Creating a sample C# application

Tutorial: A sample C# application listing "Employees"

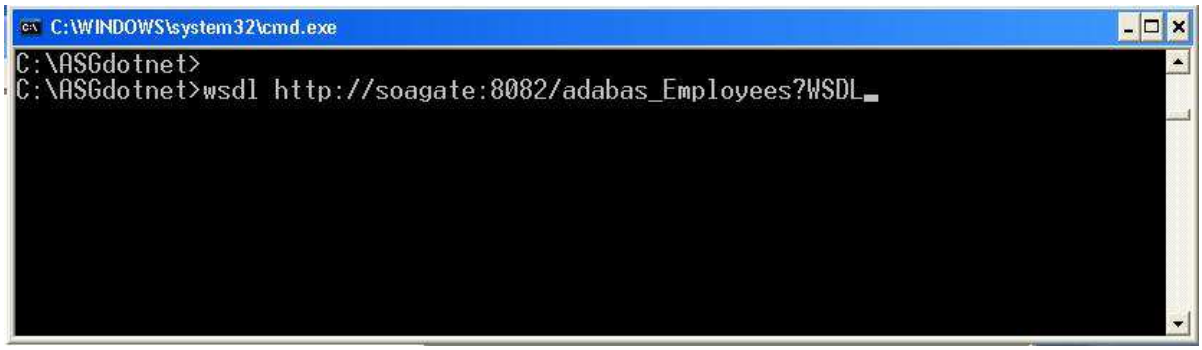
With a service description (WSDL), a proxy class can be created with the .NET Framework SDK Wsdl.exe tool. A XML Web service client can then invoke methods of the proxy class, which communicate with SOA Gateway over the network by processing the SOAP messages sent to and from the SOA Gateway server. The proxy class handles the work of mapping parameters to XML elements and then sending the SOAP message over the network.

Wsdl.exe is a Microsoft .NET tool which is used to create proxies for C#, Visual Basic .NET and JScript .NET. In this tutorial, we will be generating C#.

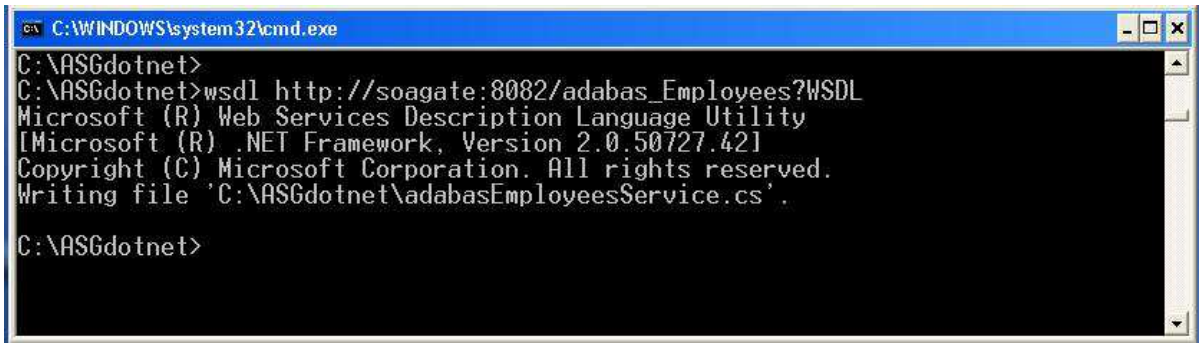
These are the steps required to generate the C# wrapper class using Wsdl.exe and create / run a program listing records from the Adabas demo file "Employees" using the generated proxy class:

1. From a command prompt, execute Wsdl.exe, specifying the URL / URI of the SOA Gateway DataSource to be exposed, append ?WSDL to instruct theSOA Gateway server to return the WSDL, not data:

   If the Wsdl.exe is not found, open the Visual Studio command prompt via the Start Menu. This location depends on what packages are installed, but often resides under "Microsoft Visual C# " or "Microsoft .NET Framework SDK ".

