Lab Guide for Introduction to ADO.NET 2.0

Education and Research

Infosys®

POWERED BY INTELLECT DRIVEN BY VALUES

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

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

Day-2 Assignments

All the assignments in this section must be completed on Day 2 of “Introduction to ADO.NET 2.0” Course.

Assignment 6: Working with parameterized queries

Objective: Writing parameterized query, creating parameters and executing the query.

Problem Description: Design a form to accept new Product details from the user and insert a new record in product table using parameterized query.

Background: In Product table suppose we would like to add new Products and the values of the columns for new record are entered in TextBoxes by the end user. Parameterized query can be used to pass values to the query at run time.
You can pass any number of parameters according to your requirement.

Estimated time: 20 Min

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

Step 1: Open the project created in the previous assignment.

Step 2: Open DataBaseLayer.cs file, open method called “**InsertNewProduct**” with below mentioned input parameters and return value. Comment the previous code and write the below code

```
public int InsertNewProduct(string Name, int UnitPrice)
{
      int InsertStatus;

SqlCommand cmdProduct = new SqlCommand(" insert into Product values
(@pName,@pUnitPrice)", ConADONET);

        cmdProduct.Parameters.AddWithValue("@pName", Pname);
        cmdProduct.Parameters.AddWithValue("@pUnitPrice", Price );
/*
//WRITE YOUR CODE HERE
Execute the command object and Handle and return the value to calling
function.
*/

      /* We can also Create parameters in this way */
     /*
      SqlParameter ParamProductName = new SqlParameter();
      ParamProductName.ParameterName = "@PName";
      ParamProductName.SqlDbType = SqlDbType.VarChar;
      ParamProductName.Size = 20;
      ParamProductName.Value = Name;

       CmdInsert.Parameters.Add(ParamProductName);
       */

}
```

Step 3: Open the frmNewProduct Design window and double click on Save Button and type the following code.

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

/*
//WRITE YOUR CODE HERE
Create a object and calling class method and handle the return calue
to a Message Box display
*/

}
```

Step 4: Build and execute the application.

Summary of this exercise:
You have just learnt
- How to create SqlParameter object
- How to pass parameters to a SQL query at run time.

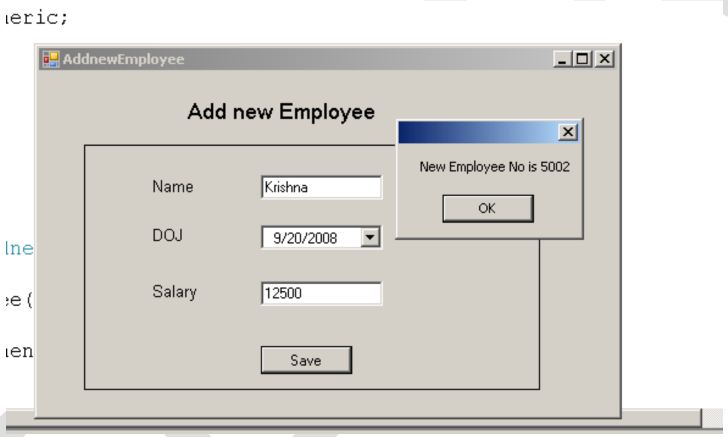
Assignment 7: Working with Stored Procedure

Objective: To illustrate how to execute stored procedures from a front end application designed in .NET.

Problem Description: Design a new form “AddNewEmployee.cs” and Add the details of a particular Employee using stored procedure **`AddNewEmployee`**.

Background: We have a stored procedure in our database that inserts the details of a Employee and return a value. This procedure has one input parameter, two output parameters and a return value.

Estimated time: 30 Min



Step 1: Drag and drop 3 labels , 2 Textbox ,Button and Date picker control and rename the control as per Coding Standard

Step 2: Include required namespace and SQL connection object to a constructor

Step 3: Open DataBaseLayer.cs file. Add a new method called “AddNewEmployee” with below mentioned input parameters and return value.

```
public int AddnewEmployee(string Name,DateTime Doj,double  Salary)
 {
  int Ret = 0;
  try
  {
      SqlCommand cmdEmployee = new SqlCommand();
      cmdEmployee.Connection = ConADONET;
      cmdEmployee.CommandText = "AddNewEmployee";
      cmdEmployee.CommandType = CommandType.StoredProcedure;

      cmdEmployee.Parameters.AddWithValue("@pName",Name );
      cmdEmployee.Parameters.AddWithValue("@pDoj",Doj);
      cmdEmployee.Parameters.AddWithValue("@pSalary", Salary);
```

```
     // to add Output
      SqlParameter ParaOut = new SqlParameter("@pnewEmpNo",
                                                          SqlDbType.Int);
        ParaOut.Direction = ParameterDirection.Output;
        cmdEmployee.Parameters.Add(ParaOut);

        // to add Return parameter
        //SqlParameter ParaRet = new SqlParameter("@pRet",
                                                          SqlDbType.Int);
        //ParaRet.Direction = ParameterDirection.ReturnValue ;
        //cmdEmployee.Parameters.Add(ParaRet);

        ConADONET.Open();
        cmdEmployee.ExecuteNonQuery();
        ConADONET.Close();
        Ret = Convert.ToInt16(ParaOut.Value);
     }
     catch (Exception er)
     {
     }
     finally
     {
     ConADONET.Close();
     }
 return Ret;
 }
}
```

Step 4: Open the AddNewemployee Design window and double click on Save Button and type the following code.

```
private void btnSave_Click(object sender, EventArgs e)
{
     int Ret;
     DatabaseLayer Obj = new DatabaseLayer();
     Ret=Obj.AddnewEmployee(txtFirstName.Text, dateTimePicker1.Value,
Convert.ToDouble (txtSalary.Text  ));

MessageBox.Show( " New Employee No is "  +Ret.ToString());
}
```

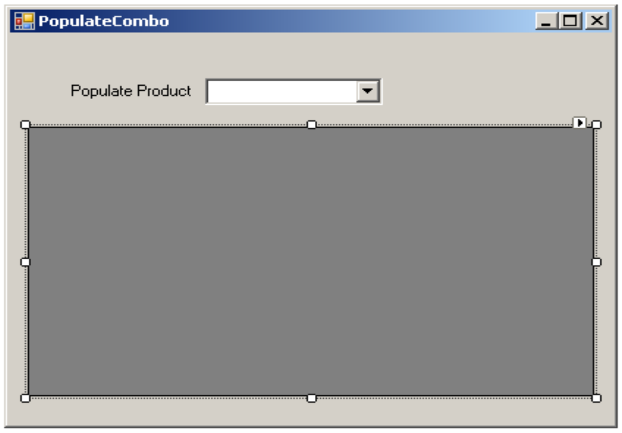
Assignment 8: Working with DataTable and DataAdapter

Objective: To illustrate how to work with the disconnected scenario with the DataTable and the DataAdapter.

Problem Description: Design a new form "PopulateCombo.cs".A Combo Box will hold a list of Product ID's. On selection of a particular Product ID a GridView displays the details of the record

Background: The Products table in the database will be used in this assignment as a source of data.

Step 1: Place a label , a Combo Box and a DataGridView control on to a new form named 'PopulateDemo' and rename the controls according to the naming conventions.
Step 2: Include the required namespace for the use of SQL related classes. Create an SQL Connection with the help of the App.Config file.



Step 3: Your code file for the form must have the following code:

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

namespace Day2Demos
{
    public partial class PopulateCombo : Form
    {
        private SqlConnection ConADONET;
        public PopulateCombo()
        {
            InitializeComponent();

           /*Creating the connection */
```

```
            ConADONET = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString
"].ToString());
        }

/*Writing code to be executed when the form loads */

private void PopulateCombo_Load(object sender, EventArgs e)
{

     /*Creating the DataAdapter */
     SqlDataAdapter daProductID = new SqlDataAdapter("select *from
Product", ConADONET);

      DataTable dtProducts = new DataTable();

     /*Filling the DataTable */
      daProductID.Fill(dtProducts);
      cmbProduct.DataSource = dtProducts;

       /*Setting the Display and the Value Member properties of the
ComboBox from the Columns of the DataTable */
      cmbProduct.DisplayMember = dtProducts.Columns[1].ToString();
      cmbProduct.ValueMember = dtProducts.Columns[0].ToString();
}
```

Step 4: In the event handler for the Selected Index changed event of the ComboBox, type the following code:

```
private void cmbProduct_SelectedIndexChanged(object sender, EventArgs
e)
{
if (cmbProduct.SelectedIndex != 0)
     {
           MessageBox.Show("Data : " + cmbProduct.Text.ToString());
           MessageBox.Show("Value : " +
cmbProduct.SelectedValue.ToString());
           string Pid = cmbProduct.SelectedValue.ToString();
           SqlDataAdapter daProductDetails = new
SqlDataAdapter("select *from product where ProductID=" + Pid,
ConADONET);

           DataTable dtProductDetails = new DataTable();

           daProductDetails.Fill(dtProductDetails);

           /*Binding the DataTable to the DataGridView */

           dgvProductDetails.DataSource = dtProductDetails;
     }
}
```

Step 5: Run and view the result

Summary of the Exercise:
You have just learned :

- How to fetch data into a DataTable using a DataAdapter.
- Populating a ComboBox with Data
- Binding the data from the DataTable to a DataGridView

Assignment 9: Working with Dataset and DataView and binding them to a DataGridView

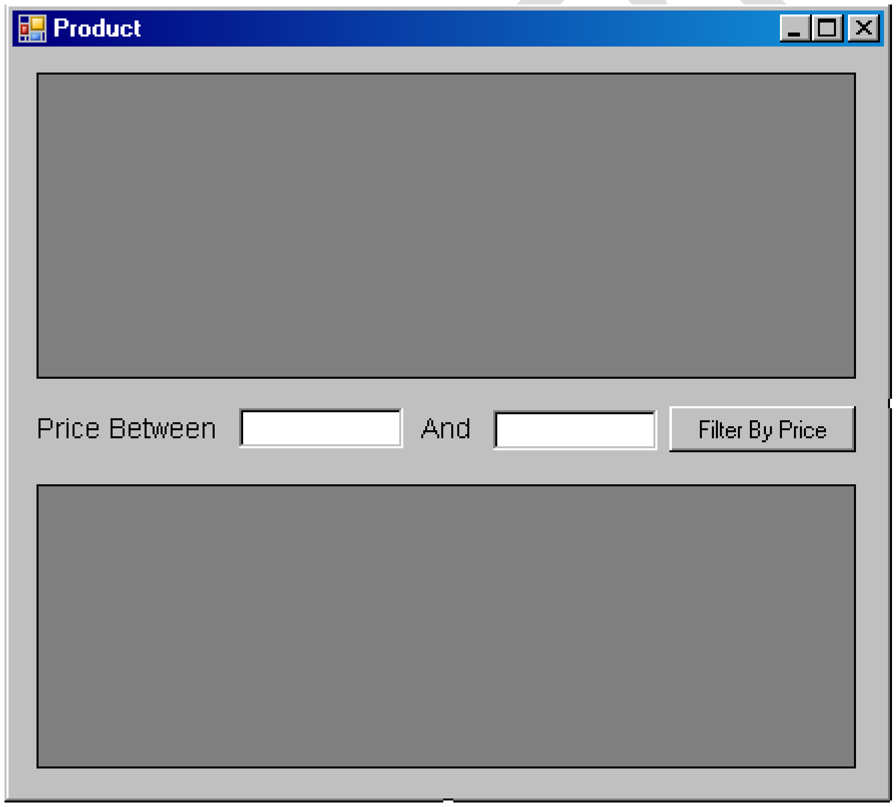
Objective: To illustrate 'How to use a Dataset and bind it to GridView and how a View can be created on tables in Dataset using Dataview and its properties like Sort and RowFilter and finally binding the DataView to GridView.

Problem Description: Create a Form "Product.cs" and fetch all products in a GridView and then filter the data on the basis of price (between start and end) and display in another GridView.

Background: We have a Table called Product in database which contains details of a particular product (Product ID, Product Name, and Unit Price of the product).

Estimated Time: 25 mins

Step 1: Create a new Project and to your application add a new form. Name the form "Product.cs". Design the form as shown below:



a) Add Two GridViews name them, gvALLProducts and gvProductinView respectively.
b) Add two Textboxes as txtPriceStart and txtPriceEnd.

c) Add one Button as btnFilter.

Step 2: Add a Dataset Variable to be used globally by all the events in the Code behind File(Product.cs)

```
public Product()
{
      InitializeComponent();
}
// ADD This Dataset to be globally used by all the events.
DataSet dsProducts = null;
```

Step 3: Add a Application configuration File (app.config) to the application and add your connection string there as shown below:

```
connectionStrings>
    <add name ="MyConString" providerName ="System.Data.SqlClient"
connectionString ="Data Source=<DataSource Given>;Initial Catalog=<Database
name>;User ID= <Your ID>;Password=<Your Password> "/>
  </connectionStrings>
```

Step 3: Add A Database Layer to your Application i.e create a DataBaseLayer.cs file and add the following code to it.

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

namespace Grid_Population_DataView
{
    class DataBaseLayer
    {
        SqlConnection ConnCustomer;
        String strConnection;

        public DataBaseLayer()
        {
            strConnection = ConfigurationManager.
                        ConnectionStrings["MyConString"].ToString();
            ConnCustomer = new SqlConnection(strConnection);
        }
```

Step 4: Add a method called LoadProductDetails which will return a dataset object containing the product table data from database as shown below :

```
        public DataSet LoadProductDetails()
        {
            /* Fetch the product details using SqlDataAdapter and store in
            Dataset dsCustomer */

            SqlDataAdapter daProduct = new SqlDataAdapter("SELECT * FROM
                  Product", ConnCustomer);

            DataSet dsCustomer = new DataSet();
            daProduct.Fill(dsCustomer, "Product_ADO");

            return dsCustomer;
        }
    }
}
```

Step 5: Double Click on the form and in Page_Load event of the form, add the following code to load data to the DataSet from the database to which you have connected. We are going to fetch the data from the database using class DataBaseLayer Object and its method LoadProductDetails() in a Dataset which will be global to all the event handling functions of this form.

```
private void Product_Load(object sender, EventArgs e)
{
      if (dsProducts != null)
      {
      MessageBox.Show(" Loaded from Local Dataset");
      gvAllProducts.DataSource =
            dsProducts.Tables["Product_ADO"];
      }
      else
      {     /* Create a DataBaseLayer object to connect and fetch data from
                  the database    */
            DataBaseLayer dbObj = new DataBaseLayer();
            MessageBox.Show(" Loading Data From Database");

            /* Load data using LoadProductDetails method of dbObj into
            Global dataset dsProducts*/
            dsProducts = dbObj.LoadProductDetails();

            /* Bind the Dataset to the first GridView */
            gvAllProducts.DataSource =
                  dsProducts.Tables["Product_ADO"];
      }
}
```

Step 6: Add the following code to the Button_ Click event of btnFilter .Now we are going to create a DataView which will be a subset of global DataSet already populated.Here we will sort the data in DataView using it's Sort property by unit price of the products and then the data will be filtered according to the starting and ending price given in respective textboxes.

```
private void btnFilter_Click(object sender, EventArgs e)
        {
            /* Create DataView Object form global Dataset dsProducts
and Sort the dataView By UnitPrice column of the Table
'Product_ADO' in the dataset.*
            DataView dvProducts = new
DataView(dsProducts.Tables["Product_ADO"]);
            dvProducts.Sort = "UnitPrice";
   try
{
/*Convert the Text in Textboxes from String to int and
 store in integer variables */
                   int PriceStart = Convert.ToInt32(txtPricestart.Text);
                   int PriceEnd = Convert.ToInt32(txtPriceEnd.Text);

/*Add the condition to RowFilter property of Dataview
dvProducts created earlier */

                  dvProducts.RowFilter = "UnitPrice > " + PriceStart
                                         + " AND "
                                         + "UnitPrice < " + PriceEnd;

                  gvProductinView.DataSource = dvProducts;
            }
            catch (Exception ee)
            {
            /* All the Exceptions are caught here and there Error messages
are displayed */
                MessageBox.Show(ee.Message);
            }
        }
```

In the above code we have created a DataView Object dvProdusts from the table “Product_ADO” present in global Dataset dsProducts. By setting the Sort property of the dvProducts as “UnitPrice” the data in the Dataview will be sorted by UnitPrice of products in ascending order. Then after validating the data from textboxes contaning starting and ending price, items falling in this range will be filtered using RowFilter property of dvProducts.

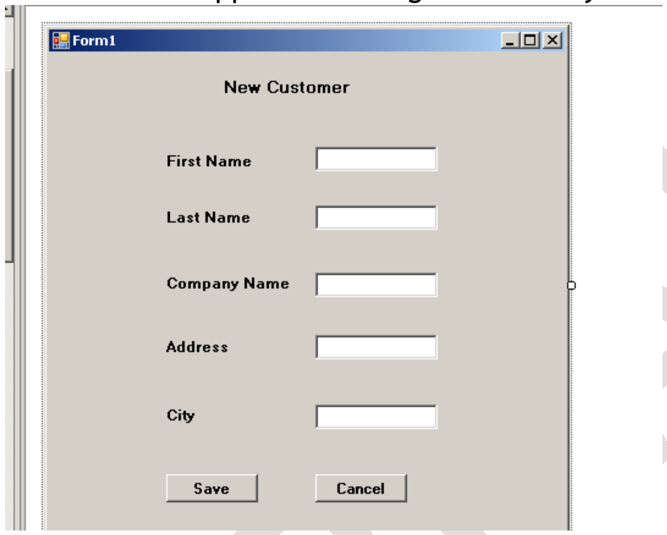
Step 7: Build and Execute the project.

Summary of the Exercise:
You have just learned :

- How to fetch data into a DataSet using adapter.
- Binding DataSet and DataView to GridView.
- Creating aView of the Data in Dataset using DataView.
- Using Sort and RowFilter property of the Dataview.

Self Assignment C: Working with Stored Procedure

Problem Description: Open 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 (it has to call the stored procedure “spAddNewCustomer” created in MS SQL)

Self Assignment D: Working with Stored Procedure

Objective: To illustrate how to execute stored procedures from a front end application designed in .NET.

Problem Description: Design a new form "FormCustomer.cs" and fetch the details of a particular customer using stored procedure.

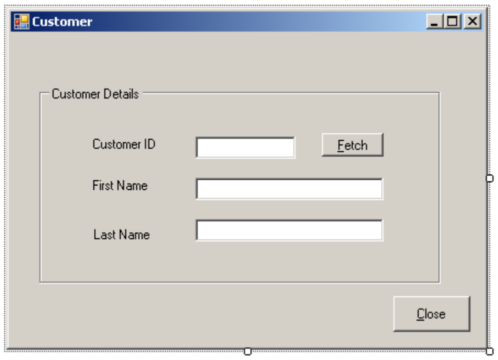
Background: We have a stored procedure in our database that returns the details of a customer. This procedure has one input parameter, two output parameters and a return value.

Estimated time: 30 Min

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

Step 1: Open the project created in the previous assignment.

Step 2: Add a new form in this project. Name the form as "FormCustomer.cs". Design the form as shown below:



Note: To add a new form go to Project → Add Windows form
OR
Right click on Project name in Solution Explorer Window, select Add → Windows Form

Step 3: Open DataBaseLayer.cs file. Add a new method called "**FetchCustomerSP**" with below mentioned input parameters and return value.

```
public int FetchCustomerSP (int CustomerId, ref string FirstName,
                             ref string LastName )
{
/*
//WRITE YOUR CODE HERE
Create a object and calling class method and handle the return calue
to a Message Box display
Assign the Output values to Ref variable (FirstName)
*/


}
```

Step 4: Open FormCustomer.cs in Design mode and double click on 'Fetch' button. Type the following code in Click event handler of Fetch button:

```
private void btnFetch_Click(object sender, EventArgs e)
{
     string FirstName=null ,LastName=null;
     int RetVal;

      /* Write code to create an object of DataBaseLayer class.
         Call FetchCustomerSP() method - pass appropriate parameters.
         Get the return value of FetchCustomerSP() method in RetVal.
      */

      /* Write code for checking the value of retVal variable to know
         if SP has been executed successfully. If CustomerID is found,
         load FirstName and LastName in the respective textboxes.
         Otherwise display appropriate message and clear the textboxes.
      */
}
```

Self Assignment E: Working with command object: using ExecuteReader()

Problem Description: Open frmNewProduct Design window form drag and drop a Combo Box on Form Load populate all the product names. On selection of Product Name the application should display its Unit Price on a Message Box (Use Database layer and return Datatable)