

Accessing MySQL from C#

Contents

1. Introduction	2
2. Prerequisites	2
3. Setup	2
3.1. Populate MySQL Database.....	2
3.2. Set up ODBC Access	3
4. Discovery.....	4
4.1. Web Service Creation using SOA Gateway	4
4.2. Accessing the WSDL	6
5. Accessing Web Service with C#.....	8
5.1. Initial Setup	8
5.1.1. Setup for Visual C# 2005	8
5.1.2. Setup for Visual C# 2008	11
5.2. Designing the Form	14
5.3 Writing the Code.....	16
5.2.1. Global and Constructor Code.....	16
5.2.2. Event Handling Code: Search Button	17
5.2.3. Code for VC# 2005	17
5.2.4. Code for VC# 2008	18
5.2.5. Event Handling Code: Account Details Button.....	19
5.2.6. Code for VC# 2005	20
5.2.7. Code for VC# 2008	20
5.4 Building the Code.....	21
5.5 Running the code	22
6. Conclusion.....	22
7. Appendix	22
Form1.cs for VC# 2005.....	22
Form1.cs for VC# 2008.....	22

1. Introduction

In this tutorial we will show you how to build a C# application to access MySQL via the SOA Gateway.

2. Prerequisites

It is assumed that you are running the 3 components, MySQL, C# and the SOA Gateway on Windows.

It is assumed you already have a SOA Gateway server and Control Centre installed. See [here](#) for more info about installing the SOA Gateway.

3. Setup

To build and run C# applications, you will need a Visual Studio IDE. If you do not already have a C# IDE installed, we recommend using the *Microsoft Visual Studio Express* range of products. They can be downloaded freely from Microsoft website, packaged for a number of languages, including C#. See [here](#) for more information about downloading, installing, and configuring *VC# Express*.

You will also need a MySQL database. Again, the Open Source version (known as the *MySQL Community Server*) can be freely downloaded from the MySQL website. See [this link](#) for download, and [here](#) to step you through the installation and configuration.

3.1. Populate MySQL Database

Now that you've got MySQL installed and configured, you will need to populate it with some demo data. For this we use the Risaribank sample. This is available [here](#).

Save this file to "C:\Temp\Risaribank.sql"

- Connect to the MySQL Server using the **mysql** command.

```
E.g shell> mysql -u root -p
```

This command connects to the server using the MySQL `root` account to make sure that you'll have permission to create the `Risaribank` database. The `-p` option tells **mysql** to prompt you for the `root` password. Enter the password when prompted. (Remember that the MySQL `root` account is not the same as the operating system `root` account and probably will have a different password.)

- Create the `Risaribank` database.

```
mysql> CREATE DATABASE Risaribank;
```

```
mysql> use Risaribank;
```

- Load the contents of `Risaribank.sql` into the `Risaribank` database. E.g.

```
mysql> SOURCE c:\Temp\Risaribank.sql
```

- After the `SOURCE` command finishes, you can view your new tables.

```
mysql> SHOW TABLES;
```

```
mysql> DESCRIBE CustomerInformation;
```

```
mysql> DESCRIBE Branch;
```

etc ...

3.2.Set up ODBC Access

The final thing to do with your MySQL Database is to set up an ODBC DSN which will be used by the SOA Gateway to access this database.

Click Start, Control Panel, Administrative Tools, Data Sources (ODBC)

From the resulting screen, choose the "System DSN" Tab.

Click Add

From the list of data source drivers, select "MySQL ODBC 3.51 Driver".

If you do not see this driver in the list, you need to install the MySQL Connector. See [here](#) for more information. We recommend installing v3.51.

Click Finish, and a window will appear allowing you to enter the DSN information. Add the following:

Data Source Name: RisarisBank

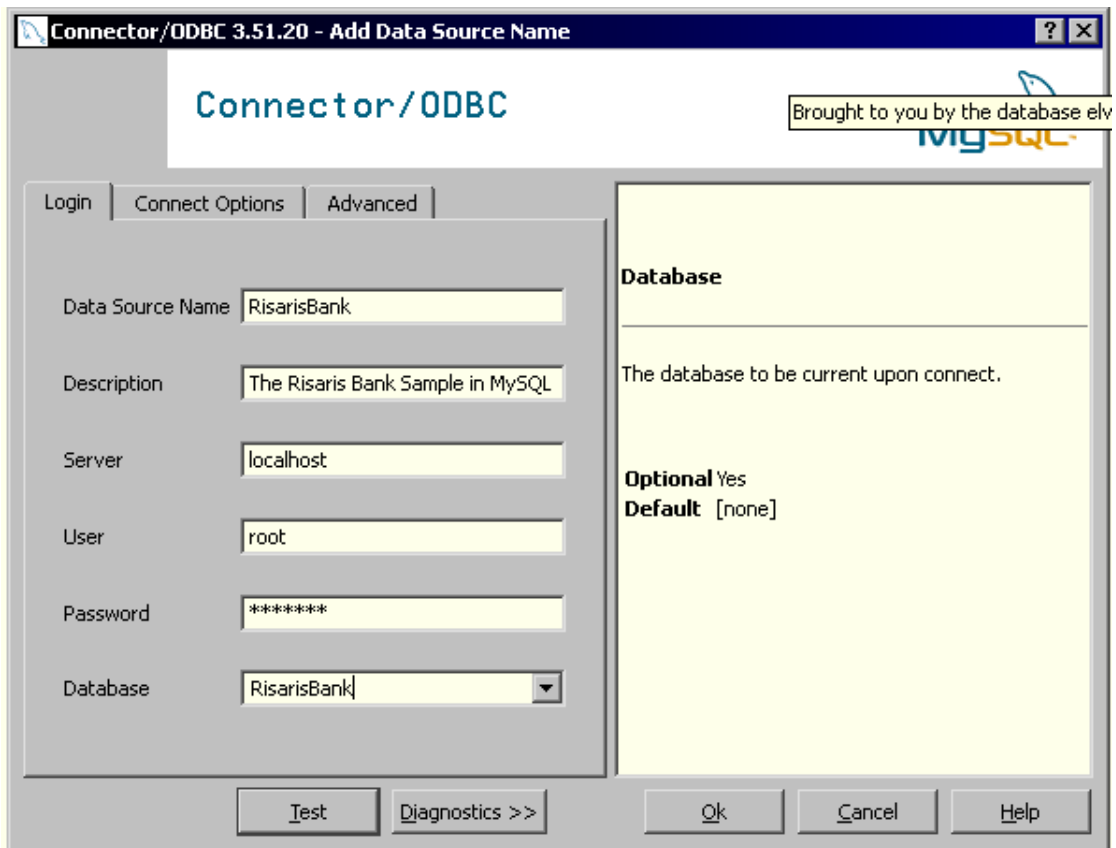
Description: The Risaris Bank Sample in MySQL

Server: localhost

User: root

Password: *** your MySQL root password ***

Database: RisarisBank (select from the drop down list)



All other options can be left as-is. Click OK.

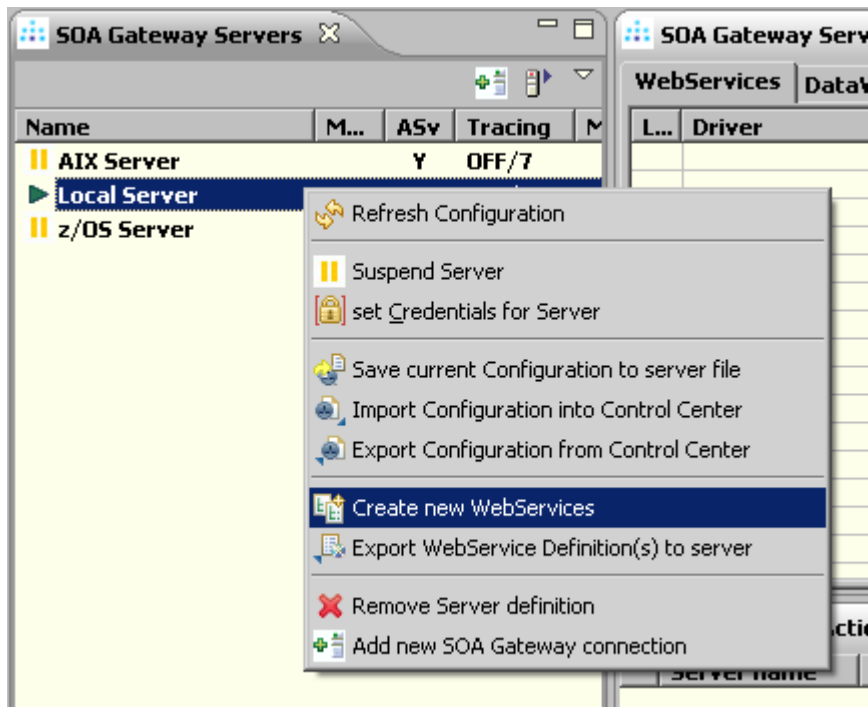
4. Discovery

At this stage you've got a C# IDE, and a MySQL database with some sample data in it. In this section we'll show you how to create web services from each of the MySQL tables. These web services can be used by the C# language (and many others) to give you direct real-time access to your MySQL Data.

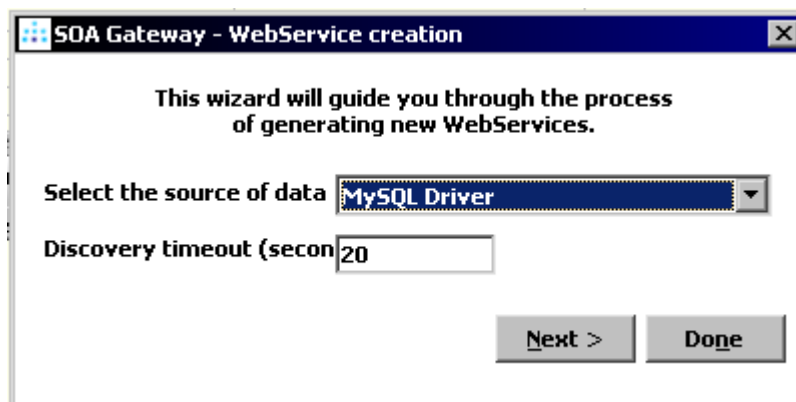
4.1. Web Service Creation using SOA Gateway

Start your SOA Gateway Control Centre. See [here](#) for an introduction to the Control Centre.

In your servers view, right click the entry which represents your local SOA Gateway Server. Select "Create New Web Services".

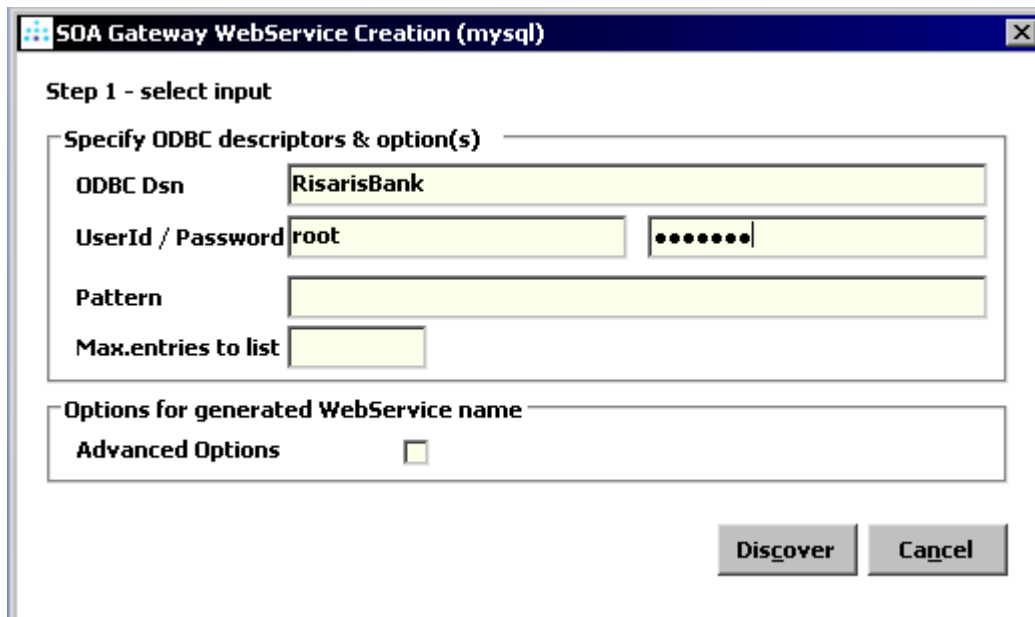


From the next dialog, choose “MySQL Driver”. If you do not see have a MySQL Driver in the list, see how to create one [here](#).



Click Next.

The next screen gives you the ability to add information about your DSN



Enter the above information and click Discover.

The wizard will display all the tables it finds at this (RisarisBank) DSN.

Click "Select All", and click "Import".

The wizard will create web services from each one of these tables.

SOA Gateway Servers		SOA Gateway Server Configuration - Local Server				
		WebServices / XSDs / XSLs				
Name	M... ASv	Mod	Driver	WebService	DataSource Id	DataView
AIX Server	Y					
DMZ	Y					
dublin dev	Y					
jk server	Y					
jk server linux	Y					
jom server	Y					
Local Server	Y					
lxbre server	Y					
PCRJW9	Y					
risaris.com server	Y					
vse	Y					
z/OS Server	Y					
z/vse	Y					
		MySQL	MySQL Driver	accountsmovements	odbcDsn=RisarisBank, tableName=accountsmovements	accountsmovements
		MySQL	MySQL Driver	audit	odbcDsn=RisarisBank, tableName=audit	audit
		MySQL	MySQL Driver	branch	odbcDsn=RisarisBank, tableName=branch	branch
		MySQL	MySQL Driver	currentaccount	odbcDsn=RisarisBank, tableName=currentaccount	currentaccount
		MySQL	MySQL Driver	customeraccountxref	odbcDsn=RisarisBank, tableName=customeraccountxref	customeraccountxref
		MySQL	MySQL Driver	customerinformation	odbcDsn=RisarisBank, tableName=customerinformation	customerinformation
		MySQL	MySQL Driver	depositaccount	odbcDsn=RisarisBank, tableName=depositaccount	depositaccount
		MySQL	MySQL Driver	tellertable	odbcDsn=RisarisBank, tableName=tellertable	tellertable

You've just created 8 Web Services from your 8 MySQL Tables!

4.2. Accessing the WSDL

Web Service Description Language (WSDL) is a standard, XML-based language that is used to describe a Web Service.

For each of the 8 web services you've created in the previous section, the SOA Gateway provides you with a WSDL to describe the Web Service. The WSDL itself is usually interpreted by a web

service client, such as C#, but it is useful to know where to find the WSDL for each of your Web Services.

As WSDL is XML-based, it will open in your browser of choice. To see the WSDL for one of your Risaris Bank web services, do the following in your SOA Gateway Control Centre:

- Click on the web service you are interested in, for example the “branch” web service.
- The properties for this web service should appear in your [Properties View](#). If you do not see the Properties view, select Window -> Show View -> Other -> General -> Properties and click OK.
- In the properties view, there is a link to your WSDL. Click it to open the WSDL in a browser.

The screenshot displays the SOA Gateway Server Configuration interface. The top window, titled "SOA Gateway Server Configuration - Local Server", has a "WebServices" tab selected. It contains a table with the following data:

Mod	Driver	WebService	DataSource Id
	MySQL Driver	accountsmovements	odbcDsn=RisarisBank, tableN
	MySQL Driver	audit	odbcDsn=RisarisBank, tableN
	MySQL Driver	branch	odbcDsn=RisarisBank, tableN
	MySQL Driver	currentaccount	odbcDsn=RisarisBank, tableN
	MySQL Driver	customeraccountxref	odbcDsn=RisarisBank, tableN
	MySQL Driver	customerinformation	odbcDsn=RisarisBank, tableN
	MySQL Driver	depositaccount	odbcDsn=RisarisBank, tableN
	MySQL Driver	tellertable	odbcDsn=RisarisBank, tableN

Below this table is the "SOA Gateway Action Log" window, which shows the following messages:

- Local Server: ODBC discovery completed, 8 WebService(s) generated
- Local Server: Configuration autosaved due to published WebService modification(s)

The bottom window, titled "Properties", shows the "WebService properties" for the "branch" service. A green arrow points to the "WSDL URL" field, which contains the text: <http://localhost:56000/branch?WSDL>. Below this, the "WebService Identification and options" section shows the following fields:

- odbcDsn: RisarisBank
- schemaName: (empty)
- tableName: branch

You can view the WSDL for the other web services by clicking the link from their properties view.

This WSDL is the starting point for using Web Services, and can be used time and again by different web service clients.

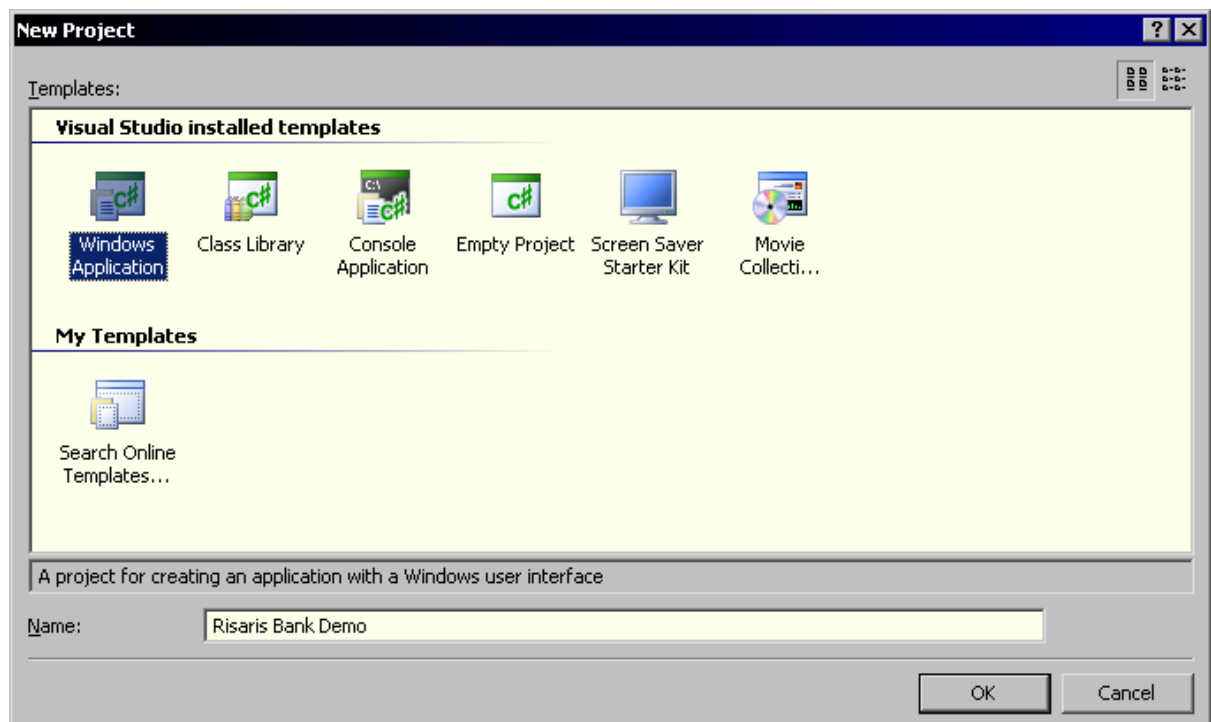
5. Accessing Web Service with C#

C# is an object orientated programming language from Microsoft as part of the .NET initiative. Its syntax is loosely based on C++, while borrowing other aspects from programming languages such as Java.

We will use C# to build an application which accesses our new Risar Bank Web Services via the WSDL.

5.1. Initial Setup

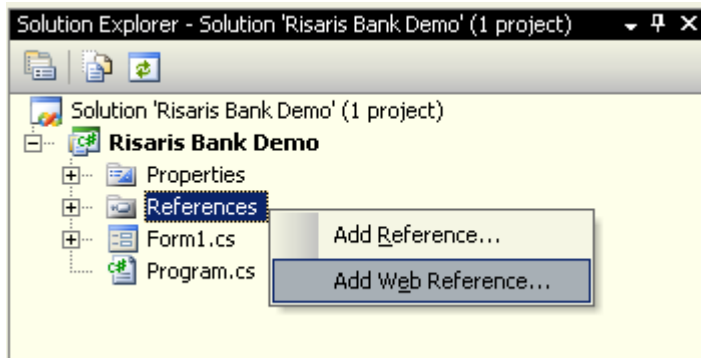
Start *Microsoft Visual C# Express* and create a New Windows Application Project named **Risar Bank Demo**.



Depending on which version of Visual Studio you are using, choose one of the following sections.

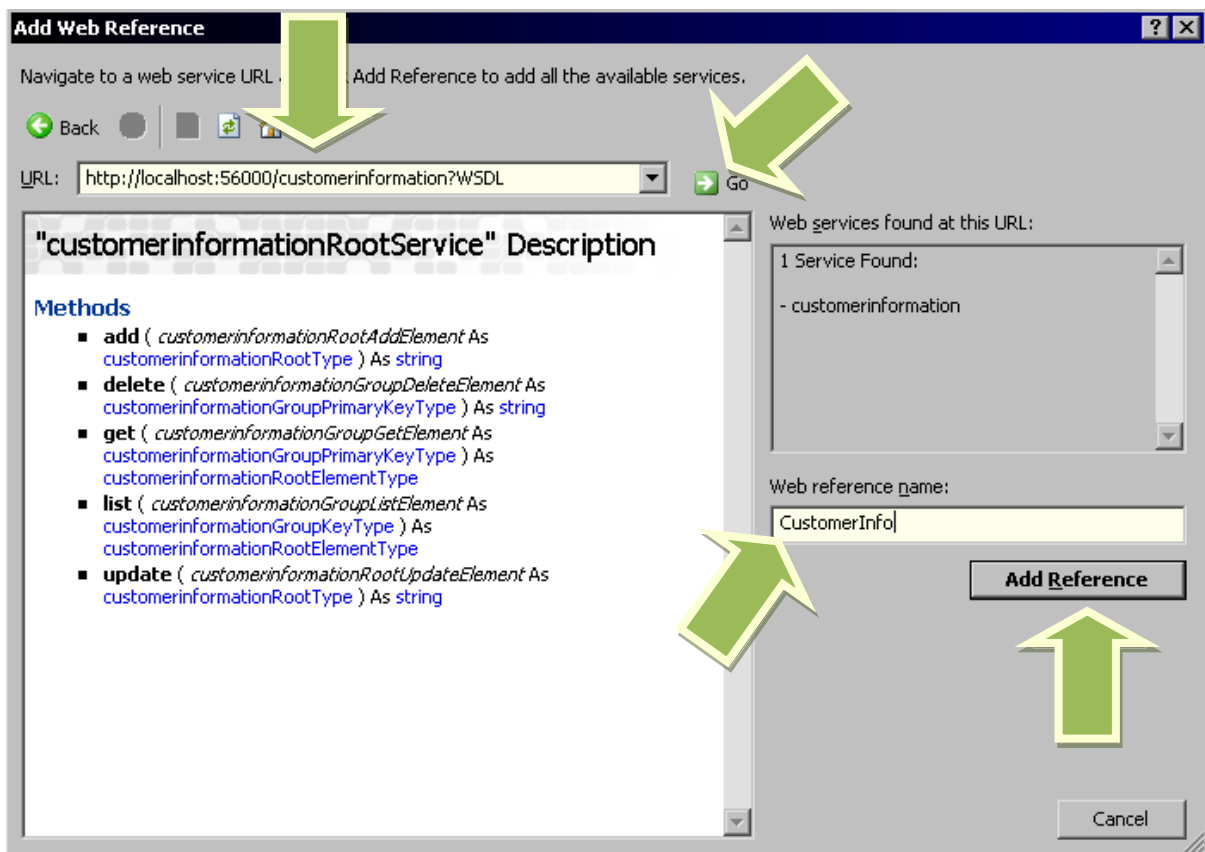
5.1.1. Setup for Visual C# 2005

In the *Solution Explorer*, right click **References**, and select **Add Web Reference**

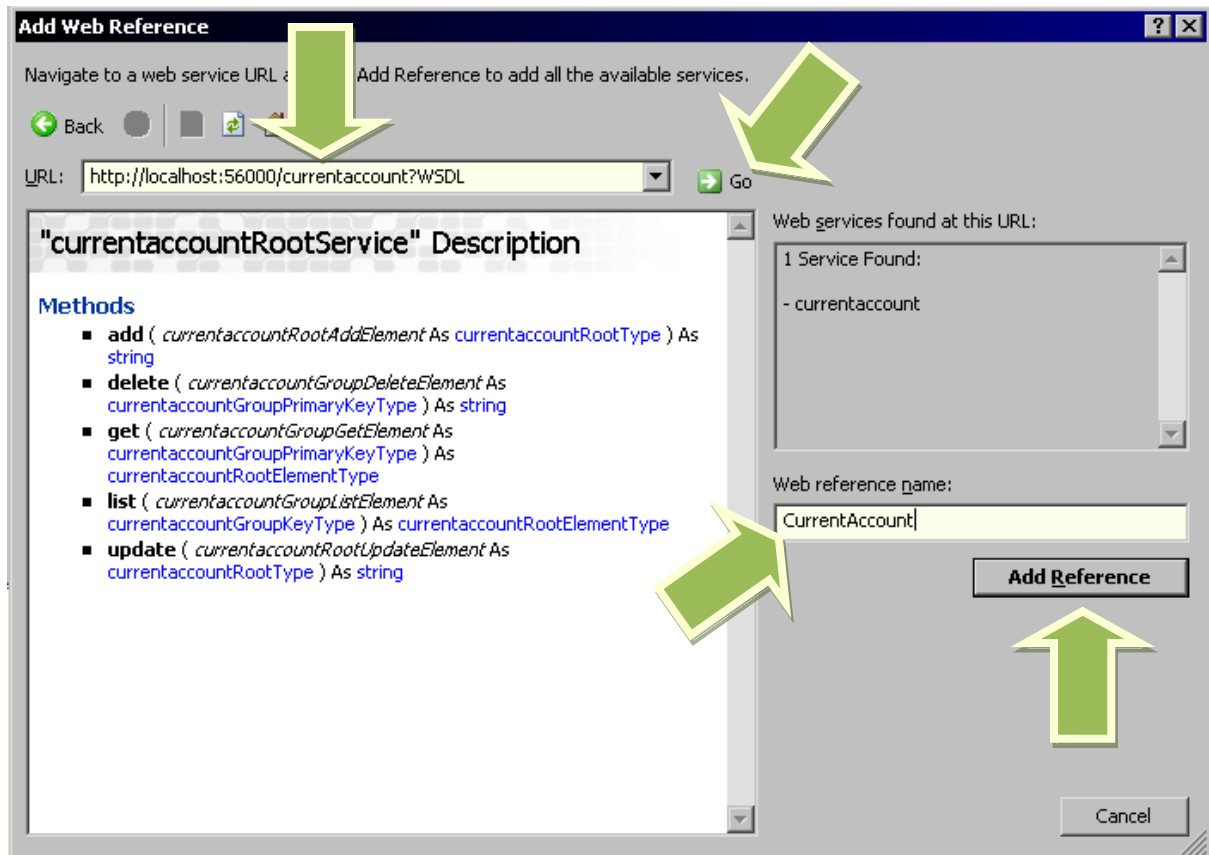


We want to use 2 of the Web Services we've created, customerinformation and the currentaccount web services.

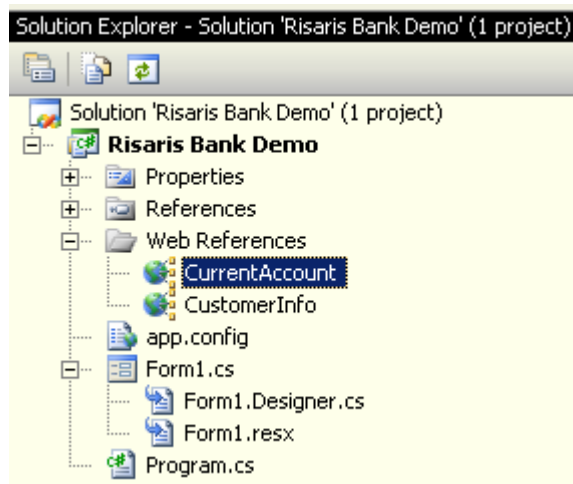
- Copy the URL of this web services WSDL into the URL box.
- Click Go.
- Once the WSDL has been loaded, change the Web Service Reference Name to **CustomerInfo**
- Click **Add Reference**



Do the same for the currentaccount WSDL, except change the Web Reference name to **CurrentAccount**

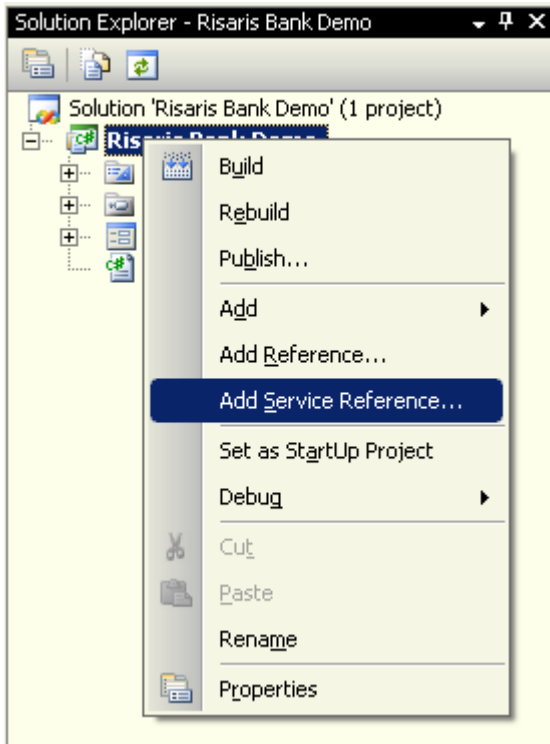


You should now have 2 new Web References loaded into your Solution Explorer



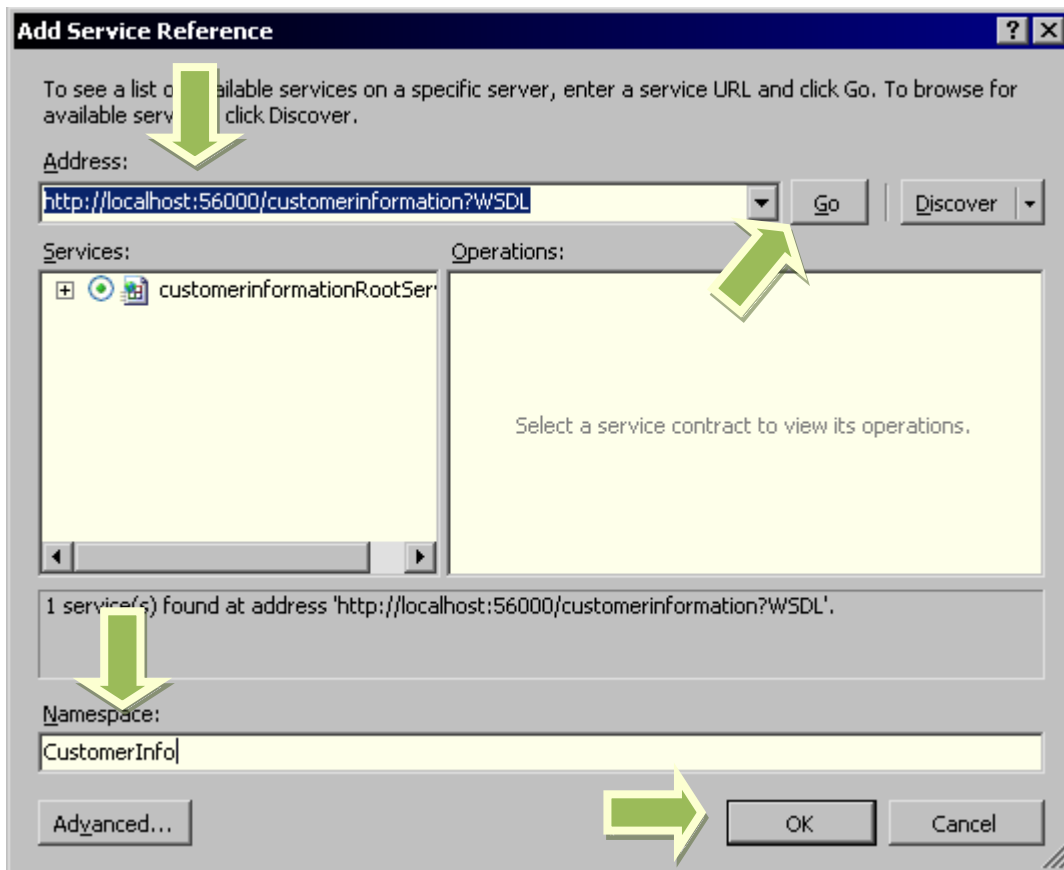
5.1.2. Setup for Visual C# 2008

Right-click your project and select “Add Service Reference”

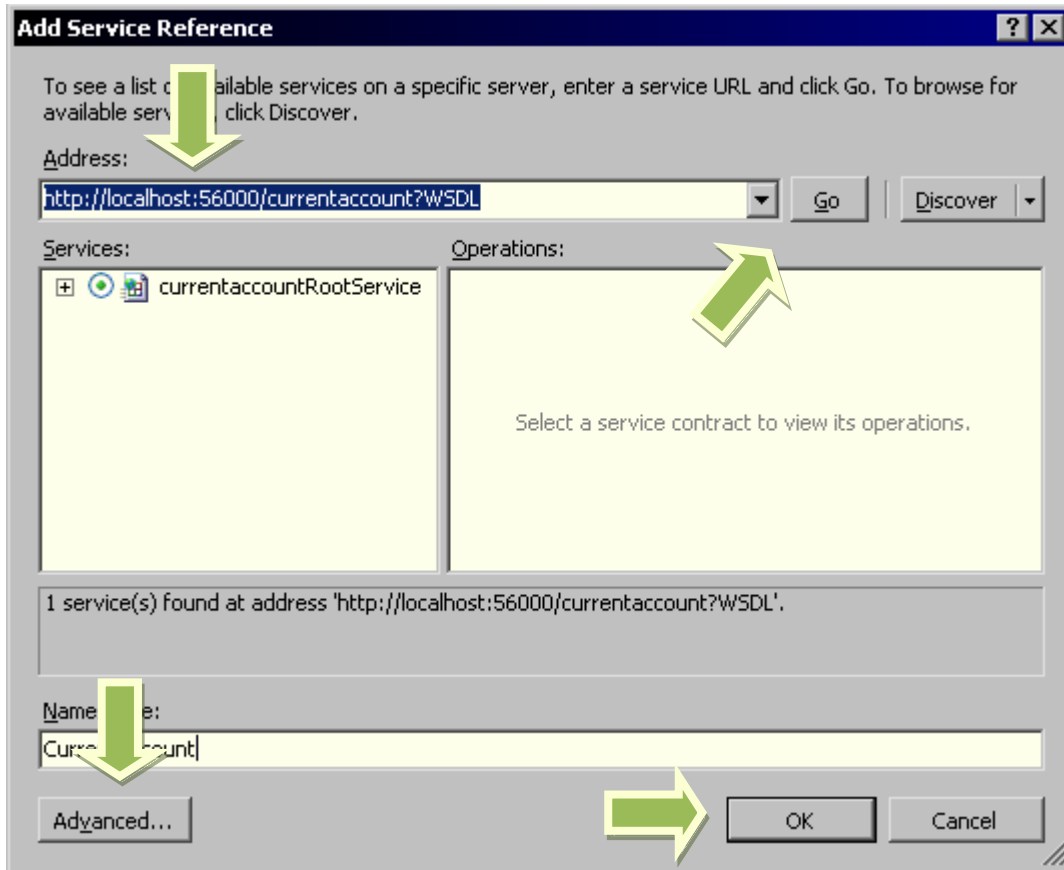


We want to use 2 of the Web Services we've created, CUSTOMERINFORMATION_SYSTEM and the CURRENTACCOUNT_SYSTEM web services.

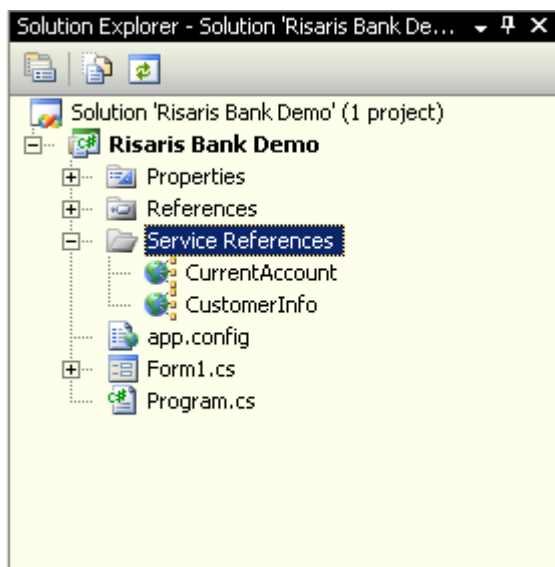
- Copy the URL of this web services WSDL into the Address box.
- Click Go.
- Once the WSDL has been loaded, change the Namespace to **CustomerInfo**
- Click **OK**



Now do the same for the CURRENTACCOUNT_SYSTEM WSDL, except change the Web Reference name to **CurrentAccount**

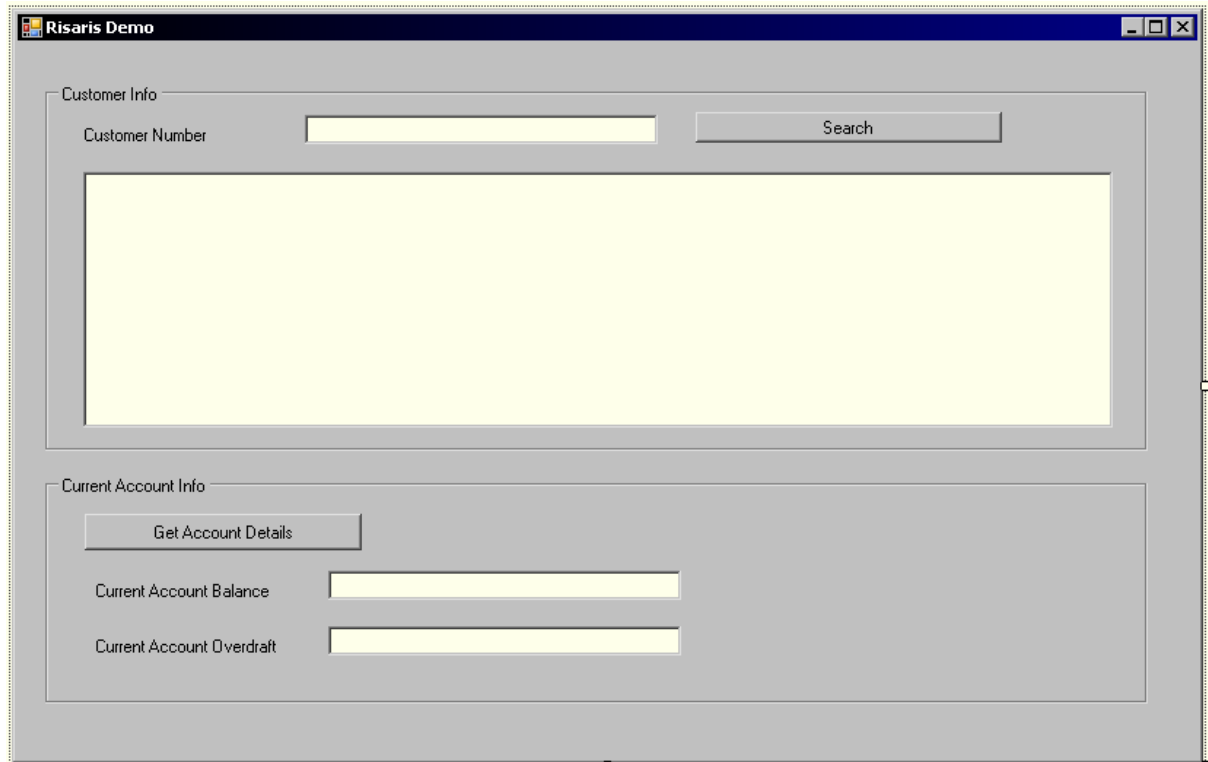


You should now have 2 new Service References loaded into your Solution Explorer



5.2.Designing the Form

In this section we'll add the necessary controls to our Form. Before you start ensure, that you are in the Designer View (View -> Designer), and that you have the control Toolbox available (View->Toolbox)



I've used the following controls in this Form

- GroupBox (Customer Info and Current Account Info)
- Label (Customer Number, Current Account Balance, Current Account Overdraft)
- TextBox (at each label)
- Button (x2)
- ListView

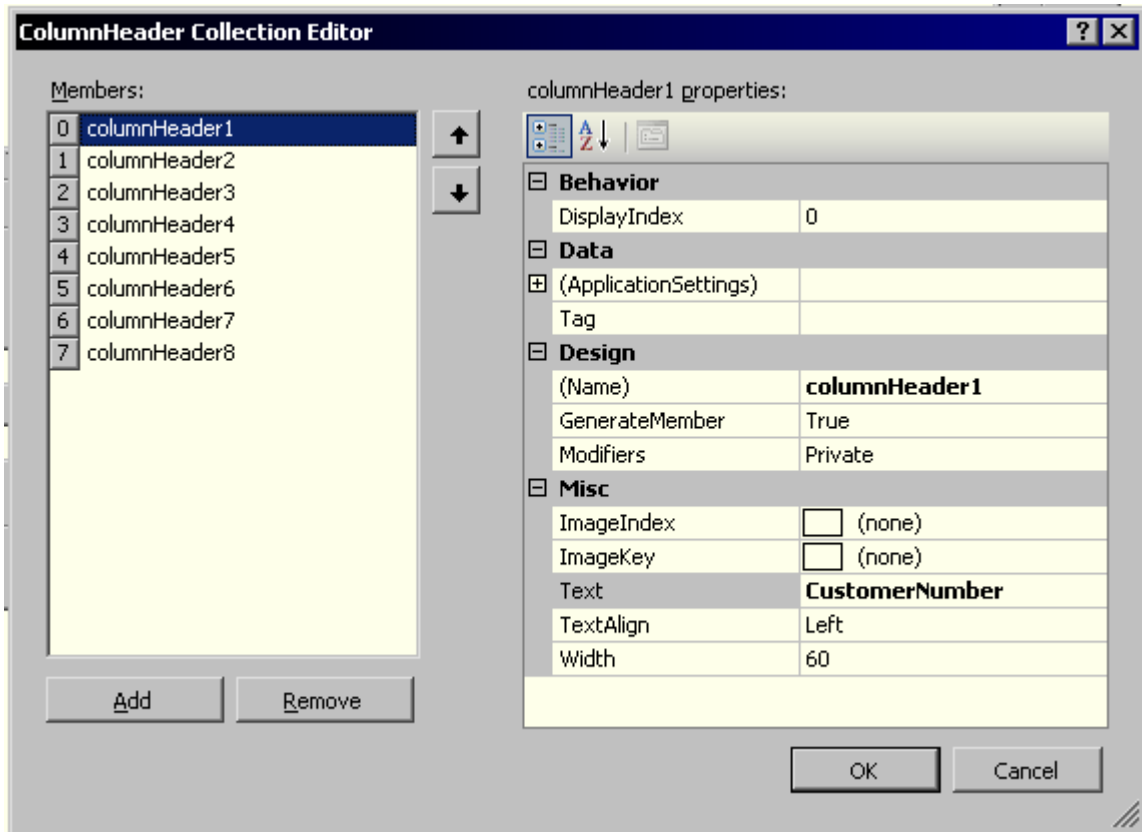
The ListView is the only control that needs additional setup.

When you add this control, edit the "Columns" property. Add 8 members, each with the following Text property.

- ✓ CustomerNumber
- ✓ FirstName
- ✓ Surname
- ✓ AddressLine1
- ✓ AddressLine2
- ✓ City
- ✓ ZipCode

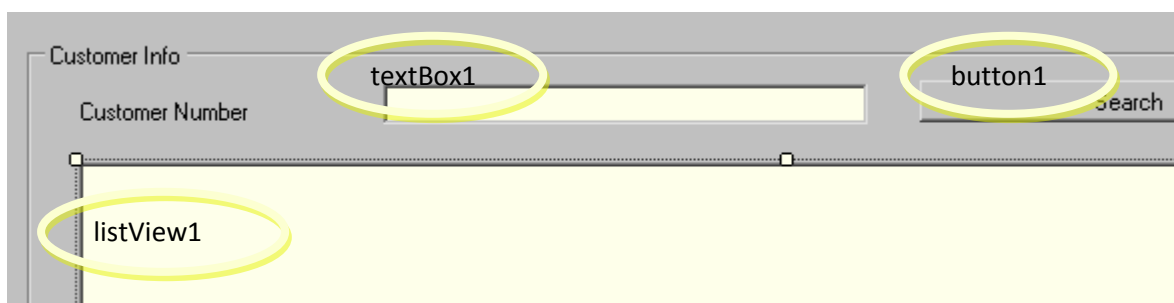
✓ DOB

You will notice that these fields match the columns that appear on the CustomerInformation table in MySQL.



Note that I haven't changed any of the default design names that the C# designer has given me. Therefore from the top of the form the design names of the text boxes are textBox1, textBox2, textBox3. Similarly, the buttons are named button1 and button2, and the list view is called listView1.

You may change these to what ever you wish, but be aware your C# code in the next section will have to be cognisant of this!



5.3 Writing the Code

5.2.1. Global and Constructor Code

Now that the Form controls have been added, we need to write the code to call our Web Services when the buttons are clicked.

Switch to your Code view, by clicking View->Code

After the “using” statements, enter the following

```
using System.Globalization;
using Risaribank_Demo.CurrentAccount;
using Risaribank_Demo.CustomerInfo;
```

These statements include your 2 Web References you added earlier.

In the “Form” class, declare the security objects you need to set to access your web services.

```
namespace Risaribank_Demo{
    public partial class Form1 : Form{

        CustomerInfo.Security customerInfoSecurity;
        CurrentAccount.Security currentAccountSecurity;

        . . .
```

In the Form1 constructor, create new instances of these objects, and set them to the correct values. In the example below, we use the username + password combination of root and letmein. These are the credentials needed to access my MySQL database, so in your situation the password will probably need to be changed.

```
public Form1(){

    InitializeComponent();

    customerInfoSecurity = new Risaribank_Demo.CustomerInfo.Security();
    customerInfoSecurity.UsernameToken = new
    CustomerInfo.SecurityUsernameToken();
    customerInfoSecurity.UsernameToken.Username = "root";
    customerInfoSecurity.UsernameToken.Password = "letmein";

    currentAccountSecurity = new Risaribank_Demo.CurrentAccount.Security();
    currentAccountSecurity.UsernameToken = new
    CurrentAccount.SecurityUsernameToken();
    currentAccountSecurity.UsernameToken.Username = "root";
    currentAccountSecurity.UsernameToken.Password = "letmein";

    . . .
```

That should be all the housekeeping we need to call our Web Services. Now we get to the stage of writing the code that is run when a user clicks a button.

5.2.2. Event Handling Code: Search Button

Switch back to your Designer view, and double-click the “Search” button in your Form. Your IDE will switch over to the code view, and a new member function to handle the button click will be created.

When this button is clicked, we want to take the contents of textBox1 (which is the Customer ID), and send this to our Customer Information web service. The web service should return the required customer information for that ID.

We break that customer information down into its respective parts, and then add that to our listView.

The code differs slightly between VC# 2005 and VC# 2008, choose the one that corresponds to your version.

5.2.3. Code for VC# 2005

```
private void button1_Click(object sender, EventArgs e)
{
    // create a new C# instance of the Web Service
    customerinformationRootService service =
        new customerinformationRootService();
    service.SecurityValue = customerInfoSecurity;

    // clear out any existing items from listView
    listView1.Items.Clear();

    // create a new key value that we send to the web service
    customerinformationGroupKeyType key =
        new customerinformationGroupKeyType();

    // set the CustomerNumber to the contents of textBox1
    key.CustomerNumber = textBox1.Text;

    // set up a variable to store the result
    customerinformationRootElementType results;

    try
    {
        // call the "list" operation of web service!
        results = service.list(key);
    }
    catch (Exception exp)
    {
        // an error occurred -
        // the web service will report the error
        // as part of the exception.
        MessageBox.Show(exp.Message, "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return;
    }
}
```

```

// Add the results to our list view
for (int i = 0;
    i != results.customerinformationRoot.Length; i++)
{
    // Store each customer found
    // in a local variable for ease of access.
    customerinformationGroupType customer =
        results.customerinformationRoot[i];

    // create a new row
    ListViewItem lv =
        new ListViewItem(customer.CustomerNumber);

    // add the rest of the items in the row
    lv.SubItems.Add(customer.FirstName);
    lv.SubItems.Add(customer.Surname);
    lv.SubItems.Add(customer.AddressLine1);
    lv.SubItems.Add(customer.AddressLine2);
    lv.SubItems.Add(customer.City);
    lv.SubItems.Add(customer.Postcode);
    lv.SubItems.Add(customer.DateOfBirth);

    // add the row to the listView
    listView1.Items.Add(lv);
    listView1.View = View.Details;
    listView1.FullRowSelect = true;
}
}

```

5.2.4. Code for VC# 2008

```

private void button1_Click(object sender, EventArgs e)
{
    // create a new C# instance of the Web Service
    customerinformationRootPortTypeClient service =
        new customerinformationRootPortTypeClient();

    // clear out any existing items from listView
    listView1.Items.Clear();

    // create a new key value that we send to the web service
    customerinformationGroupKeyType key =
    new customerinformationGroupKeyType();

    // set the CustomerNumber to the contents of textBox1
    key.CustomerNumber = textBox1.Text;

    // empty header
    customerinformationGroupHeader header = null;

    // set up a variable to store the result
    customerinformationRootElementType results;

    try
    {
        // call the "list" operation of web service!
    }
}

```

```

        results = service.list(customerInfoSecurity, ref header,
key);
    }
    catch (Exception exp)
    {
        // an error occurred -
        // the web service will report the error
        // as part of the exception.
        MessageBox.Show(exp.Message, "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return;
    }

    // Add the results to our list view
    for (int i = 0;
i != results.customerinformationRoot.Length; i++)
    {
        // Store each customer found
        // in a local variable for ease of access.
        customerinformationGroupType customer =
            results.customerinformationRoot[i];

        // create a new row
        ListViewItem lv =
        new ListViewItem(customer.CustomerNumber);

        // add the rest of the items in the row
        lv.SubItems.Add(customer.FirstName);
        lv.SubItems.Add(customer.Surname);
        lv.SubItems.Add(customer.AddressLine1);
        lv.SubItems.Add(customer.AddressLine2);
        lv.SubItems.Add(customer.City);
        lv.SubItems.Add(customer.Postcode);
        lv.SubItems.Add(customer.DateOfBirth);

        // add the row to the listView
        listView1.Items.Add(lv);
        listView1.View = View.Details;
        listView1.FullRowSelect = true;
    }
}

```

5.2.5. Event Handling Code: Account Details Button

Now that we have the ability to call a web service to get the customer information, we can also call a web service which will find the Current Account details for that customer.

To do this, we will take the Customer ID of the row selected in the list View, and call our CurrentAccount web service with this ID. This will give the user the ability to select a customer from a list, and then see what balances their current account has.

Again, we switch back to our Design view, and double-click the “Get Account Details” button. A new member function called “button2_click” will be created in the code view.

Again, the code differs slightly between versions. Choose the suitable one from below

5.2.6. Code for VC# 2005

```
private void button2_Click(object sender, EventArgs e)
{
    // create a new C# instance of the CurrentAccount web service
    currentaccountRootService service =
        new currentaccountRootService();
    service.SecurityValue = currentAccountSecurity;

    // get the currently selected Customer ID
    String currentCustomerId =
        listView1.SelectedItems[0].SubItems[0].Text;

    // create a new key value that we send to the web service
    currentaccountGroupKeyType key = new currentaccountGroupKeyType();

    // set the CustomerNumber to the CID
    key.CustomerNumber = currentCustomerId;

    // set up a variable to store the result
    currentaccountRootElementType results;

    try
    {
        // call the "get" operation of the web service!
        results = service.list(key);
    }
    catch (Exception exp)
    {
        // an error occurred -
        // the web service will report the
        // error as part of the exception.
        MessageBox.Show(exp.Message, "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return;
    }

    // Now put the results of the web service
    // into the Balance and Overdraft text boxes
    float bal =
        float.Parse(results.currentaccountRoot[0].Balance,
            NumberStyles.Currency) / 100;
    float od =
        float.Parse(results.currentaccountRoot[0].OverdraftLimit,
            NumberStyles.Currency) / 100;

    textBox2.Text = bal.ToString();
    textBox3.Text = od.ToString();
}
```

5.2.7. Code for VC# 2008

```
private void button2_Click(object sender, EventArgs e)
{
```

```

// create a new C# instance of the CurrentAccount web service
currentaccountRootPortTypeClient service =
    new currentaccountRootPortTypeClient();

// get the currently selected Customer ID
String currentCustomerId =
listView1.SelectedItems[0].SubItems[0].Text;

// create a new key value that we send to the web service
currentaccountGroupKeyType key = new
currentaccountGroupKeyType();

// set the CustomerNumber to the CID
key.CustomerNumber = currentCustomerId;

// empty key
currentaccountGroupHeader header = null;

// set up a variable to store the result
currentaccountRootElementType results;

try
{
    // call the "get" operation of the web service!
    results = service.list(currentAccountSecurity, ref header,
key);
}
catch (Exception exp)
{
    // an error occurred -
    // the web service will report the
    // error as part of the exception.
    MessageBox.Show(exp.Message, "Error",
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    return;
}

// Now put the results of the web service
// into the Balance and Overdraft text boxes
float bal =
float.Parse(results.currentaccountRoot[0].Balance,
NumberStyles.Currency) / 100;
float od =
float.Parse(results.currentaccountRoot[0].OverdraftLimit,
    NumberStyles.Currency) / 100;

textBox2.Text = bal.ToString();
textBox3.Text = od.ToString();
}

```

5.4 Building the Code

You can compile the code by hitting F6, or Build -> Build Solution.

Hopefully, all is well, but in the case of errors try the following:

- Obviously misspellings are often the cause of compilation errors. Ensure that all object names, and variable names are spelt correctly.

- The C# has a neat trick where you can type the first few letters of an object, and by hitting Control + Space, it will bring up the suggested object names. For names starting with “cu”, such as “customerinformation” or “currentaccount”, try typing this and hitting Control + Space. It may give you an indication of a misspelt object name.
- Similarly, if you need the name of a member of an object, type the object name, followed by a dot (“.”) and Control+Space. The list of available proposals should appear.

5.5 Running the code

By hitting F5 or Debug -> Start Debugging, you can run your code. In the Customer Number text box, you may enter * and hit the Search button to call the Customer Information web service and get a list of all the customers in the CustomerInformation table.

From the resultant list, select the record you are interested in, and hit Get Account Details. This will call CurrentAccount web service to retrieve the current account balances for this customer.

If you hit problems, you may wish to debug your code by adding breakpoints in your code. See the IDE documentation for further information.

6. Conclusion

This tutorial shows how to access MySQL from C# using the SOA Gateway. As you can see, you have built a powerful application that uses Web Services to retrieve information in real-time.

7. Appendix

Form1.cs for VC# 2005

[Code available here](#)

Form1.cs for VC# 2008

[Code available here](#)