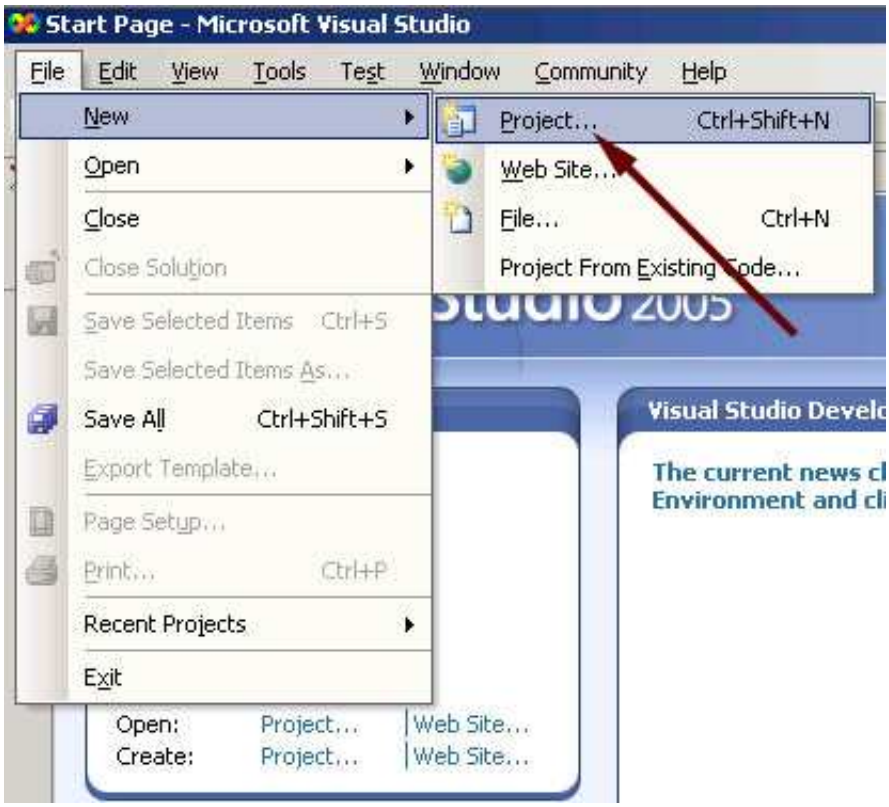
2. A single source file is generated, its name is <rootElementName>Service.cs, in this case the "root element" within the XSD is "adabasEmployees", thus the name of the proxy class source file adabasEmployeesService.cs



This file contains a proxy class exposing both synchronous and asynchronous methods for each SOAP operation provided by SOA Gateway for the DataSource. 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 SOA Gateway synchronously, but the Beginlist and Endlist methods are used to communicate with the SOA Gateway server asynchronously.

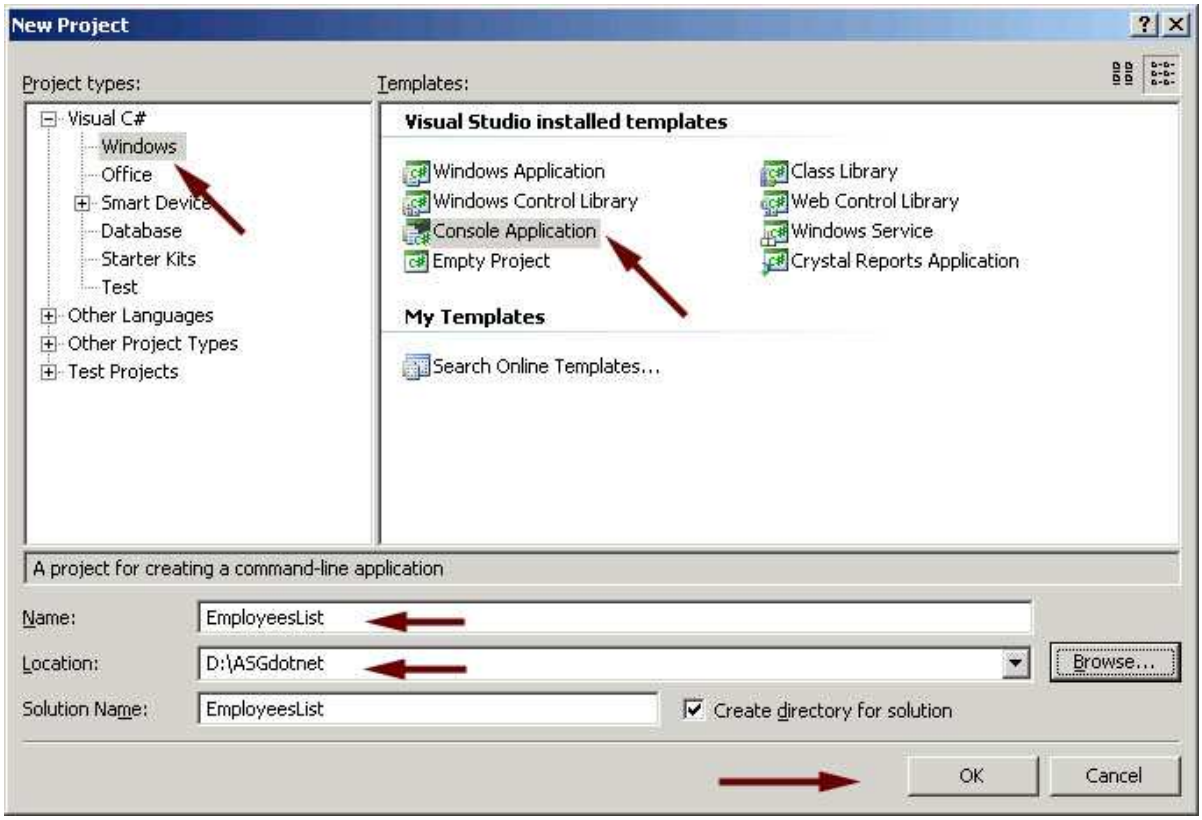
For more information about asynchronous communication with a Web Service please refer to the .NET documentation.

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

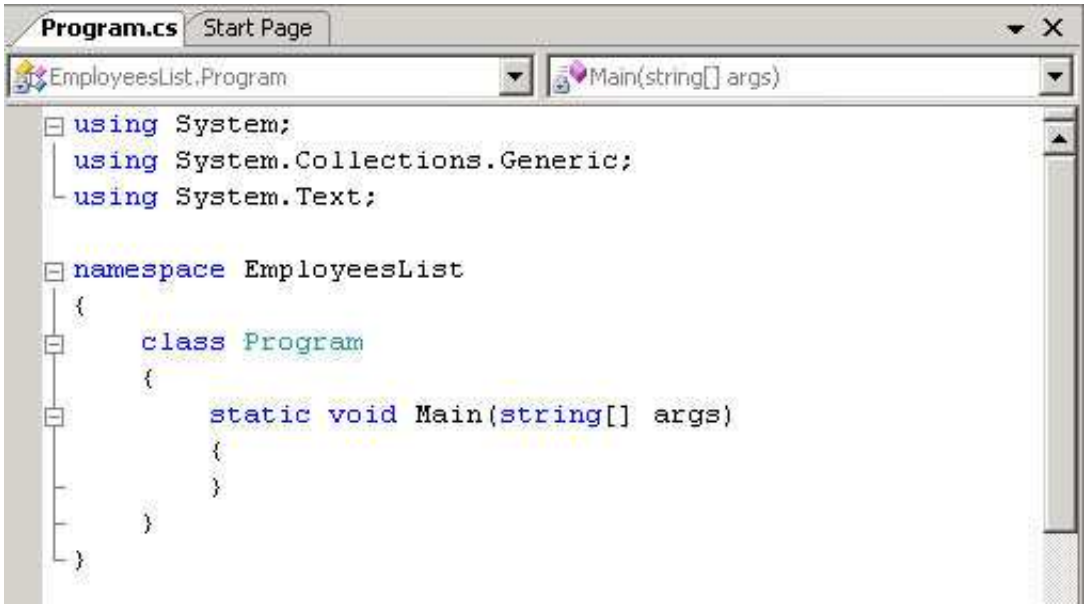


Create a C# Console Application, assign a name to it, specify the storage location, click **OK**

