can now use ASP.NET to create Web Services based on industrial standards including XML, SOAP, and WSDL.
- ⇒ A Web Service is a software program that uses XML to exchange information with other software via common internet protocols. In a simple sense, Web Services are a way of interacting with objects over the Internet.

➤ A web service is

- Language Independent.
- Protocol Independent.
- Platform Independent.
- It assumes a stateless service architecture.
- Scalable (e.g. multiplying two numbers together to an entire customer-relationship management system).
- Programmable (encapsulates a task).
- Based on XML (open, text-based standard).

- Self-describing (metadata for access and use).
- Discoverable (search and locate in registries)- ability of applications and developers to search for and locate desired Web services through registries. This is based on UDDI.

➤ **Web Service History**

- Microsoft coined the term "Web services" in June 2000, when the company introduced Web services as a key component of its .Net initiative, a broad new vision for embracing the Internet in the development, engineering and use of software.
- As others began to investigate Web services, it became clear that the technology could revolutionize (be the next stage in) distributed computing.
- Web services encompass a set of related standards that can enable any two computers to communicate and exchange data via a network, such as the Internet.
- The primary standard used in Web services is the Extensible Markup Language (XML) developed by the World Wide Web Consortium (W3C).
- Developers use XML tags to describe individual pieces of data, forming XML documents, which are text-based and can be processed on any platform.
- XML provides the foundation for many core Web services standards (SOAP, WSDL, and UDDI) and vocabularies (XML-based markup for a specific industry or purpose).
- Almost every type of business can benefit from Web services such as expediting software development, integrating applications and databases, and automating transactions with suppliers, partners, and clients.

➤ **Key Web Service Technologies**

- **XML**- Describes only data. So, any application that understands XML-regardless of the application's programming language or platform has the ability to format XML in a variety of ways (well-formed or valid).
- **SOAP**- Provides a communication mechanism between services and applications.
- **WSDL**- Offers a uniform method of describing web services to other programs.
- **UDDI**- Enables the creation of searchable Web services registries.

When these technologies are deployed together, they allow developers to package applications as services and publish those services on a network.

➤ **Web services advantages**

- Use open, text-based standards, which enable components written in various languages and for different platforms to communicate.
- Promote a modular approach to programming, so multiple organizations can communicate with the same Web service.
- Comparatively easy and inexpensive to implement, because they employ an existing infrastructure and because most applications can be repackaged as Web services.
- Significantly reduce the costs of enterprise application (EAI) integration and B2B communications.
- Implemented incrementally, rather than all at once which lessens the cost and reduces the organizational disruption from an abrupt switch in technologies.
- The Web Services Interoperability Organization (WS-I) consisting of over 100 vendors promotes interoperability.

➤ **Web Services Limitations**

- SOAP, WSDL, UDDI- require further development.

- Interoperability.
- Royalty fees.
- Too slow for use in high-performance situations.
- Increase traffic on networks.
- The lack of security standards for Web services.
- The standard procedure for describing the quality (i.e. levels of performance, reliability, security etc.) of particular Web services – management of Web services.
- The standards that drive Web services are still in draft form (always will be in refinement).
- Some vendors want to retain their intellectual property rights to certain Web services standards.

WSDL, UDDI, SOAP concepts involved in Web Services

➤ **SOAP :** -

- ⇒ Simple Object Access Protocol(SOAP) is a protocol that is used to exchange structured information at the time of implementing a web service.
- ⇒ SOAP is relied on XML. Message format of SOAP usually relies on another protocol of different application layers.
- ⇒ Among these the most notable application layer is Remote Procedure Call(RPC) and HTTP. SOAP forms the foundation layer for web services protocol stack.
- ⇒ This stack provides the basic framework for messaging on which the web services are built.

➤ **WSDL :**

- ⇒ Web Service Definition Language(WSDL) is used to describe a web service based on XML
- ⇒ WSDL is used for describing web services and to locate the services. WSDL consists of the information on what the service is all about, its residing location and the way of invocation the service.

➤ **UDDI :**

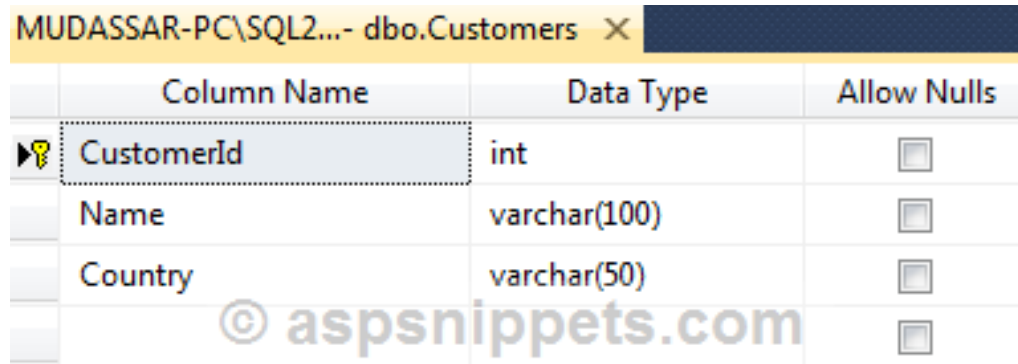
- ⇒ Universal Discovery Description Integration(UDDI) is used to publish and discover the information about the web services,
- ⇒ UDDI is a specification. It is an XML based standard.
- ⇒ This standard is used for describing, publishing, and finding the services.
- ⇒ These services are found in a distributed environment through the use of a server called registry server.

Connecting a Web Services to Database in ASP.NET

⇒ a simple Web Method will be created in Web Service to fetch data from database and the fetched data will be displayed in GridView.

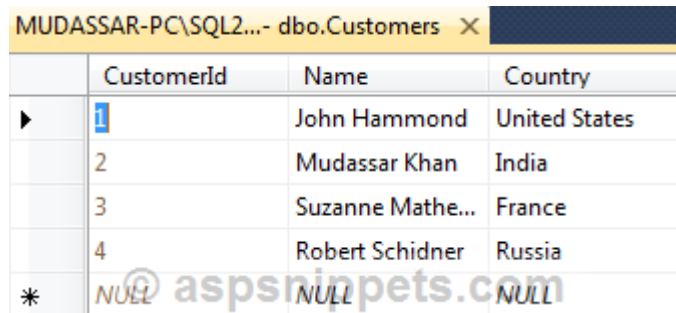
➤ Database

⇒ We have made use of the following table Customers with the schema as follows.



Column Name	Data Type	Allow Nulls
CustomerId	int	<input type="checkbox"/>
Name	varchar(100)	<input type="checkbox"/>
Country	varchar(50)	<input type="checkbox"/>

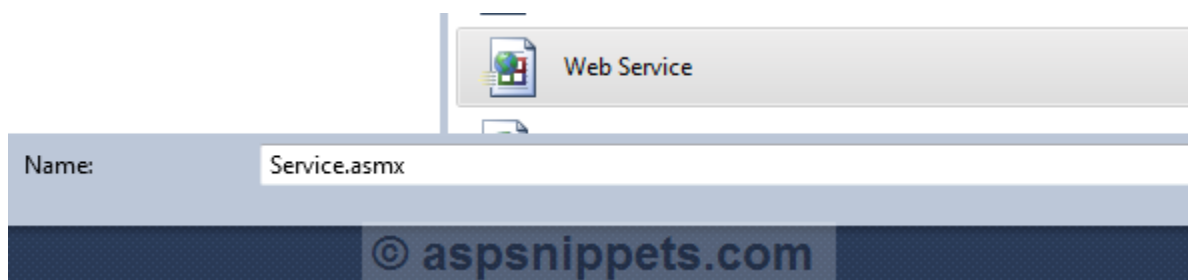
⇒ We have already inserted few records in the table.



CustomerId	Name	Country
1	John Hammond	United States
2	Mudassar Khan	India
3	Suzanne Mathe...	France
4	Robert Schidner	Russia
* NULL	NULL	NULL

➤ Building the Web Service

1. First step is to add a Web Service.



2. Once the Web Service is added. You will need to add the following namespaces.

C#

```
using System.Data;  
using System.Configuration;
```

```
using System.Data.SqlClient;
```

VB.Net

```
Imports System.Data
```

```
Imports System.Configuration
```

```
Imports System.Data.SqlClient
```

3. Third step is to add the WebMethod for getting data from database.

C#

```
[WebMethod]
```

```
public DataTable Get()
```

```
{  
    string constr = ConfigurationManager.ConnectionStrings["constr"].ConnectionString;  
    using (SqlConnection con = new SqlConnection(constr))  
    {  
        using (SqlCommand cmd = new SqlCommand("SELECT * FROM Customers"))  
        {  
            using (SqlDataAdapter sda = new SqlDataAdapter())  
            {  
                cmd.Connection = con;  
                sda.SelectCommand = cmd;  
                using (DataTable dt = new DataTable())  
                {  
                    dt.TableName = "Customers";  
                    sda.Fill(dt);  
                    return dt;  
                }  
            }  
        }  
    }  
}
```

VB.Net

```
<WebMethod()> _
```

```
Public Function Get() As DataTable
```

```
    Dim constr As String = ConfigurationManager.ConnectionStrings("constr").ConnectionString
```

```
    Using con As New SqlConnection(constr)
```

```
        Using cmd As New SqlCommand("SELECT * FROM Customers")
```

```
            Using sda As New SqlDataAdapter()
```

```
                cmd.Connection = con
```

```
                sda.SelectCommand = cmd
```

```
                Using dt As New DataTable()
```

```
                    dt.TableName = "Customers"
```

```
                    sda.Fill(dt)
```

```
                Return dt
```

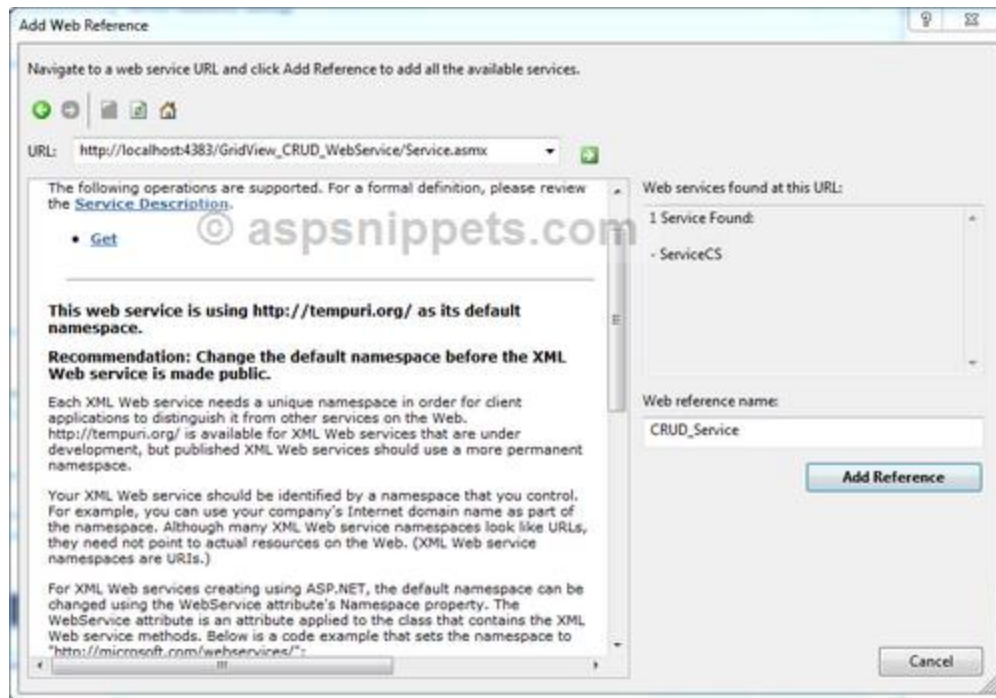
```
            End Using
```

```
        End Using
```

```
    End Using
```

```
End Function
```

4. Finally you will need to add the Web Reference of the Web Service we just created to the project as shown below.



HTML Markup

The HTML Markup consists of an ASP.Net GridView which will be populated using Web Service.

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="false">
<Columns>
  <asp:TemplateField HeaderText="Name" ItemStyle-Width="50">
    <ItemTemplate>
      <asp:Label ID="lblId" runat="server" Text="<%# Eval("CustomerId") %>"></asp:Label>
    </ItemTemplate>
  </asp:TemplateField>
  <asp:TemplateField HeaderText="Name" ItemStyle-Width="150">
    <ItemTemplate>
      <asp:Label ID="lblName" runat="server" Text="<%# Eval("Name") %>"></asp:Label>
    </ItemTemplate>
  </asp:TemplateField>
  <asp:TemplateField HeaderText="Country" ItemStyle-Width="150">
    <ItemTemplate>
      <asp:Label ID="lblCountry" runat="server" Text="<%# Eval("Country") %>"></asp:Label>
    </ItemTemplate>
  </asp:TemplateField>
</Columns>
</asp:GridView>
```

Binding the GridView using Web Service

The GridView is populated from the database inside the Page Load event of the page.

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!this.IsPostBack)
    {
        this.BindGrid();
    }
}

private void BindGrid()
{
    CRUD_Service.ServiceCS service = new CRUD_Service.ServiceCS();
    GridView1.DataSource = service.Get();
    GridView1.DataBind();
}
```

VB.Net

```
Protected Sub Page_Load(sender As Object, e As EventArgs) Handles Me.Load
    If Not Me.IsPostBack Then
        Me.BindGrid()
    End If
End Sub

Private Sub BindGrid()
    Dim service As New CRUD_ServiceVB.ServiceVB()
    GridView1.DataSource = service.[Get]()
    GridView1.DataBind()
End Sub
```

Following is the GridView populated using Web Service.

Customer Id	Name	Country
1	John Hammond	United States
2	Mudassar Khan	India
3	Suzanne Mathews	France
4	Robert Schidner	Russia

Example Program : Develop a web service to fetch a data from a table and send it across to the client

Web Service (Service.vb)

Imports System.Web

Imports System.Web.Services

Imports System.Web.Services.Protocols

Imports System.Data

Imports System.Data.SqlClient

<WebService(Namespace:="http://tempuri.org/")> _

<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _

<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _

Public Class Service

Inherits System.Web.Services.WebService

<WebMethod()> _

Public Function find(**ByVal** n1 **As** String) **As** String

Dim con **As** **New** SqlConnection

Dim ad **As** **New** SqlDataAdapter

Dim ds **As** **New** DataSet

Dim ret **As** String

con = **New** SqlConnection

con.ConnectionString = "Data Source=.\\SQLEXPRESS;AttachDbFilename=C:\\Documents and Settings\\student\\My Documents\\Visual Studio

2005\\WebSites\\WebSite30\\App_Data\\Database.mdf;Integrated Security=True;User Instance=True"

con.Open()

ad = **New** SqlDataAdapter("Select * from book where code=" & n1, con)

ds = **New** DataSet

ad.Fill(ds)

If ds.Tables(0).Rows.Count > 0 **Then**

ret = ds.Tables(0).Rows(0).Item("bname")

Else

ret = "No Such Record"

End If

Return ret

End Function

End Class


```
</body>
</html>
```

View Code (default.aspx.vb)

Partial Class _Default

Inherits System.Web.UI.Page

Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click

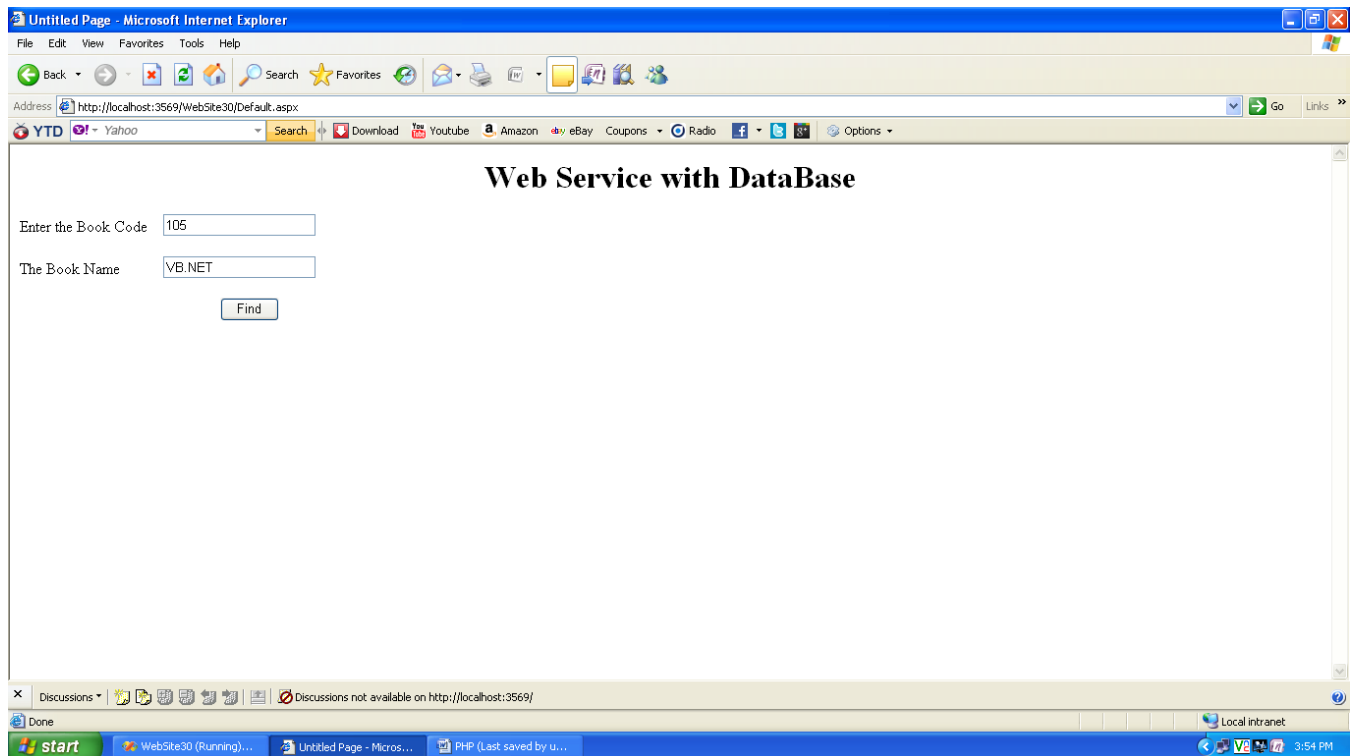
Dim a As New Service()

TextBox2.Text = a.find(TextBox1.Text)

End Sub

End Class

Output:



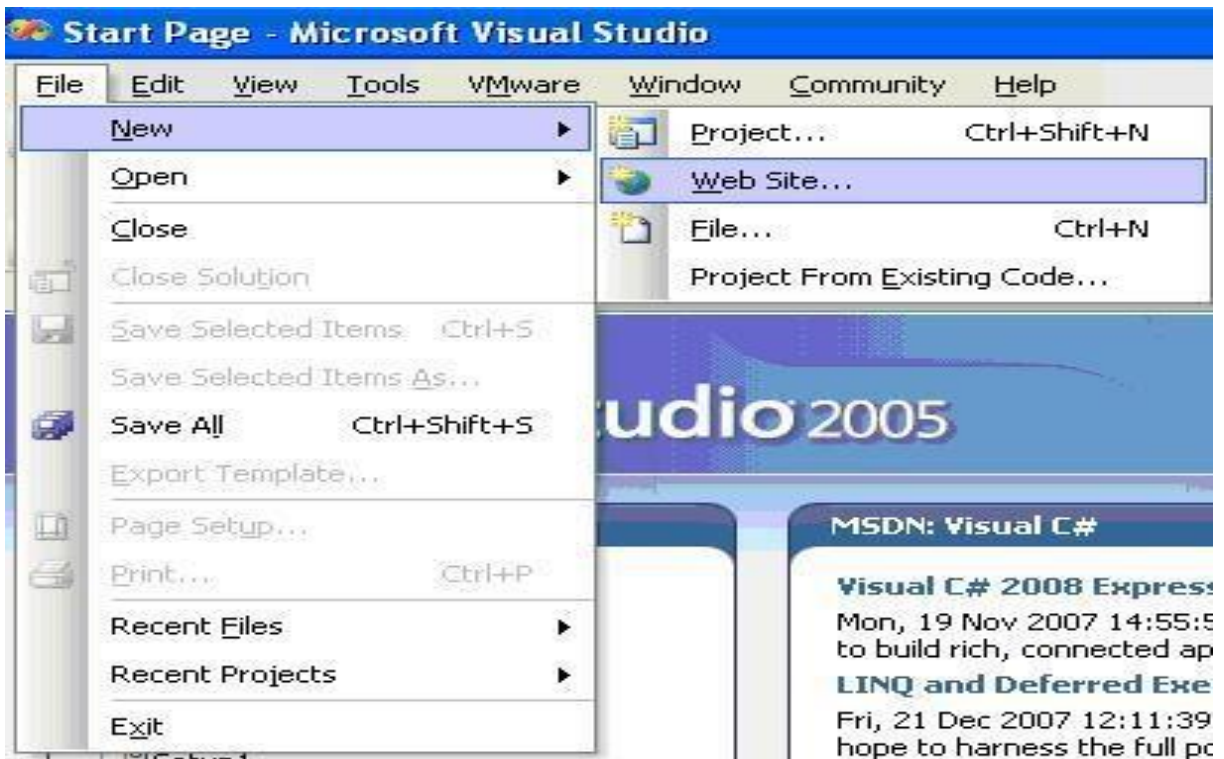
Accessing a Web Service through an ASP.Net Application

- ⇒ Web services signal a new age of trivial distributed application development. While
- ⇒ Web services are not intended nor do they have the power to solve every distributed application problem, they are an easy way to create and consume services over the Internet.
- ⇒ One of the design goals for Web Services is to allow companies and developers to share services with other companies in a simple way over the Internet.
- ⇒ Web services take Web applications to the next level.
- ⇒ Using Web services, your application can publish its function or message to the rest of the world.
- ⇒ Web services use XML to code and decode your data and SOAP to transport it using open protocols.
- ⇒ With Web services, your accounting departments Win 2K servers' billing system can connect with your IT suppliers UNIX server.
- ⇒ Using Web services, you can exchange data between different applications and different platforms.
- ⇒ With Microsoft .NET platform, it is a simple task to create and consume Web Services. Simple Steps to Consume a Web Service

1. Create a Web Site project
2. Add a Web Reference
3. Call the Web services APIs inside the code

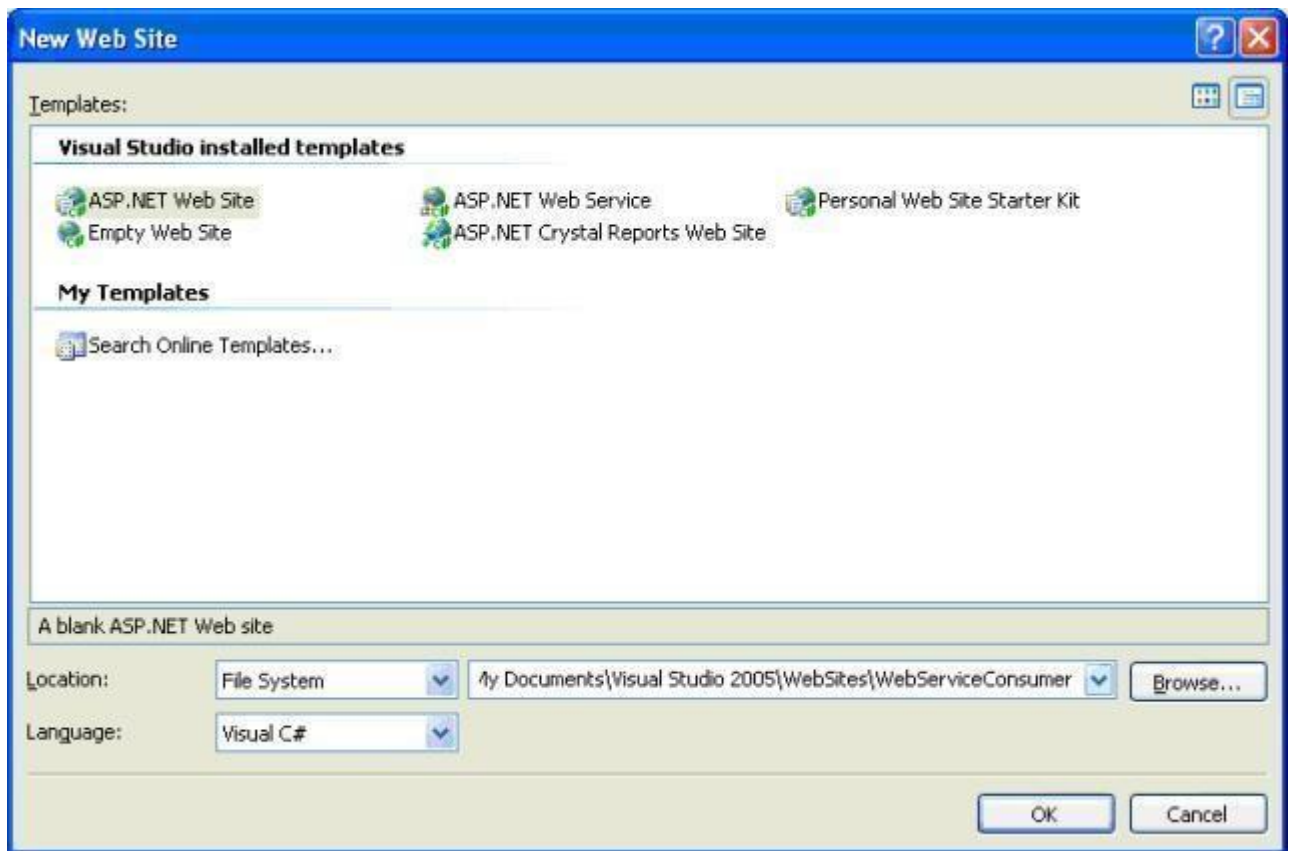
First Step: Create a Web Site Project

1. To create a new Web Site project, choose New from File menu, then choose Web Site as shown below:



2. C

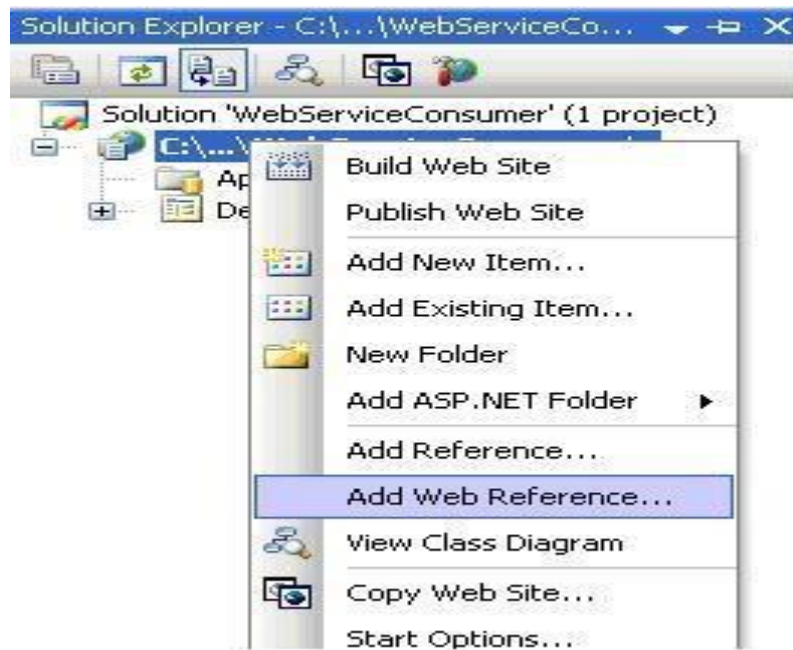
Choose ASP.NET Web Site. Name the project and click OK:



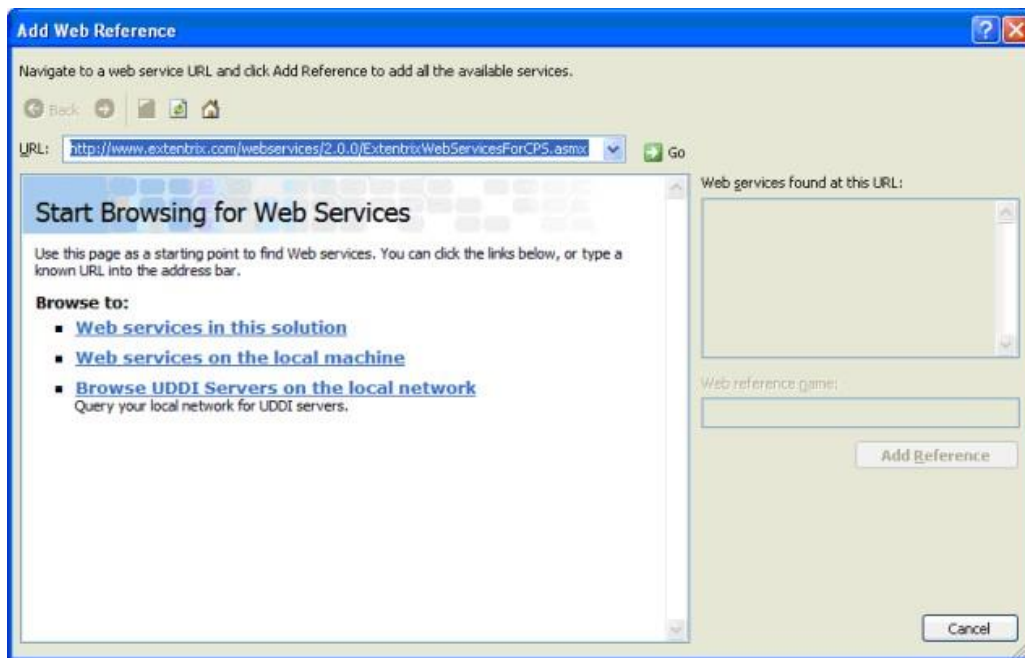
Second Step: Add a Web Reference

After creating the Web Site project, it's time to add a Web reference for our Web service.

1. In the solution explorer, right click the project node, choose Add Web Reference:

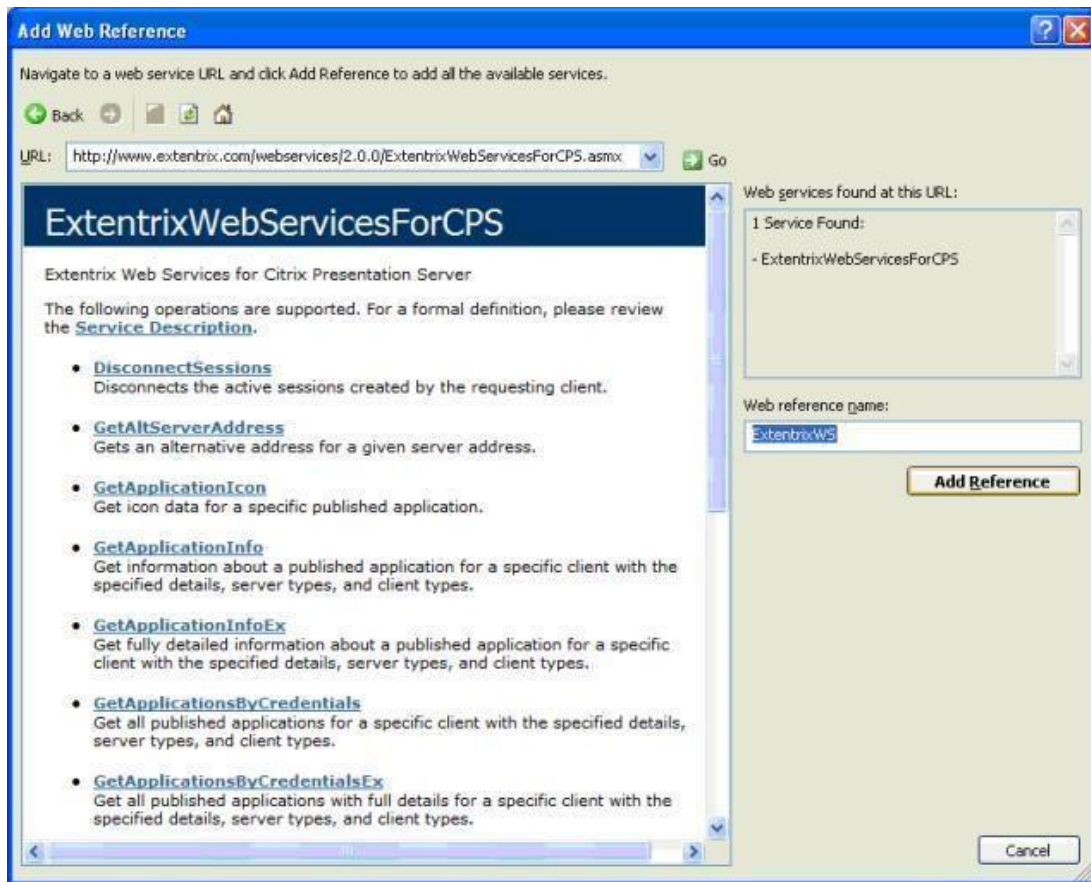


2. A new window with Add Web Reference title will be opened:



After clicking the Go button, you will see the Web services APIs.

3. Set a name for your Web service reference in the Web reference name field and click Add Reference:



Third Step: Call the Web Services APIs Inside the Code

After successfully adding to the Web service, now we are ready to call the Web services APIs inside our project.

Example Program : Develop a web service that has an ASP.NET client

Web Service (Service.vb)

Imports System.Web

Imports System.Web.Services

Imports System.Web.Services.Protocols

<WebService(Namespace:="http://tempuri.org/")> _

<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _

<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _

Public Class Service

Inherits System.Web.Services.WebService

<WebMethod()> _

Public Function sum(**ByVal** n1 **As Integer**, **ByVal** n2 **As Integer**) **As Integer**

Return n1 + n2

End Function

Public Function sub1(**ByVal** n1 **As Integer**, **ByVal** n2 **As Integer**) **As Integer**


```
<asp:Button ID="Button1" runat="server" Text="Addition" />&nbsp;<asp:Button ID="Button2"
    runat="server" Text="Subtraction " />
<asp:Button ID="Button3" runat="server" Text="Multiplication" />
<asp:Button ID="Button4" runat="server" Text="Division" />&nbsp;<asp:Button ID="Button5"
    runat="server" Text="clear" /></p>
</form>
</body>
</html>
```

View Code (default.aspx.vb)

Partial Class _Default

Inherits System.Web.UI.Page

Dim a As New Service()

Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click

 TextBox3.Text = a.sum(Val(TextBox1.Text), Val(TextBox2.Text))

End Sub

Protected Sub Button2_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button2.Click

 TextBox3.Text = a.sub1(Val(TextBox1.Text), Val(TextBox2.Text))

End Sub

Protected Sub Button3_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button3.Click

 TextBox3.Text = a.mul1(Val(TextBox1.Text), Val(TextBox2.Text))

End Sub

Protected Sub Button4_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button4.Click

 TextBox3.Text = a.div1(Val(TextBox1.Text), Val(TextBox2.Text))

End Sub

Protected Sub Button5_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button5.Click

 TextBox1.Text = ""

 TextBox2.Text = ""

 textbox3.text = ""

End Sub

End Class

Output:

