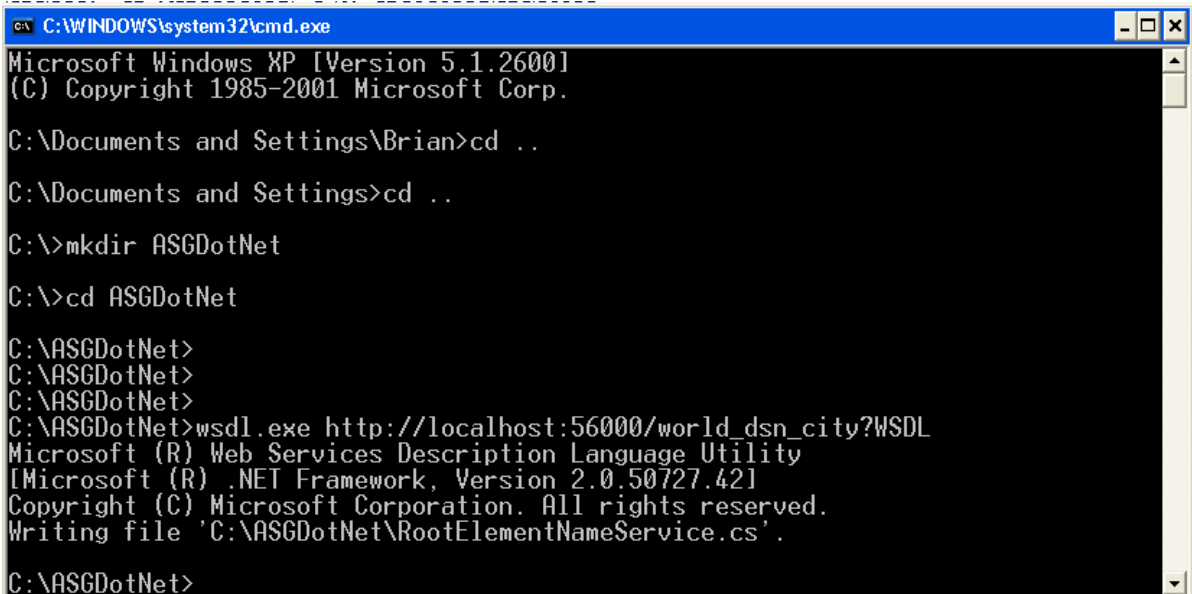


Using C# to access MySQL

This tutorial shows how to access the "City" table in MySQL from the C# environment. It assumes a C# environment is available, and some basic knowledge of the C# language.

With a service description (WSDL), a proxy class can be created with the .NET Framework SDK Wsd.exe tool. A XML Web service client can then invoke methods of the proxy class, which communicate with the SOA Gateway over the network by processing the SOAP messages sent to and from the SOA Gateway server. The proxy class handles the work of mapping parameters to XML elements and then sending the SOAP message over the network. Wsd.exe can be used to create proxies for C#, Visual Basic .NET and JScript .NET, for the purpose, we will be generating C#. These are the steps required to generate the C# wrapper class using Wsd.exe and create / run a program listing records from the City table using the generated proxy class