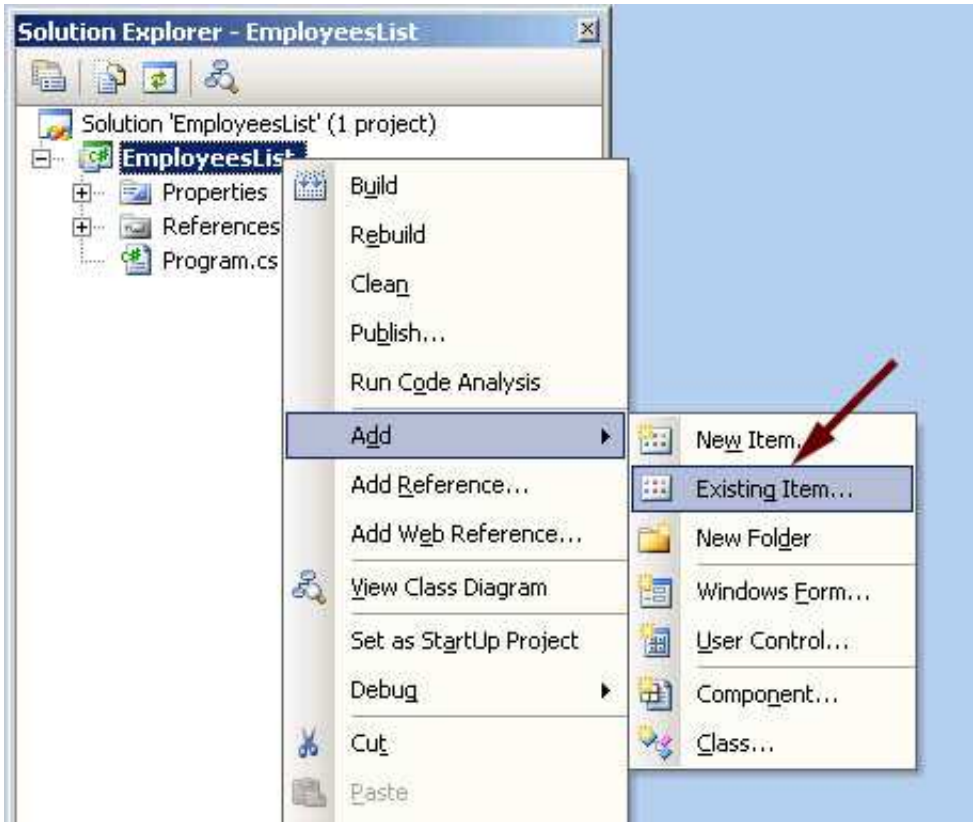
 Creating a sample C# application



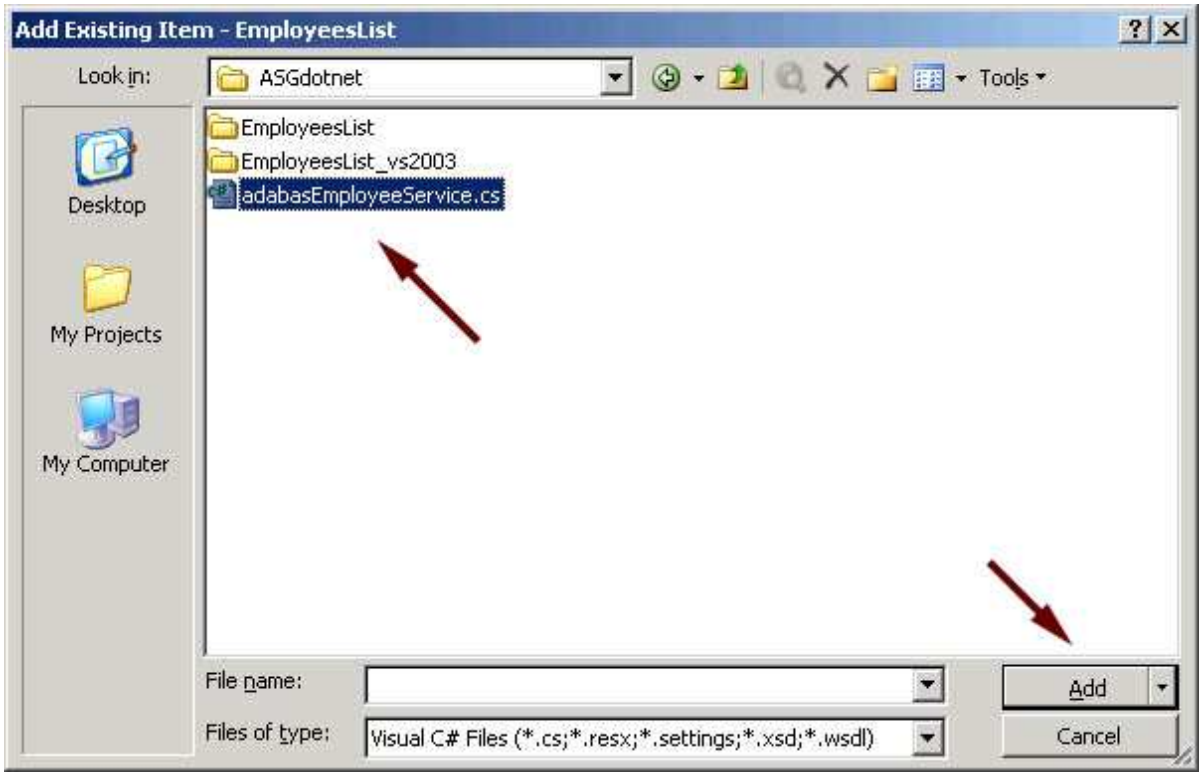
A skeleton class file has been generated into your project workspace, with the required class definition and an empty *Main* method



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

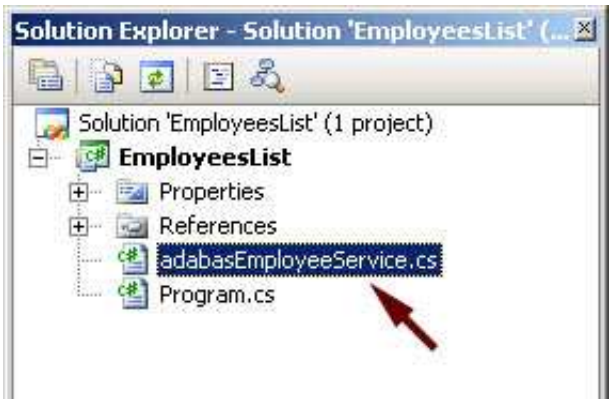


select the AdabasEmployeeService.cs proxy, click **Add**

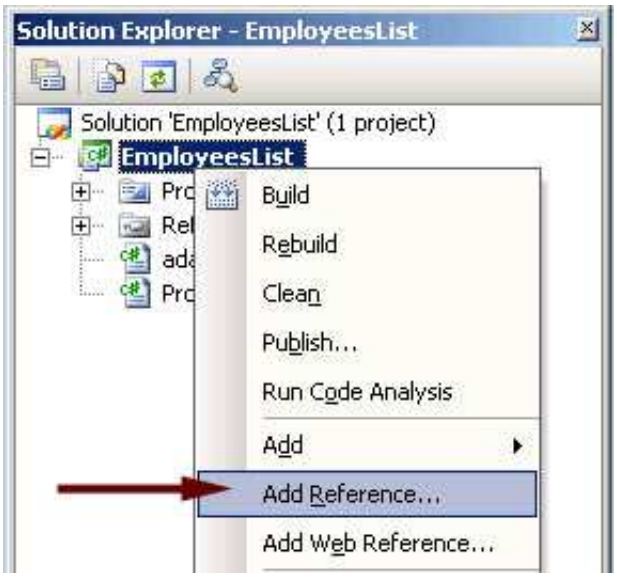


 Creating a sample C# application

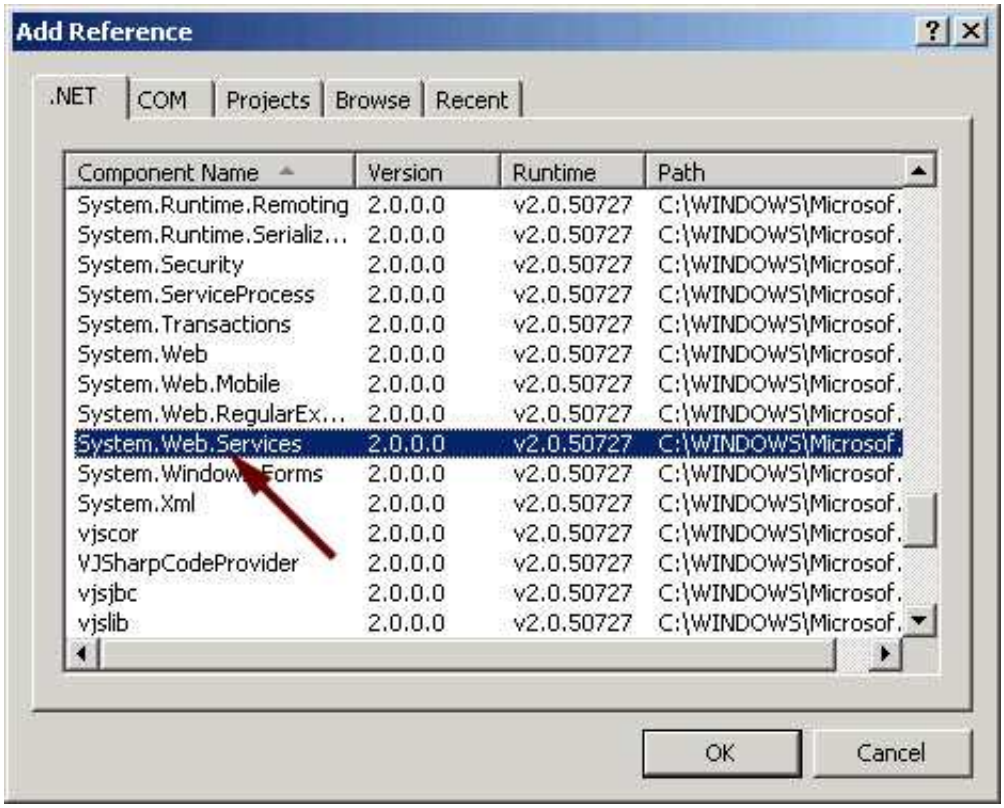
The proxy has been added to the project



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**



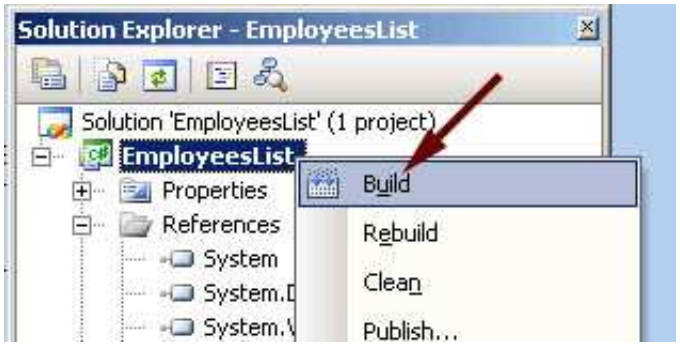
Scroll down to System.Web.Services, click to select it, click to select it, click **OK** to import the reference



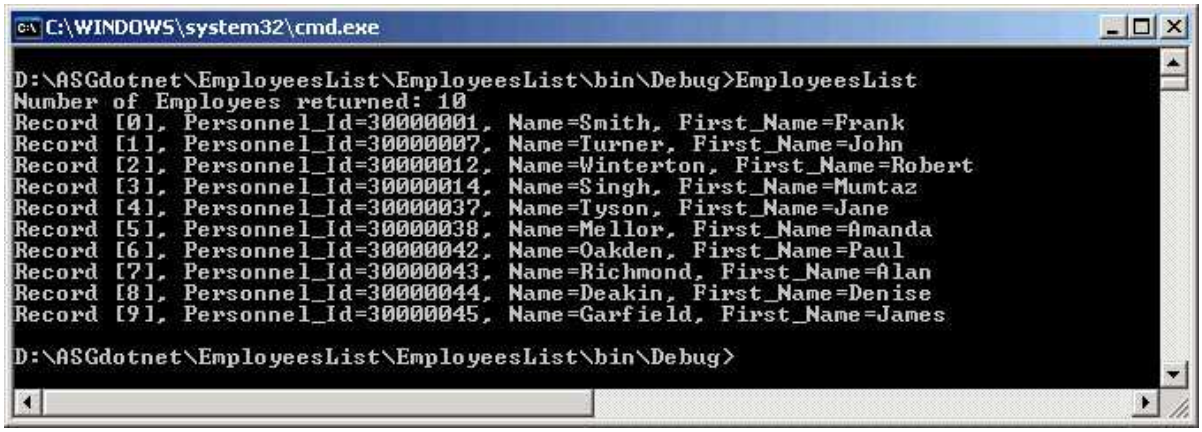
5. Remove the generated code from the newly added class entirely, use (paste) the code from ASGDemo.cs to create your first test program accessing Adabas data via SOA Gateway.

```
Program.cs
EmployeesList.Program                                   Main(string[] args)
using System;
using System.Collections.Generic;
using System.Text;

namespace EmployeesList
{
    class Program
    {
        static void Main(string[] args)
        {
            adabasEmployeesService emplService;
            adabasEmployeeKeyType keys;
            adabasEmployeesElementType result;
            adabasEmployeeType[] empl;

            keys = new adabasEmployeeKeyType();
            keys.personnel_id = "300000*";

            emplService = new adabasEmployeesService();

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

            empl = result.adabasEmployees;
```

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



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



8. This sample selects all "Employees" records with a personnel-id of 4000004n, you may want to experiment varying the key data, this is easily done by modifying the properties passed to the generated classes. E.g. try the following to list all records for "Employees" whose names start "SMI", living in cities with names starting "D".

```
keys.name = "SMI*";
keys.city = "D*;
```

The output will look like this: