Lab Guide for Introduction to ADO.NET 2.0

Author(s)	Sachin Kumar, Pawan Kumar Deulkar
Authorized by	Srikantan Moorthy
Creation/Revision Date	May-2009
Version	1.2

Education and Research Department
Infosys Technologies Limited
Electronic City
Hosur Road
Bangalore - 561 229, India.

Tel: 91 80 852 0261-270
Fax: 91 80 852 0362
www.infy.com
mailto:E&R@infy.com

Document Revision History

Version	Date	Author(s)	Reviewer(s)	Description
1.0	Nov 2007	Nagasubramanya, Prakash Khude, Leena Patil	Pushpalatha Devendra, Komal Papdeja	Baselined
1.1	May 2008	Leena Patil, Mahesh M S, Nagasubramanya	Pushpalatha Devendra	Baselined
1.2	May-2009	Pawan Kumar Deulkar, Sachin Kumar	Sachin Kumar	Changed the flow of topics, included more good technology usage practices, rewritten content for data binding and extended/refined for few others.

Contents

Context

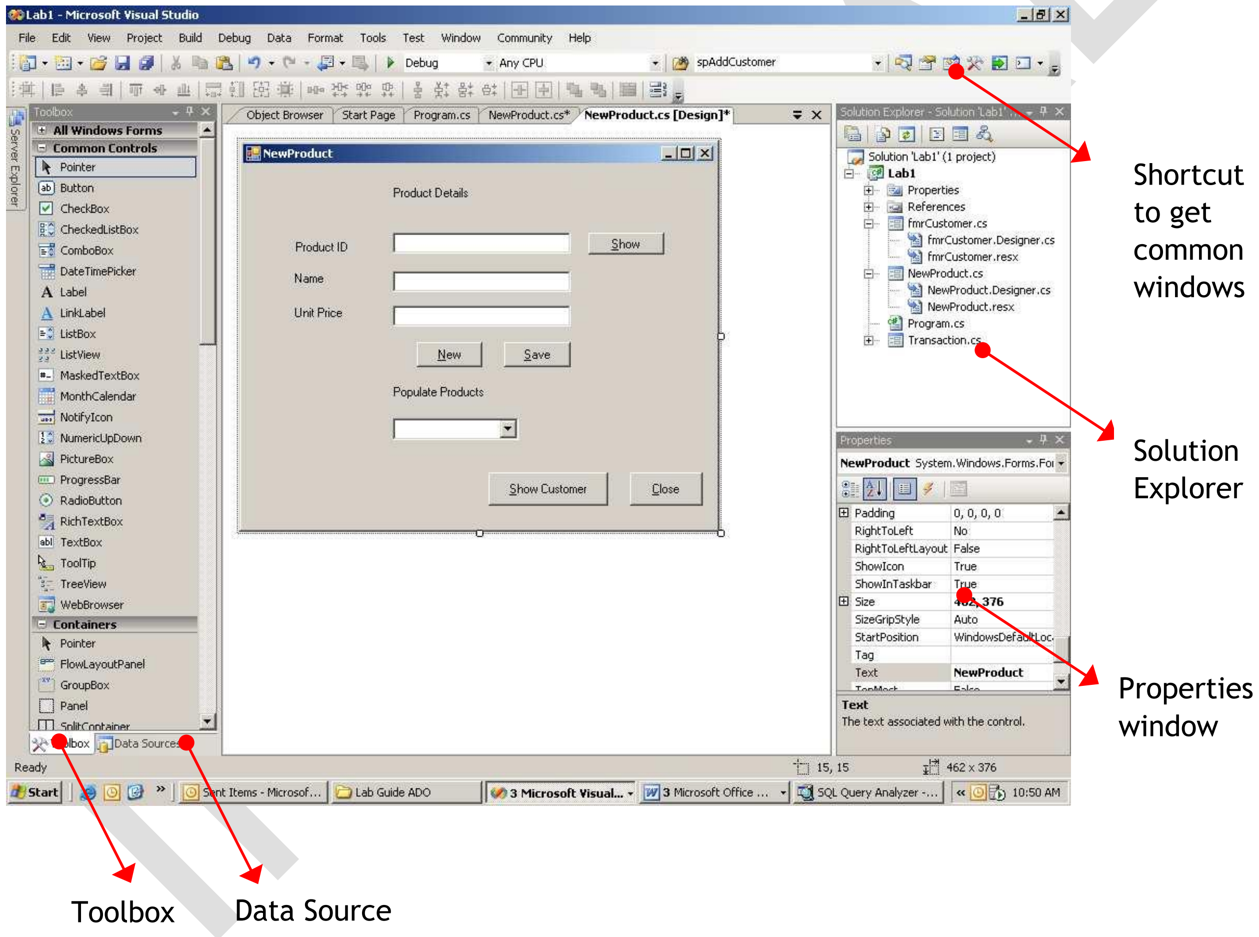
This document contains assignments to be completed as part of the hands on for the subject "Introduction to ADO.NET 2.0" (Course code: LA1008).

Pre-requisite

Before running any of the given Demo, perform the following task.

- Copy and run the script provided to you in your database.

Familiarization with Visual Studio 2008 IDE



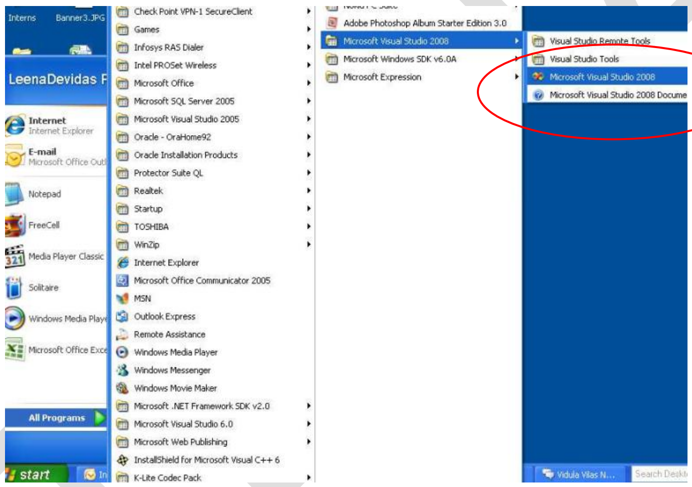
Day-1 Assignments

All the assignments in this section must be completed on Day 1 of “Introduction to ADO.NET 2.0” Course.

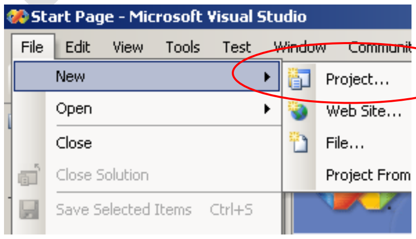
Head Start Assignment : Getting Started with Visual Studio 2008

Objective: To create first Windows application with database connectivity and execute it.
Problem Description: Design a Windows application and work with Visual studio 2008 IDE.
Background: To create a windows application, study different controls and properties of those controls.

Step 1: Go to Start → All Programs → Microsoft Visual Studio 2008 → select Microsoft Visual Studio 2008 as shown in the figure below.

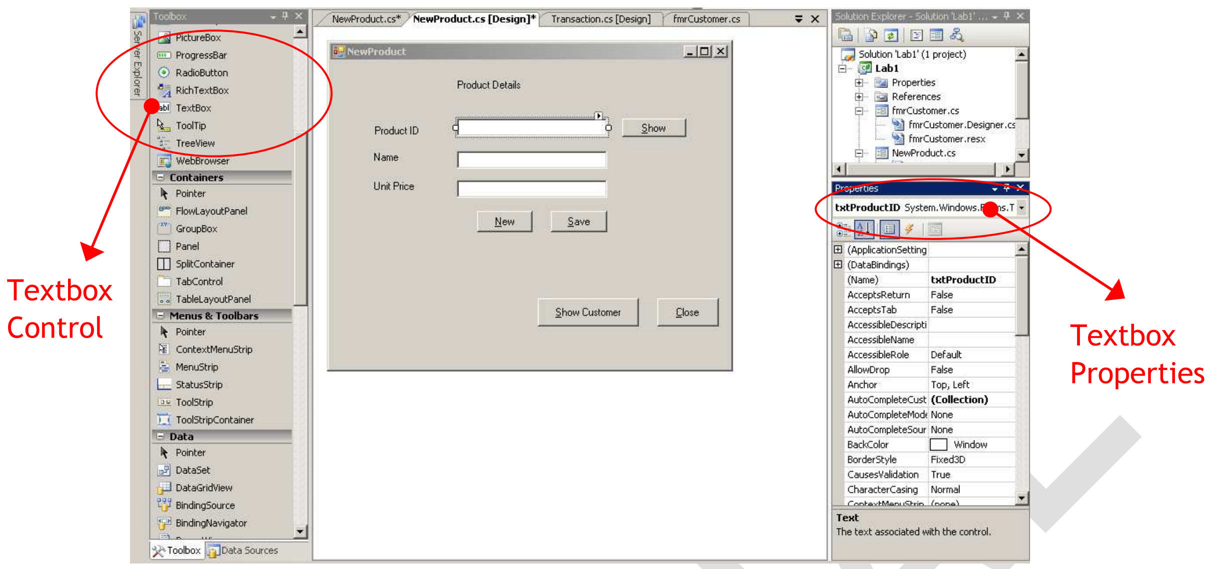

Microsoft Visual Studio 2008
Visual Studio Remote Tools
Visual Studio Tools
Microsoft Visual Studio 2008
Microsoft Visual Studio 2008 Docume
All Programs
start

Step 2: Once you have opened Visual Studio .NET 2008 IDE, go to:



Step 3: File →New Project→C# →Windows Forms Application →Provide project Name (Ex. Lab1) → Select Folder Location → OK

Step 4: Drag and drop Textbox, Label and Button controls from Toolbox and rename the form as ‘frmNewProduct’ as shown in the below screenshot.



txtProductID System.Windows.Forms.T

(ApplicationSetting	
(DataBindings)	
(Name)	**txtProductID**
AcceptsReturn	False
AcceptsTab	False
AccessibleDescripti	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoCompleteCust	**(Collection)**
AutoCompleteMode	None
AutoCompleteSour	None
BackColor	Window
BorderStyle	Fixed3D
CausesValidation	True
CharacterCasing	Normal
ContextMenuStrip	(none)

Text
The text associated with the control.

Rename the controls as given in the following table.

Control Type	Control Naming Standard
TextBox1	txtProductID
TextBox2	txtName
TextBox3	txtUnitPrice
Button1	btnShow
Button2	btnNew
Button3	btnSave
Button4	btnClose
Button5	btnShowCustomer
Label1,Label2,Label3,Label4	lblProductDetails, lblProductID, etc.

Summary of this exercise:
You have just learnt

To create a Windows Application, design the form.

To know various Controls and properties of those controls.

Assignment 1: Working with app.config and command object: using ExecuteScalar()

Objective: To illustrate how to create app.config file, with appropriate connection string and use ExecuteScalar () method of SqlCommand object.
Problem Description: Get next Product Id from the database. Use Max function of MS-SQL-Server and get connected to the database using connected architecture.
Estimated time: 20 Min

Note: This assignment is in continuation with the previous assignment.
Step 1: To add “App.config” file, right click on Project Name in solution explorer. Select Add → New Item → Application Configuration File → Add.

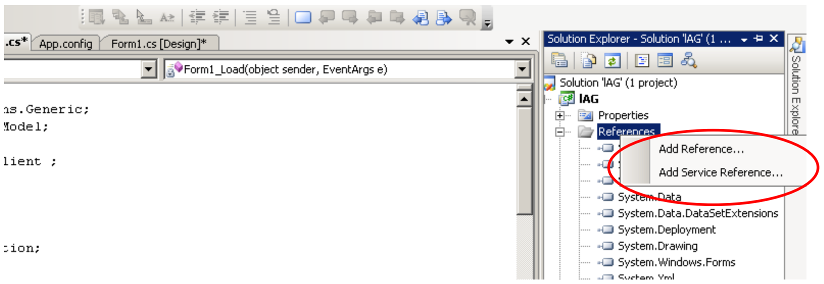
Step 2: Add connection string credentials in “App.config” file. Write the code given below between the “configuration” root node.
Note:
Data Source =mysgecsqldb02 (SQL Server Name)
Initial Catalog=NET07TempDB (SQL Data Base Name)
User Id and Password (User ID and Password of SQL Server)

```
<configuration>
  <connectionStrings>
    <add name ="ConnectionString"
providerName="System.Data.SqlClient" connectionString ="Data
Source=mysgecsqldb02; User ID=Trainer001; Password=infy@123;
Initial Catalog=ADONET; "/>

  </connectionStrings>
</configuration>
```
Replace above code with your MS SQL User details

Attributes of “add” tag:
Name: The connection string name used to access this tag from the application.
ProviderName: Name of the provider you are going to use for your application. Set this attribute to "System.Data.SqlClient" provider.
ConnectionString: The connection string you are going to work on.

Step 3: Right click on Add Reference → Select → System.Configuration → Add

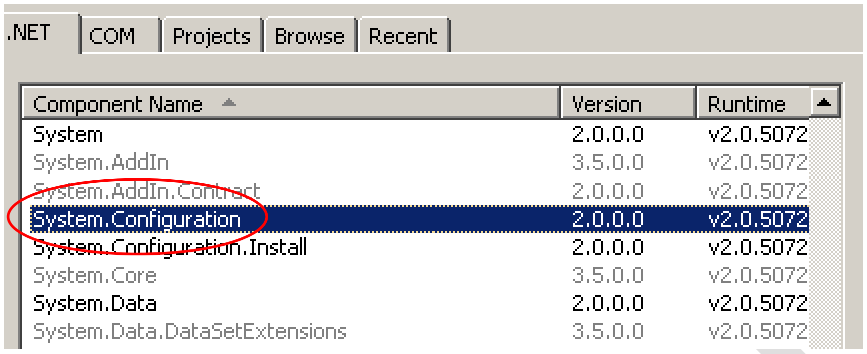


```
ns.Generic;
Model;

lient ;



tion;
```



Component Name	Version	Runtime
System	2.0.0.0	v2.0.5072
System.AddIn	3.5.0.0	v2.0.5072
System.AddIn.Contract	2.0.0.0	v2.0.5072
System.Configuration	2.0.0.0	v2.0.5072
System.Configuration.Install	2.0.0.0	v2.0.5072
System.Core	3.5.0.0	v2.0.5072
System.Data	2.0.0.0	v2.0.5072
System.Data.DataSetExtensions	3.5.0.0	v2.0.5072

Step 4: Double click on the form. You will get the source code file as shown below: The namespace usage is to be placed at the top

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

Step 5: Double click on Form and choose 'View Code' option. In Code behind file (eg. Lab1.cs) inside forms constructor write following code:

```
public partial class FrmNewProduct : Form
{
    //Declaring the connection
    object private SqlConnection
    ConADONET; public
    FrmConnection()
    {
      InitializeComponent();
      //extracting the connection string from 'App.config'file.
      ConADONET = new SqlConnection( ConfigurationManager.
             ConnectionStrings["ConnectionString"].ToString());
    }
}
```

Step 6: Go to 'frmNewProduct' form, double click on 'New' Button and type the following code

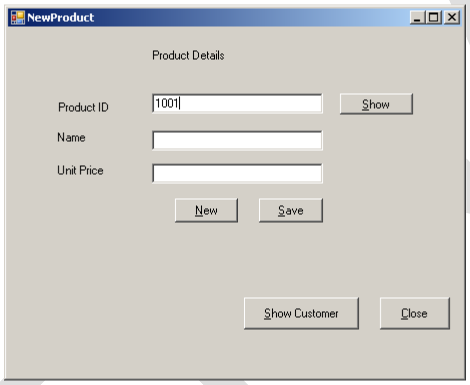
```
private void btnNew_Click(object sender, EventArgs e)
{
      /* Disable txtProductID so that the user will not be able to
      provide the value, Clear Name and Unit Price textbox and
      enable Save button */
       txtName.Text="";
       txtUnitPrice.Text="";
       btnSave.Enabled = true;
       int NextProductID;
```

```
        SqlCommand CmdNextProductID = new SqlCommand("select
        Max(ProductID) + 1 from Product", ConADONET);

        ConADONET.Open();
        NextProductID = (int)CmdNextProductID.ExecuteScalar();
        ConADONET.Close();

        txtProductID.Text = NextProductID.ToString();
  }
```

Step 7: Build the application: use Ctrl-Shift-B.
Run the application: use Ctrl-F5.
You will get the output as shown in following screenshot.



Summary of this exercise:
You have just learnt
Create and retrieving the information from App.config file
To create a Windows Application for database connectivity and write code.
How to configure connection and command objects.

Assignment 2: Working with Data Layer and command object: using ExecuteScalar()

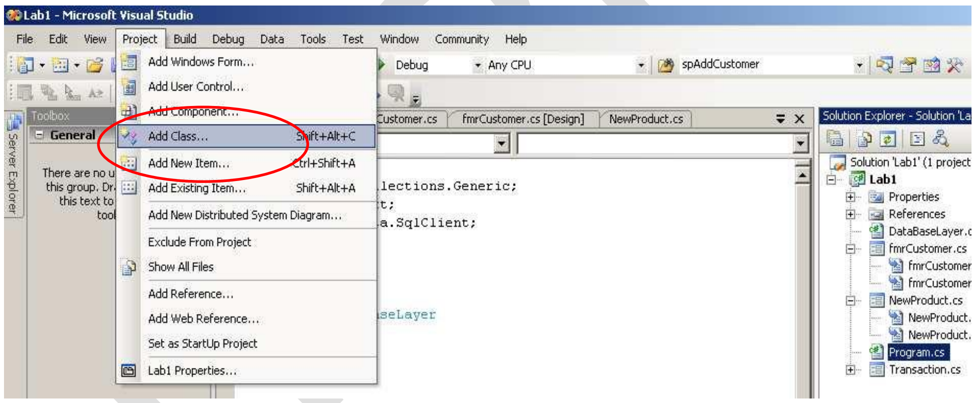
Objective: To illustrate how to create Data Base layer and use ExecuteScalar () method of SqlCommand object.

Problem Description: Get next Product Id from the database. Use Max function of MS-SQL-Server and get connected to the database using connected architecture.

Background: To perform the query operation which is returning a single value as output from the query, ExecuteScalar () method is used. This method will return an object.

Estimated time: 20 Min
Note: This assignment is in continuation with the head start assignment and is similar to previous assignment.

Step 1: To create data base layer for your application,
Go to Project → Add Class → name it as "DataBaseLayer.cs"



You will find the code as shown below:

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

namespace Infosys.Lab1
{
     class DatabaseLayer
     {
     }
}
```

Add the below namespace

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

Add the Connection Object in the Default Constructor

```
public class DatabaseLayer
{
   private SqlConnection ConADONET;
   //default Contructor added
   public DatabaseLayer()
   {
     ConADONET = new SqlConnection( ConfigurationManager.
     ConnectionStrings["ConnectionString"].ToString();
   }
}
```

Step 2: Create a method called "GetNextProductID" as shown below within DatabaseLayer class:

```
public int GetNextProductID()
{
     int NextProductID;
     SqlCommand CmdNextProductID = new SqlCommand("select
     Max(ProductID) + 1 from Product", ConADONET);
     ConADONET.Open();
     NextProductID = (int)CmdNextProductID.ExecuteScalar();
     ConADONET.Close();
     return NextProductID;
}
```

Step 3: Go to 'frmNewProduct' form, double click on 'New' Button and type the following code.

```
private void btnNew_Click(object sender, EventArgs e)
{
/* Disable txtProductID so that the user will not be able to
provide the value, Clear Name and Unit Price textbox and enable
Save button */
      txtName.Text="";
      txtUnitPrice.Text="";
       btnSave.Enabled = true;
      DataBaseLayer Obj = new DataBaseLayer();
      txtProductID.Text =Convert.ToString (Obj.GetNextProductID());
}
```

Step 4: Double click on ‘Close’ Button and type following code.

```
        this.Close();
```

Step 5: Build the application: use Ctrl-Shift-B.
Run the application: use Ctrl-F5.

Step 6: Click on ‘New’ button and check if the application is displaying new product ID in the textbox.

Summary of this exercise:
You have just learnt

How to create Data Layer (separating data access code and presentation code) To execute the query which returns a single value using ExecuteScalar method

Assignment 3: Working with Data Layer and command object: using ExecuteNonQuery()

Objective: To illustrate the usage of ExecuteNonQuery () method of Command object.

Problem Description: Use ExecuteNonQuery () method to insert a new record in database table. Write code in such a way that data access layer is separated from the presentation layer.

Background: To execute DML or DDL commands on the database using command object, ExecuteNonQuery () method is used. This method will return the number of row affected.

Estimated time: 20 Min

Note: This assignment is in continuation with the previous assignment.

Step 1: Write a method named "InsertNewProduct" in DataBaseLayer.cs which should accept Name and Unit Price and return an integer value.

```
public int InsertNewProduct(string Name, int UnitPrice)
{
int Ret=0;
SqlCommand CmdInsertProduct = new SqlCommand ("Insert into
Product values ('" + Name + "'," + UnitPrice + " )",
ConADONET);
        ConADONET.Open();
        Ret = (int)
        CmdInsertProduct.ExecuteNonQuery();
        ConADONET.Close();
return Ret

}
```

Step 2: Go to 'frmNewProduct' form, double click on 'Save' Button and type the required code.

```
private void btnSave_Click(object sender, EventArgs e)
{
/*
//WRITE YOUR CODE HERE
Validate the details (Name and Unit Price) provided by the user.
Both the fields are mandatory
If proper data is provided, write code to execute 'InsertRecord'
method by passing the textbox values entered by the user as
parameters to the method. Get return-value of ‘InsertRecord’
method and display appropriate messages to the user(use
MessageBox)
*/
}
```

Step 3: Build and execute your application.
Summary of this exercise:

You have just learnt
How to work with ExecuteNonQuery () method of SqlCommand object.

Note: Pressing F7 switches to Code Window whereas pressing Shift-F7 switches to Form Window

Assignment 4: Working with command object: using ExecuteReader()

Objective: To create Windows application with database connectivity and execute it. Illustrate use of ExecuteReader () method of SqlCommand object.

Problem Description: Design a Windows application to get connected to the backend and execute SELECT command returning multiple rows.

Background: To read the data from the database using command object, ExecuteReader () method is used. This method will return the records set, we can read those records one by one using Read () method.

Estimated time: 20 Min

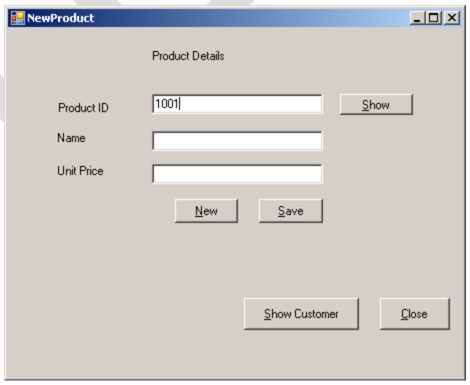
Step 1: Open the 'frmNewProduct' form created in assignment no 1. Set enable property of 'btnSave' button to false in the Properties Window.

Step 2: Double click on 'Show' Button and form declaration type the following code.

```
public partial class FrmNewProduct: Form
{
   //Declaring the connection
   object private SqlConnection
   ConADONET; public
   FrmConnection()
   {
   InitializeComponent()
   ;
   //extracting the connection string from 'App.config'
   file. ConADONET = new SqlConnection(
   ConfigurationManager.
          ConnectionStrings["ConnectionString"].ToString ());
   }
   private void btnShow_Click(object sender, EventArgs e)
   {
         /* Create a command object having following properties:
            CommandText: To select the details of product ID
            provided by the user.
            Connection: The connection object ConADONET*/
         SqlCommand CmdShowProducts = new SqlCommand("select * from
         Product where ProductId=" + txtProductID.Text, ConADONET);
         /* Open the connection before executing the command */
         ConADONET.Open();
```

```
          /* Execute Select command to get Product details
             and initialize SqlDataReader object */
          SqlDataReader DrShowProduct
          =CmdShowProducts.ExecuteReader();
          /* Check if Product ID exists in the database */
          if (DrShowProduct.Read())
          {
            txtName.Text = DrShowProduct["ProductName"].ToString();
            txtUnitPrice.Text =
            DrShowProduct["unitprice"].ToString();
          }
          else
          {
             MessageBox.Show(" No Records Found!!!");
              /* Clear the contents of Product ID textbox */
              txtProductID.Text = "";
          }
            DrShowProduct.Close();
            ConADONET.Close();
      }
```

Step 3: Build the application: use Ctrl-Shift-B.
Run the application: use Ctrl-F5.
You will get the output as shown in following screenshot.



1001

Summary of this exercise:
You have just learnt
To create a Windows Application for database connectivity and write code.
How to configure connection and command objects.

Further assignment for the Day
Add the following requirements for Product Form
1. Validate Product ID should not be Blank
2. On form Load disable Save button
3. Make Product name and Unit Price ready only on form load.

Assignment 5: Implementing MARS Concept

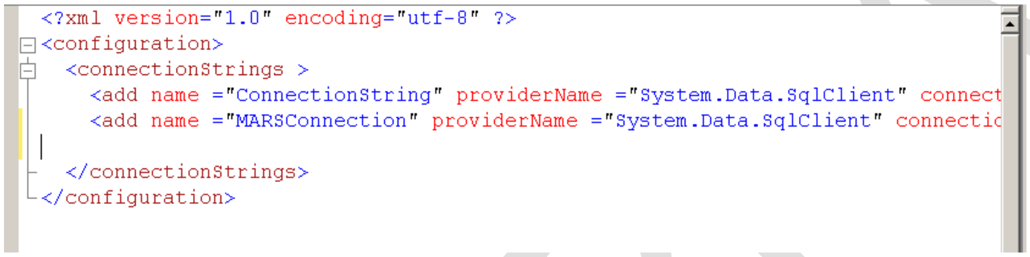
Objective: Working with Multiple Active Result Sets of the commands that are working on a single connection.

Problem Description: Create an application that accepts Customer ID as input. The application should display

The detail information of customer.

The orders and Order Date for that customer.

Background: MARS in SQL Server works only in MS-SQL-Server 2005 and above. Goto to App.Cong file add new connection name as



```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings >
    <add name ="ConnectionString" providerName ="System.Data.SqlClient" connect
    <add name ="MARSConnection" providerName ="System.Data.SqlClient" connectio

  </connectionStrings>
</configuration>
```

```
<add name ="MARSConnection" providerName ="System.Data.SqlClient"
connectionString ="Data Source=mysgecsqldb02;Initial Catalog
=ADONET; User ID=Trainer001;Password=infy@123;
MultipleActiveResultsets=true"/>
```

By doing so, you instruct ADO.NET that you are interested in keeping multiple active result sets open on this connection. Following instructions will run if and only if you are using “SqlClient” provider.

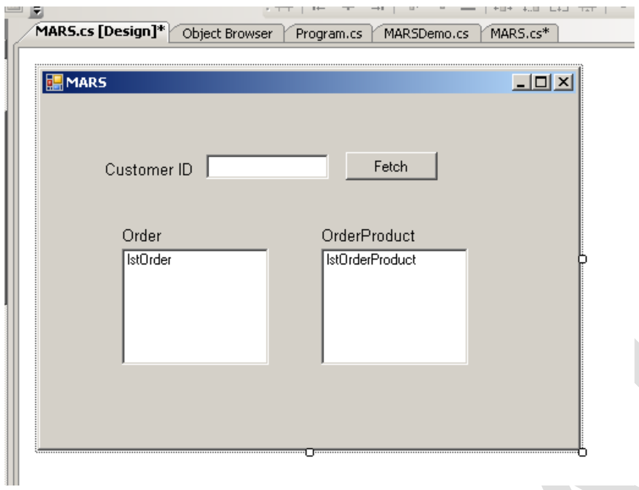
Estimated time: 20 Min.

Step 1: Add a new form to the application created in previous assignment. Name the form as “MarsDemo.cs”.

Design the form as shown below:

- Add two ListBox controls name as lstOrder & lstOrderProduct
- One TextBox
- One Label
- One Button

Rename all the Controls as per required



Step 2: Declare SQL Connection object

```
public partial class MARS : Form
{
    private SqlConnection ConMARS;
    public MARS()
    {
     InitializeComponent();
     ConMARS = new SqlConnection( ConfigurationManager.
     ConnectionStrings["MARSConnection"].ToString());
    }
}
```

Step 3: Double click on 'Fetch' Button and type following code.

```
private void btnFetch_Click(object sender, EventArgs e)
{
      string OrderStr1 = "select orderid from orders where
      customerid="+txtCustomerID.Text;

      string OrderStr2 = "select ProductID from orderProduct where
      orderID=@porderID";
      SqlCommand cmdOrder = new SqlCommand(OrderStr1, ConMARS);
      SqlCommand cmdOrderProduct = new SqlCommand(OrderStr2,
      ConMARS);
      //Connection is opened once
      ConMARS.Open();
      // create and add parameter to second command object
      SqlParameter ParaOrderID = new SqlParameter("@pOrderID",
      SqlDbType.Int);
      cmdOrderProduct.Parameters.Add(ParaOrderID);
      //First Reader
      SqlDataReader DRorder1 = cmdOrder.ExecuteReader();
      while (DRorder1.Read())
      {
            //Populate First ListBox
            lstOrder.Items.Add(DRorder1["Orderid"].ToString());
            cmdOrderProduct.Parameters["@pOrderID"].Value =
            DRorder1["Orderid"];

           //Second Reader
           SqlDataReader DRorder2 = cmdOrderProduct.ExecuteReader();
           while (DRorder2.Read())
           {
                 //Populates Second ListBox
                 lstOrderProduct.Items.Add(
                 DRorder2["ProductID"].ToString());
            } DRorder2.Close();
      }
      DRorder1.Close();
      //Connection is closed
      ConMARS.Close();
}
```

Step 4: Design the above application using Generic Factory Model and check the output.

Summary of this exercise:
You have just learnt

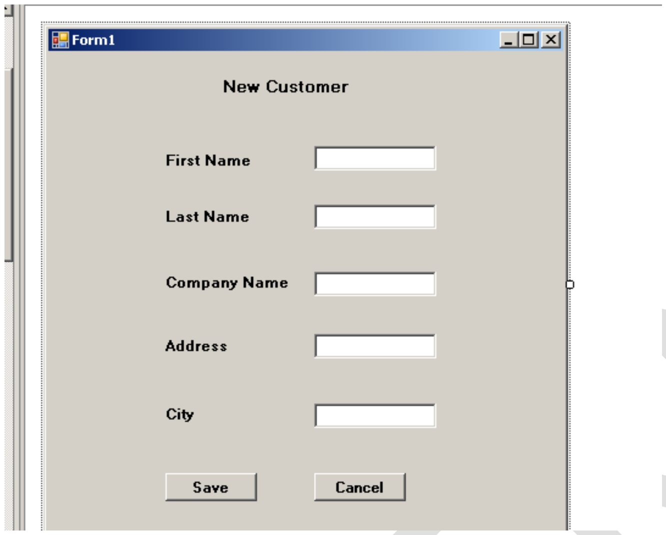
To create a Windows Application for database connectivity and write code.

To work with Multiple Active Result Sets of the commands that are working on a single connection.

Self Assignment A: Working with Database layer and Execute command

Problem Description: Open new frmCustomer Design window form drag and drop two buttons name as “btnNew” and “btnSave”.

Design the form and write the whole application using Database layer



On click BtnSave create a method “AddNewCustomer” in Database Layer to insert a new record to Customer table call on btnSave button event.

Give the Link from Product form to Add new Customer Form (Next Self Assignment).

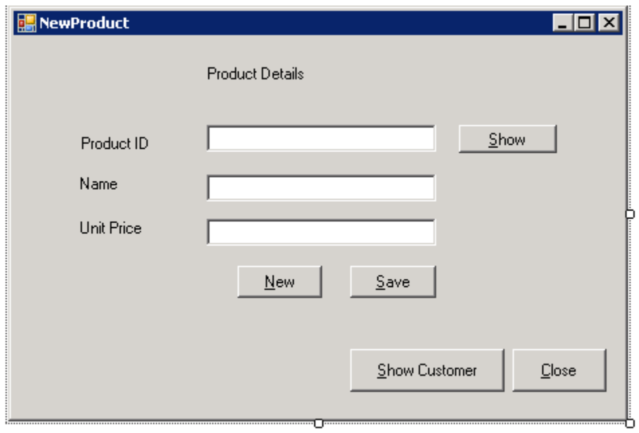
Summary of this exercise:
You have just learnt

- How to execute stored procedure which is having Input, Output parameters and Return value.
- How to set the values for input parameter and get values of output parameter and return value.

Follow the next assignment to implement the last point i.e. navigation between two forms.

Self Assignment B: Working with Main and Show methods

Step 1: Open the FormNewProduct.cs form and double click on 'Show Customer' button.



Type the following code in Click event handler of 'Show Customer' button:

```
private void btnShowCustomer_Click(object sender, EventArgs e)
{
    FormCustomer NewCustomer = new
    FormCustomer(); NewCustomer.Show();
    this.Hide();
}
```

Step 5: Execute it and see the output.

```
/* Try this out
   Would you like to execute Customer Form
   only? Goto Program.cs
*/
namespace Infosys.Lab1
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
          Application.EnableVisualStyles();
          Application.SetCompatibleTextRenderingDefault(false);
          /* Comment Product Run event and add Customer Run event
     */
          //Application.Run(new FormNewProduct());
          Application.Run(new FormCustomer());
        }
    }
}
```

Summary of this exercise:
You have just learnt

- How to navigate between two forms.