1. From a command prompt, execute Wsd.exe, specifying the URL / URI of the SOA Gateway Web Service to be exposed.



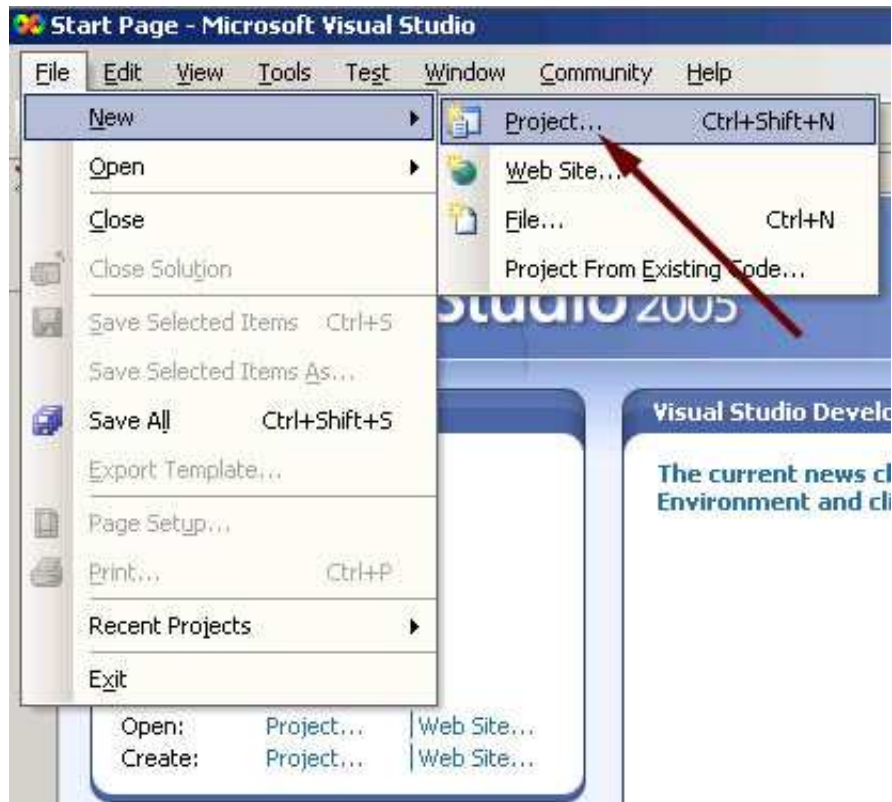
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Brian>cd ..
C:\Documents and Settings>cd ..
C:\>mkdir ASGDotNet
C:\>cd ASGDotNet
C:\ASGDotNet>
C:\ASGDotNet>
C:\ASGDotNet>
C:\ASGDotNet>wsdl.exe http://localhost:56000/world_dsn_city?WSDL
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 2.0.50727.421]
Copyright (C) Microsoft Corporation. All rights reserved.
Writing file 'C:\ASGDotNet\RootElementNameService.cs'.
C:\ASGDotNet>
```

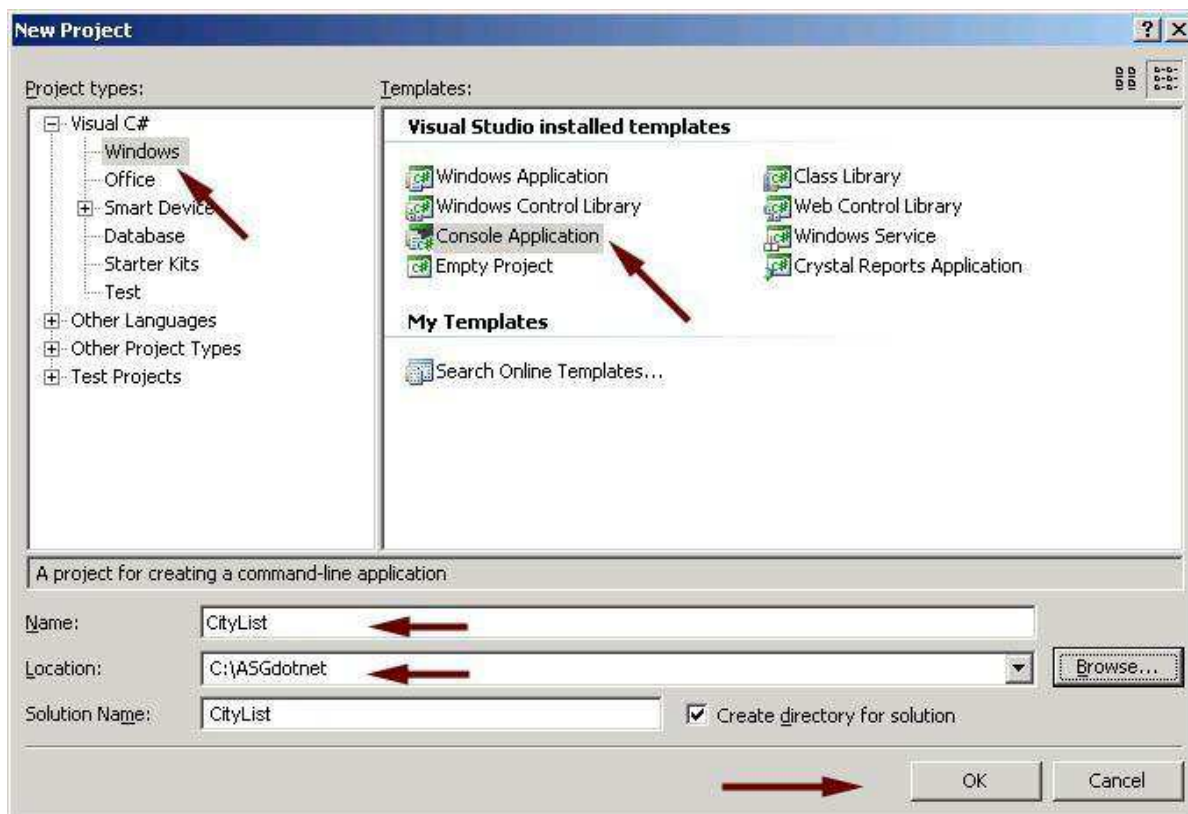
A single source file is generated, its name is <rootElementName>Service.cs, in this case the "root element" within the XRD is "RootElementName", thus the name of the proxy class source file RootElementNameService.cs

This file contains a proxy class exposing both synchronous and asynchronous methods for each Web Service operation provided by the SOA Gateway. For instance, for the list operation, the proxy class has the following methods: list, Beginlist, and Endlist. The list method of the proxy class is used to communicate with the SOA Gateway synchronously, but the Beginlist and Endlist methods are used to communicate with the SOA Gateway server asynchronously. For more information about asynchronous communication with a Web Service please refer to the .NET documentation.

2. Start MS Visual Studio, create a new project with File -> New - Project (or the shortcut Ctrl+Shift+N):



Create a C# Console Application, assign a name to it, specify the storage location, click **OK**



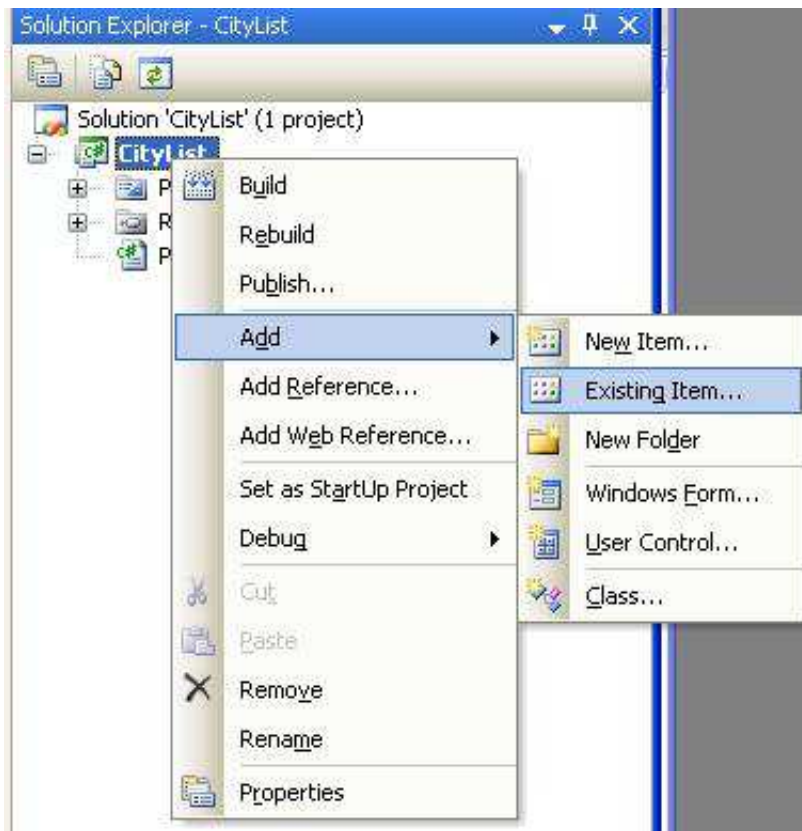
A skeleton class file has been generated into your project workspace, with the required class definition and an empty Main method

```
Program.cs
CityList.Program

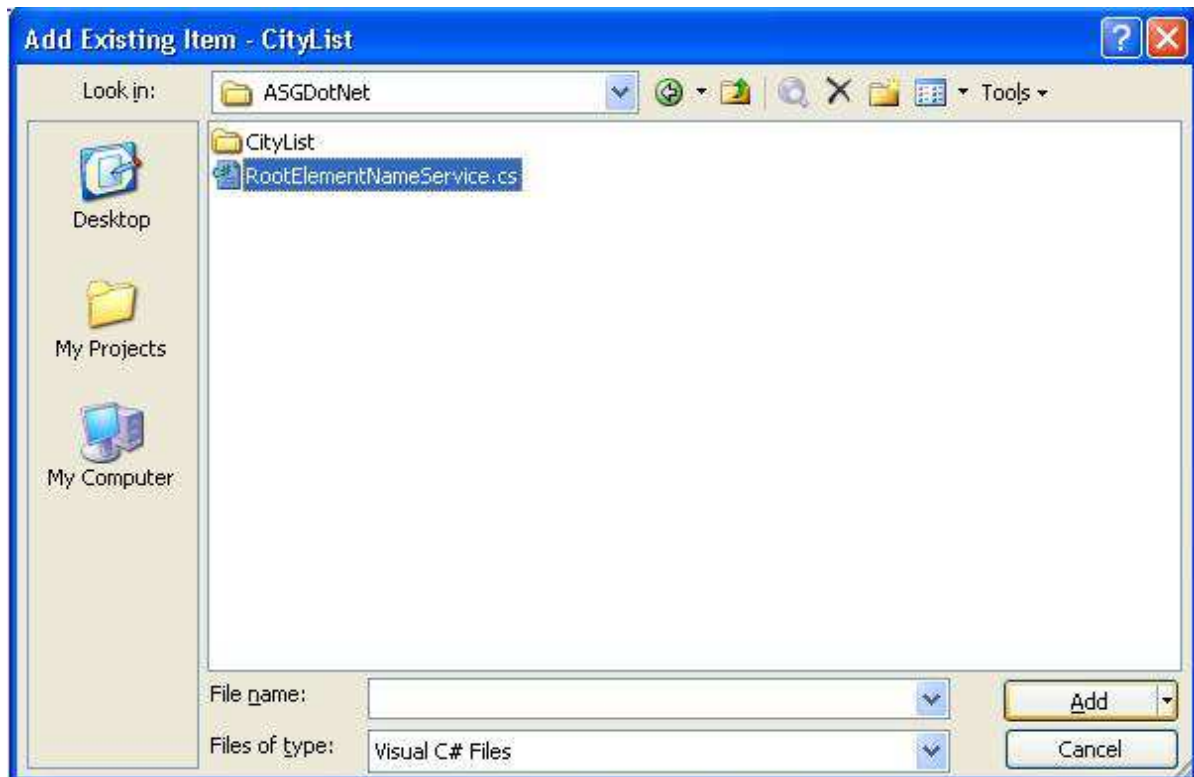
using System;
using System.Collections.Generic;
using System.Text;

namespace CityList
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

3. First of all, import the generated proxy into the project, right-click on the project name, select **Add Existing Item**

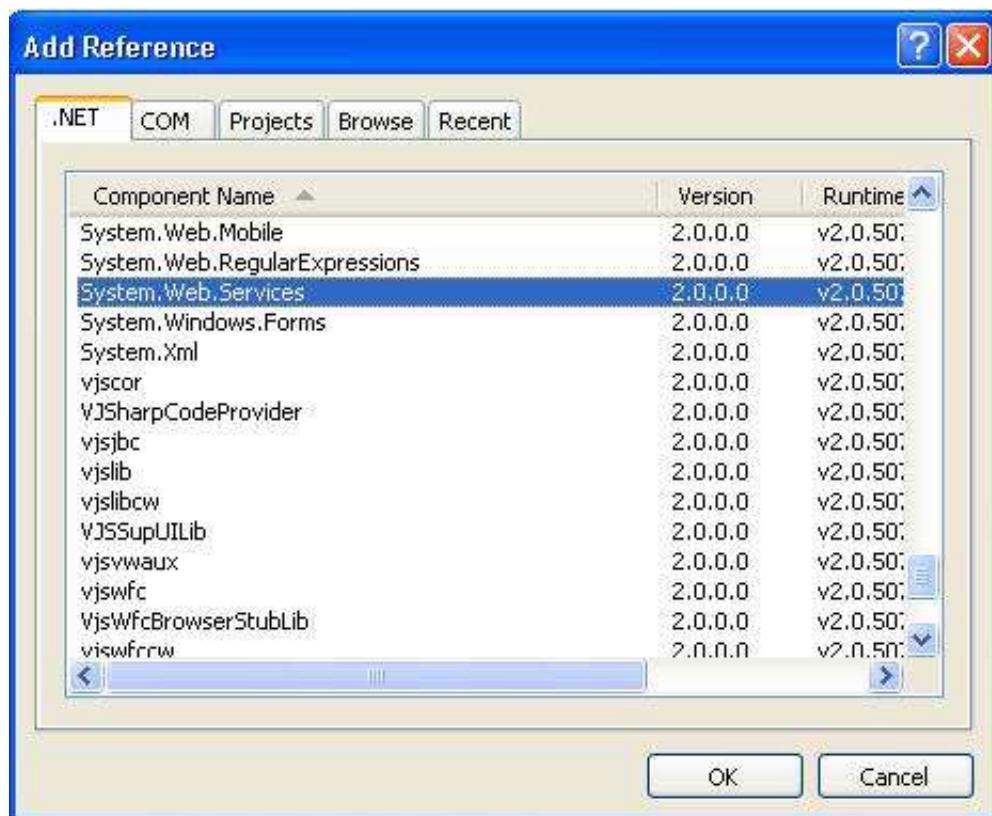


select the RootElementNameService.cs proxy file you created earlier, and click **Add**



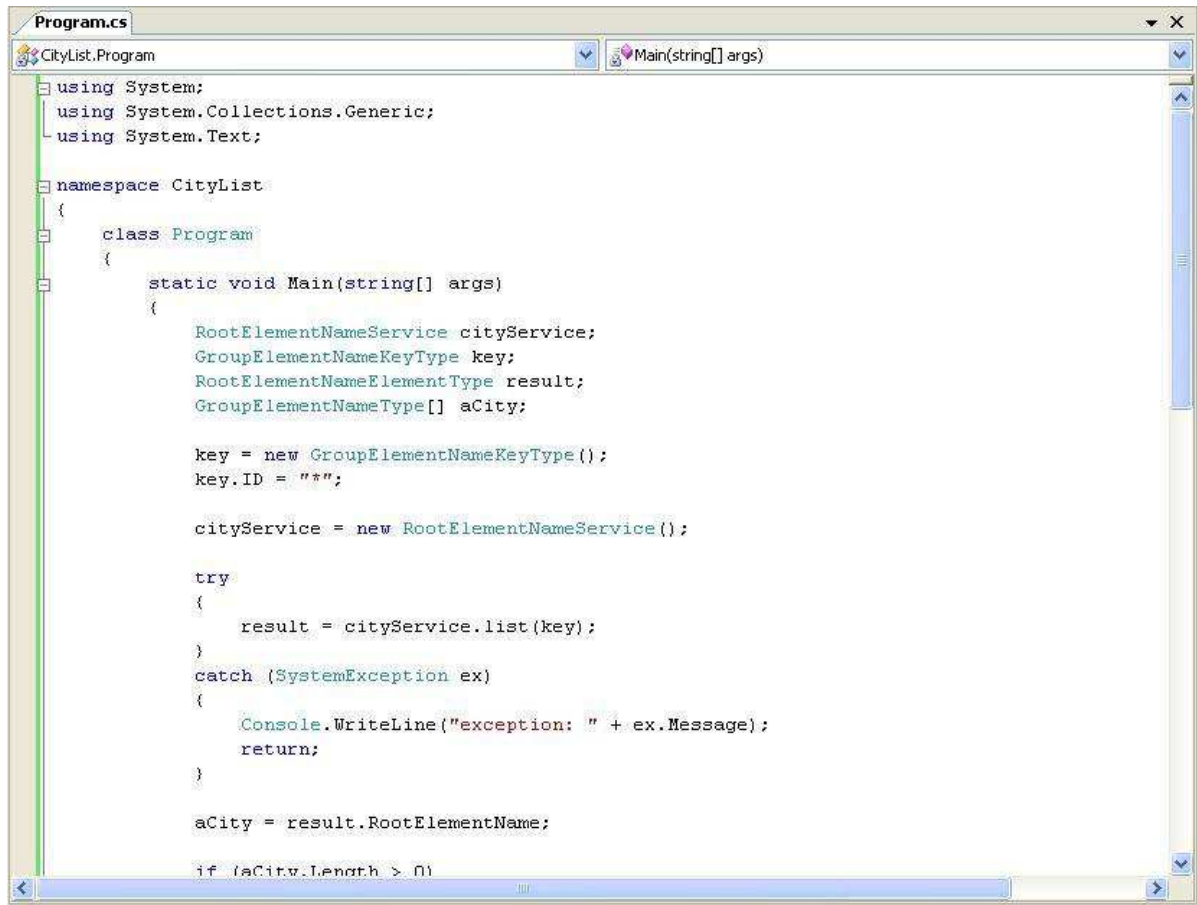
The proxy has been added to the project

You now need to add a reference to the .NET System.Web.Services component implementing the SOAP interface. In the project explorer, right click on the project name, select **Add Reference**



Select **System.Web.Services** and click **OK**

4. Remove the generated code from the newly added class entirely, use (paste) the code from MySQL_CityDemo.cs to create your first C# program accessing MySQL Web Service via the SOA Gateway.



```
Program.cs
CityList.Program
Main(string[] args)

using System;
using System.Collections.Generic;
using System.Text;

namespace CityList
{
    class Program
    {
        static void Main(string[] args)
        {
            RootElementNameService cityService;
            GroupElementNameKeyType key;
            RootElementNameElementType result;
            GroupElementNameType[] aCity;

            key = new GroupElementNameKeyType();
            key.ID = "";

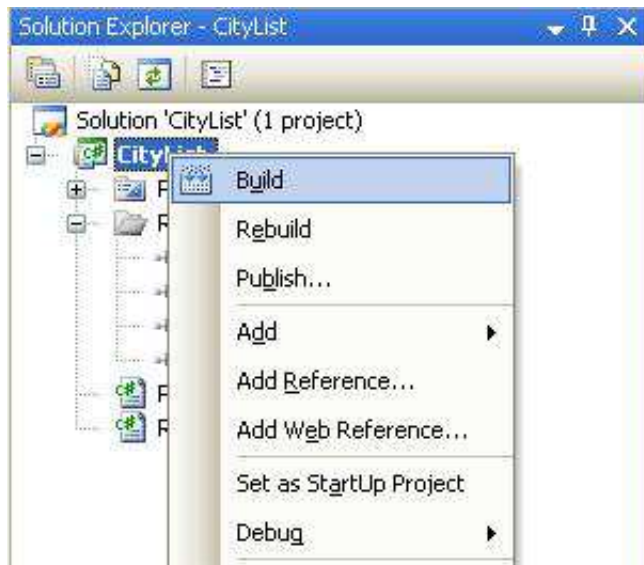
            cityService = new RootElementNameService();

            try
            {
                result = cityService.list(key);
            }
            catch (SystemException ex)
            {
                Console.WriteLine("exception: " + ex.Message);
                return;
            }

            aCity = result.RootElementName;

            if (aCity.Length > 0)
```

5. Build the application. Right-click on the project name in the project explorer, click **Build**



6. Open a command window, change to the project's build-directory Execute the compiled console application, CityList.exe, the output will look as follows:

