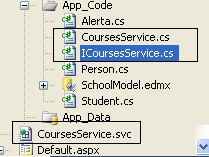
Add a web service type **WCF Service** name **CoursesService.svc**, it will create 3 files:



The file **ICoursesService** is the interface the contract, you define the methods to use and the data to retrieve:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

**// Define methods to interact**

[ServiceContract]

public interface ICoursesService

{

[OperationContract]

CoursesList GetCourses(int departmentID);

}

**// Define data to interact**

[DataContract]

public class CoursesList

{

[DataMember]

public string CourseID { get; set; }

[DataMember]

public string Title { get; set; }

}

If you receive this error when compiling:

**WCF INTERFACES CANNOT DECLARE TYPES**

It is because the class CouseList is inside the interface, take it out, check the brankets ( { } )

If you still have the same message start all over again.

Now work with the file **Courses.cs** in App\_Code Add code to the top of the interface implementation source file to get the connection string from the database.

private string \_cnnString =

ConfigurationManager.ConnectionStrings["SchoolConnectionString"].ToString();

Intclude this libraries

**using System.Data;**

**using System.Data.SqlClient;**

**using System.Configuration;**

Implement the method GetCourses

public class CoursesService : ICoursesService

{

private string \_cnnString = ConfigurationManager.ConnectionStrings["SchoolConnectionString"].ToString();

public CoursesList GetCourses(int departmentID)

{

StringBuilder sql = new StringBuilder();

sql.Append("SELECT CourseID, Title from Courses");

if (departmentID>0) sql.Append("WHERE departmentID = "+departmentID.ToString());

SqlConnection cnn = new SqlConnection(\_cnnString);

SqlCommand cmd = new SqlCommand(sql.ToString(), cnn);

SqlDataAdapter adp = new SqlDataAdapter(cmd);

DataSet ds = new DataSet();

adp.Fill(ds);

CoursesList lista = new CoursesList();

lista.CourseID = ds.Tables[0].Rows[0]["CourseID"].ToString();

lista.Title = ds.Tables[0].Rows[0]["Title"].ToString();

return lista;

}

}

Right-click your **CoursesService.svc** file and choose **Set As Start Page**. Compile and run your service application. Here you will see details on how your WCF service should be called.

If you get the following error

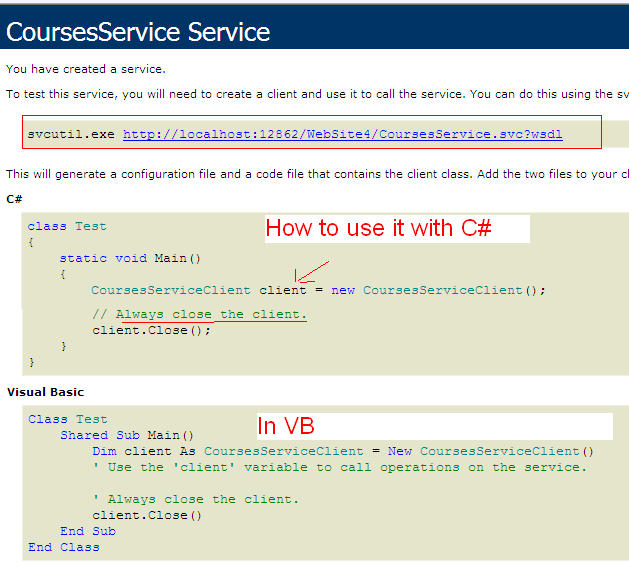
***The service cannot be activated because it does not support ASP.NET compatibility***

Change

[AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]

Or change the web config updte the following variables

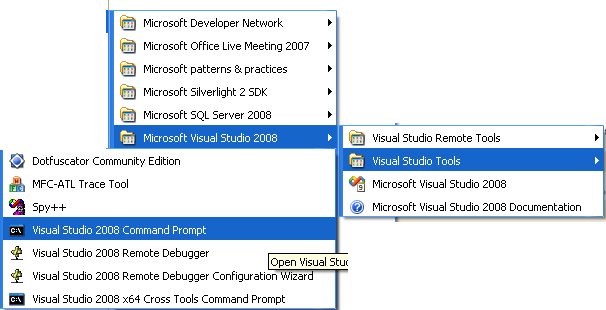
**<serviceHostingEnvironment multipleSiteBindingsEnabled="true" aspNetCompatibilityEnabled="false" />**



Click the link at the top of the webpage to see the WSDL for your WCF service. Click the back button to return to the page. Copy the line that starts with svcutil.exe.

<http://localhost:12862/WebSite4/CoursesService.svc?wsdl>

Generate a test client for your WCF service by using **Svcutil.exe**. Start by opening a Visual Studio command prompt (Start | All Programs | Microsoft Visual Studio 2010 | Visual Studio Tools). Use commands to navigate to a directory to which you want to generate the test client.



Paste the command copied from the top of the webpage into the command window, and press Enter.

**C:\prueba Svcutil.exe** <http://localhost:12862/WebSite4/CoursesService.svc?wsdl>

It will generate the following **output.config** file

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<system.serviceModel>

<bindings>

<basicHttpBinding>

<binding name="BasicHttpBinding\_ICoursesService" closeTimeout="00:01:00"

openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"

allowCookies="false" bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"

maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"

messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"

useDefaultWebProxy="true">

<readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"

maxBytesPerRead="4096" maxNameTableCharCount="16384" />

<security mode="None">

<transport clientCredentialType="None" proxyCredentialType="None"

realm="" />

<message clientCredentialType="UserName" algorithmSuite="Default" />

</security>

</binding>

</basicHttpBinding>

</bindings>

<client>

<endpoint address="http://localhost:12862/WebSite4/CoursesService.svc"

binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding\_ICoursesService"

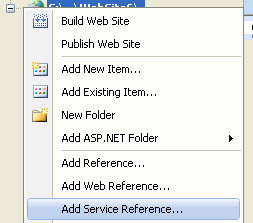
contract="ICoursesService" name="BasicHttpBinding\_ICoursesService" />

</client>

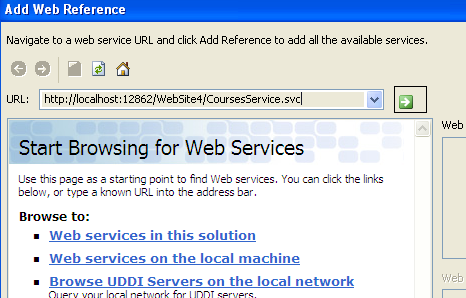
</system.serviceModel>

</configuration>

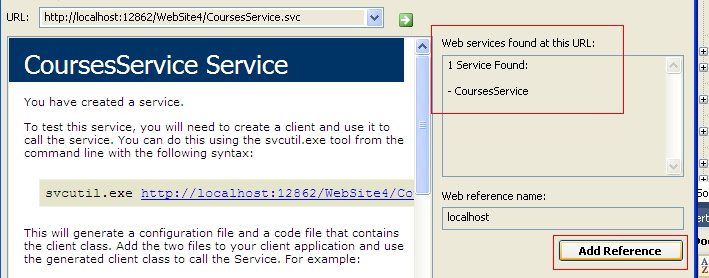
To be allowed to use the web service: Right-click the website and choose **Add Service Reference**.



Because we know already the exact URL, we type it and click the next button, if not we can brown in the local machine, the local project or externally



It was found, not add the reference





Add a control DropDown (DrowDownList1) and a button to submit the request

From the client side you can populate the combo box in the following way

DataSet ds = new DataSet();

CoursesService proxy = new CoursesService();

ds= proxy.GetCourses(0);

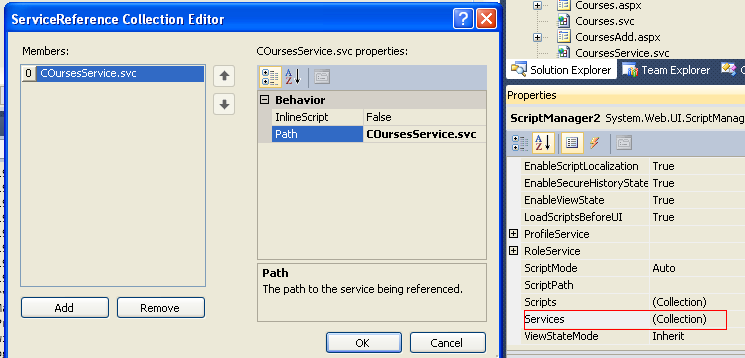
DropDownList1.DataSource = ds;

DropDownList1.DataValueField = "CustomerID";

DropDownList1.DataTextField = "Title";

Working with AJAX

1. Open the page and select the **Design** view.
2. From the **View** menu, select **Toolbox**.
3. Expand the **AJAX Extensions** node and drag and drop a **ScriptManager** on to the Default.aspx page.
4. Right-click the **ScriptManager** and select **Properties**.
5. Expand the **Services** collection in the **Properties** window to open up the **ServiceReference Collection Editor** window.
6. Click **Add**, specify **CoursestService.svc** as the **Path** referenced, and click **OK**.



<asp:ScriptManager ID="ScriptManager1" runat="server">

<Services>

<asp:ServiceReference Path="CoursesService.svc" />

</Services>

</asp:ScriptManager>

1. Expand the **HTML** node in the **Toolbox** and drag and drop an **Input (Button)** on to the Default.aspx page.
2. Double-click the **Button** to access the Javascript code.
3. Pass in the following Javascript code within the <**script**> element.

var service = new SandwichServices.CostService(); // namespace/ service

service.CostOfSandwiches(3, onSuccess, null, null);

**function Button1\_onclick()** {

var service = new CoursesService();

service.GetCourses(0, **onSuccess**, null, null);

}

**function onSuccess(result){**

alert(result);

}

If you would like to use AJAX to populate the dataset use:

function Submit\_onclick() {

$.ajax({

type: "GET", // GET to read data

url: "http://localhost:12862/WebSite4/CoursesService.svc/GetCourses",

data: "{'departmentID': '0'}",

success: function(xml) { // if read is success what to do with xml

$(xml).fild("CoursesList").each(function () { // For each row

var id = $(this).attr("CourseID"); // var id the ID

var tx = $(this).find("Title").text(); // var tx for Name

$("<option>").attr("value", id)

.text(tx).appedTo("DropDownList1"); // Append dropdown

});

}

});

}

$.ajax({

type: "GET",

url: "sites.xml",

dataType: "xml",

success: **function**(xml) {

$(xml).fild("CoursesList").each(function () { // For each row

var id = $(this).attr("CourseID"); // var id the ID

var tx = $(this).find("Title").text(); // var tx for Name

$("<option>").attr("value", id)

.text(tx).appedTo("DropDownList1"); // Append dropdown

});

}

});

<script type="text/javascript">

$(document).ready(function() {

$("#ButtonSearch").click(function() {

$("#empDetails").hide("slow");

var empId = $("#TextBoxEmpId").val();

$.ajax({

type: "POST",

dataType: "json",

contentType: "application/json",

url: "EmployeeService.asmx/GetEmployeeById",

data: "{'employeeId': '" + empId.toString() + "'}",

success: function(data) {

$("#textId").html(data.d.ID);

$("#textName").html(data.d.FullName);

$("#empDetails").show("slow");

},

error: function() { alert("Error calling the web service."); }

});

});

});

**public static void Main()**

**{**

**// Create a ServiceHost.**

**Uri uri = new Uri("http://localhost:8000/EmployeeService1");**

**using (ServiceHost serviceHost = new ServiceHost(typeof(EmployeeService), uri))**

**{**

**serviceHost.Open();**

**Console.WriteLine();**

**Console.WriteLine("The service is ready");**

**Console.WriteLine();**

**Console.WriteLine("Press ENTER to shut down service.");**

**Console.WriteLine();**

**Console.ReadLine();**

**// Close the service.**

**serviceHost.Close();**

**}**

**}**

**Output config**

The best way to diagnose WCF errors (the ones that really don't tell you much) is to enable tracing. In your web.config file,

<system.diagnostics>   
    <sources>   
      <source name="System.ServiceModel"    
              switchValue="Information"    
              propagateActivity="true">   
        <listeners>   
          <add name="ServiceModelTraceListener"    
               type="System.Diagnostics.XmlWriterTraceListener, System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"    
               initializeData="wcf-traces.svclog"/>   
        </listeners>   
      </source>   
    </sources>   
  </system.diagnostics>

public class Person   
    {   
        private int gID;   
        private String gFirstName="";   
        private String gLastName = "";

        public int ID   
        {   
            get      {     return gID;     }   
            set      {     gID = value;    }   
        }   
   
        public String FirstName   
        {   
            get       {   return gFirstName;   }   
            set       {   gFirstName= value;   }   
        }   
   
        public String LastName   
        {   
            get      {     return gLastName;     }   
            set      {      gLastName = value;    }   
        }

}

public class Persons   
    {   
        private List<Person> gListOfPerson;

        public List<Person> FxAll   
        {   
            get   
            {   
                if (gListOfPerson == null) gListOfPerson= new List<Person>();

    return gListOfPerson;   
            }   
            set   
            {   
                gListOfPerson=value;   
            }   
        }   
    }

public Persons ReturnData()   
{               
      DataSet  vDS = new DataSet();   
    
      foreach(System.Data.DataTable t in vDS.Tables)   
         {   
            Persons  vPersons = new Persons();   
            foreach(System.Data.DataRow dr in t.Rows)   
                 {   
                      Person vPerson = new Person();   
                      int vtryInt;   
                      int.TryParse(dr["ID"].ToString(), out vtryInt);   
                      vPerson.ID = vtryInt;   
                      vPerson.FirstName = dr["FirstName"].toString();   
                      vPerson.LastName = dr["LastName"].toString();   
   
                      vPersons.FxAll.Add(vPerson);   
                 } 

       return vPersons ;   
 }