

ASP.NET Data Access Lab

Exercise 1: Working with ASP.NET DataBound Control



Developing user friendly commercial applications using ASP.NET 3.5 is now easy using various DataBound controls. These controls provide richness in data representation and interaction with it. Following are some of the examples of DataBound controls:

- GridView.
- DataList.
- ListView.

In this lab we will explore GridView control for performing operations like:

- Update.
- Delete.
- Select.
- Pagination.

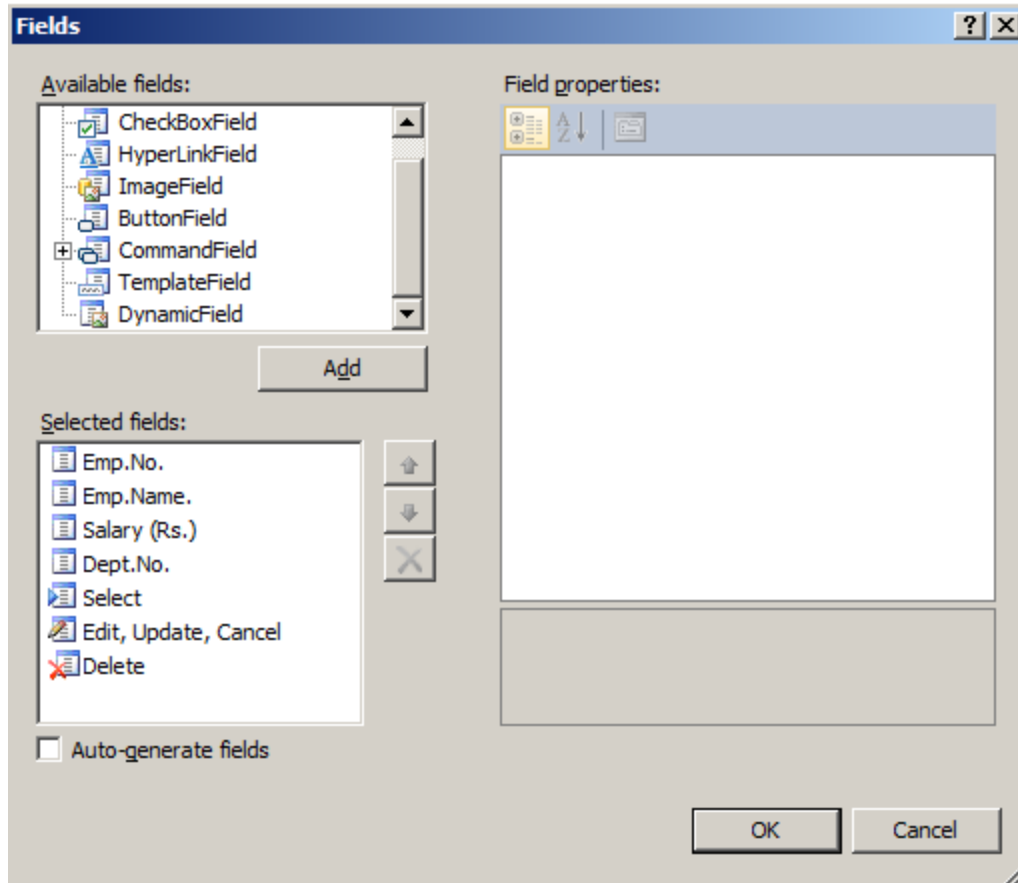
In this lab we will be using Employee table created as below:

| | Column Name | Data Type | Allow Nulls |
|---|-------------|-------------|--------------------------|
|  | EmpNo | int | <input type="checkbox"/> |
| | EmpName | varchar(50) | <input type="checkbox"/> |
| | Salary | int | <input type="checkbox"/> |
| | DeptNo | int | <input type="checkbox"/> |
|  | | | <input type="checkbox"/> |

Task 1: Open Vs2008/VS2010 and create a new Web Application project or Web Site. Name this as 'ASPNET_DatabaseProgramming'. Rename 'Default.aspx' to 'frmGridViewProgramming.aspx'.

Task 2: On the 'frmGridViewProgramming.aspx', drag and drop the 'GridView' control from the toolbox. Name this as 'gdvEmp'.

Task 3: In this task we will be defining BoundFields and CommandFields columns for the grid view, so select 'Smart Tag' of the control (upper right corner arrow.). Select edit columns and add columns as below:



Also from the property palate set following properties:

- AllowPaging -> True.
- PageSize -> 3.

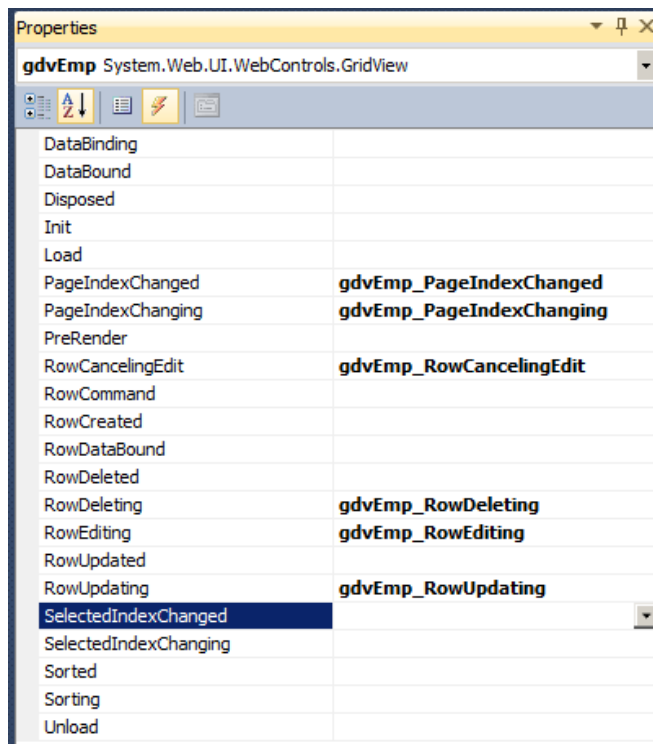
You can select the background color as per your wish. Put the Label control below the GridView

The GridView control will now look as below:

| Emp.No. | Emp.Name. | Salary (Rs.) | Dept.No. | | | |
|-----------|-----------|--------------|-----------|--------|------|--------|
| Databound | Databound | Databound | Databound | Select | Edit | Delete |
| Databound | Databound | Databound | Databound | Select | Edit | Delete |
| Databound | Databound | Databound | Databound | Select | Edit | Delete |
| 1 2 | | | | | | |

[!blerr]

Register to the events of the GridView as shown below:



Task 4: Open 'frmGridViewProgramming.aspx.cs' and use the following namespaces

System.Data and System.Data.SqlClient

At class level declare the following object references:

```

SqlConnection Conn;
SqlCommand Cmd;
SqlDataAdapter AdEmp;
DataSet Ds;

```

Write the following method in class:

```

private DataSet GetDs()
{
    AdEmp.Fill(Ds, "Employee");
    return Ds;
}

private void BindGrid(DataSet Ds)
{
    gdvEmp.DataSource = Ds.Tables["Employee"];
    gdvEmp.DataBind();
}

```

The code above fill the data in dataset and in 'GetDs()' method and in 'BindGrid()' method the dataset is bind with the GridView gdvEmp.

In the Page.Load event, we will connect to database server and call methods for Databinding to GridView as below:

```
protected void Page_Load(object sender, EventArgs e)
{
    Conn = new SqlConnection("Data Source=.;Initial
Catalog=Company;Integrated Security=SSPI");
    AdEmp = new SqlDataAdapter("Select * from Employee",Conn);

    Ds = new DataSet();

    if (this.IsPostBack == false)
    {
        Ds = GetDs();
        BindGrid(Ds);
    }
}
```

Now we will implement pagination operation for the GridView, so implement PageIndexChanged event as below:

```
protected void gdvEmp_PageIndexChanged(object sender, EventArgs e)
{
}

protected void gdvEmp_PageIndexChanging(object sender,
GridViewPageEventArgs e)
{
    gdvEmp.PageIndex = e.NewPageIndex;
    Ds = GetDs();
    BindGrid(Ds);
}
```

NewPageIndex property will return the value of the page number which we will be selecting from the GridView.

Run the page and check the output for pagination.

Now we will implement the functionality for Update. As you see that we have and 'Edit' button for each row of the GridView, when this button is clicked, it is then replaced by Update and Cencel button as below:

| Emp.No. | Emp.Name. | Salary (Rs.) | Dept.No. | | | |
|---------------|---------------|------------------------------------|---------------------------------|---------------------------------------|---|---------------------------------------|
| 101 | Natrajan | <input type="text" value="72000"/> | <input type="text" value="10"/> | | <input type="button" value="Update"/> <input type="button" value="Cancel"/> | |
| 102 | Makrand P. | 34500 | 30 | <input type="button" value="Select"/> | <input type="button" value="Edit"/> | <input type="button" value="Delete"/> |
| 103 | Mahehs Sabnis | 76000 | 20 | <input type="button" value="Select"/> | <input type="button" value="Edit"/> | <input type="button" value="Delete"/> |
| 1 2 3 4 5 6 7 | | | | | | |

So we need to implement following 3 events for Update and Cancel:

- RowEditing.
 - Fires when Edit button is clicked.
- RowUpdating.
 - Fires when Update button is clicked.
- CancelingEdit.
 - Fires when Cancel button is clicked.

The implementation is as below:

```

protected void gdvEmp_RowEditing(object sender, GridViewEditEventArgs
e)
{
    gdvEmp.EditIndex = e.NewEditIndex;
    Ds = GetDs();
    BindGrid(Ds);
}
protected void gdvEmp_RowUpdating(object sender,
GridViewUpdateEventArgs e)
{
    int Eno, Dno =0, Sal=0;
    try
    {
        #region Logic For Reading Selected Row
        Eno =
Convert.ToInt32(gdvEmp.Rows[e.RowIndex].Cells[0].Text);
        Sal = Convert.ToInt32(((TextBox)
gdvEmp.Rows[e.RowIndex].Cells[2].Controls[0]).Text );
        Dno =
Convert.ToInt32(((TextBox)gdvEmp.Rows[e.RowIndex].Cells[3].Controls[0]
).Text);
        #endregion

        Conn.Open();
        Cmd = new SqlCommand();

```

```

        Cmd.Connection = Conn;
        Cmd.CommandText = "Update Employee Set
Salary=@Salary,DeptNo=@DeptNo where EmpNo=@EmpNo";
        Cmd.Parameters.AddWithValue("@EmpNo",Eno);
        Cmd.Parameters.AddWithValue("@DeptNo", Dno);
        Cmd.Parameters.AddWithValue("@Salary", Sal);
        Cmd.ExecuteNonQuery();

    }
    catch (Exception ex)
    {
        lblerr.Text = ex.Message;
    }
    finally
    {
        Conn.Close();
    }
    gdvEmp.EditIndex = -1;
    Ds = GetDs();
    BindGrid(Ds);
}
protected void gdvEmp_RowCancelingEdit(object sender,
GridViewCancelEventArgs e)
{
    gdvEmp.EditIndex = -1;
    Ds = GetDs();
    BindGrid(Ds);
}

```

Finally we need to implement the Delete functionality. The code is as below:

```

protected void gdvEmp_RowDeleting(object sender,
GridViewDeleteEventArgs e)
{
    try
    {
        int Eno = 0;
        Eno =
Convert.ToInt32(gdvEmp.Rows[e.RowIndex].Cells[0].Text);
        Conn.Open();
        Cmd = new SqlCommand();
        Cmd.Connection = Conn;
        Cmd.CommandText = "Delete Employee where EmpNo=@EmpNo";
        Cmd.Parameters.AddWithValue("@EmpNo", Eno);
        Cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        lblerr.Text = ex.Message;
    }
}

```

```

    }
    catch (Exception ex)
    {
        lblerr.Text = ex.Message;
    }
    finally
    {
        Conn.Close();
    }
    gdvEmp.EditIndex = -1;
    Ds = GetDs();
    BindGrid(Ds);
}

```

Task 5: Run the application and try Pagination, Edit, Update, Cancel and Delete functionality.

Exercise 2: Working with Component based Programming and Database Programming Connected Architecture.

In this lab we will have an application development approach, where we will be creating DataAccess Layer, Value Object Layer and Client application for ASP.NET data access.

For this application, we will be using Department table as below:

| | Column Name | Data Type | Allow Nulls |
|---|-------------|-------------|--------------------------|
| ? | DeptNo | int | <input type="checkbox"/> |
| | Dname | varchar(50) | <input type="checkbox"/> |
| | Location | varchar(50) | <input type="checkbox"/> |
| ▶ | | | <input type="checkbox"/> |

Task 1: Open VS2008/VS2010 and create a Blank solution, name it as 'ASPNET_DataAccess'.

Task 2: In this solution add a new project of Class Library type, name it as 'Com.App.VO'. This will create 'Com.App.VO.dll'.

Task 3: Rename 'Class1.cs' to 'clsDepartmentVO' and write the class definition as below:

```

public class clsDepartmentVO
{
    private int _DeptNo;

    public int DeptNo

```

```

    {
        get { return _DeptNo; }
        set { _DeptNo = value; }
    }
    private string _Dname;

    public string Dname
    {
        get { return _Dname; }
        set { _Dname = value; }
    }
    private string _Location;

    public string Location
    {
        get { return _Location; }
        set { _Location = value; }
    }
}

```

Task 4: Build the project and make sure that it is error free.

Task 5: In the same solution add a new class library project, name it as 'Com.App.DataAccessObject'. This will create 'Com.App.DataAccessObject.dll'. This library will define connection manager and connect to database server and will perform DML Operations. Add the reference of 'Com.App.VO.dll' in this project.

Task 6: In this project, rename 'Class1.cs' to 'clsConnectionManager.cs' and write the following code:

```

public class clsConnectionManager
{
    SqlConnection Conn;
    SqlCommand Cmd;
    DataTable DtData;

    //Method for establishing Connection to Database
    public static SqlConnection
    CreateAndOpenConnection(SqlConnection Conn)
    {
        Conn.ConnectionString = "Data Source=.;Initial
        Catalog=Company;Integrated Security=SSPI";
        Conn.Open();
        return Conn;
    }
    //Method for closing Connection.

```

```

        public static void CloseConnection(SqlConnection Conn)
        {
            if (Conn.State == ConnectionState.Open)
            {
                Conn.Close();
            }
        }
//Method for Executing DMP Opartions
        public static int ExecutedMLOperations(SqlConnection Conn,
        SqlCommand Cmd)
        {
            Cmd.Connection = Conn;
            int intRes = Cmd.ExecuteNonQuery();
            return intRes;
        }
//Method for returning all rows in DataTable for specific //Table
        public static DataTable GetAllRowsFromTable(SqlConnection
        Conn, SqlCommand Cmd)
        {
            Cmd.Connection = Conn;
            SqlDataReader Reader = Cmd.ExecuteReader();
            DataTable DtAllRows = new DataTable();
            DtAllRows.Load(Reader);

            return DtAllRows;
        }
//Method for returning a specific value
        public static object GetSpecificColumnsValue(SqlConnection
        Conn, SqlCommand Cmd)
        {
            Cmd.Connection = Conn;
            SqlDataReader Reader = Cmd.ExecuteReader();
            object RetVal = Reader[0];
            return RetVal;
        }
    }

```

The above class is specific to Sql Server Database but CRUD operations can be performed on any table. Please study the above class carefully. You can think of optimization.

Task 7: In the same project add a new class; name it as 'clsDepartmentDAO.cs'. This class will make use of 'clsConnectionManager' to perform CRUD operations on Department Table. The code for the class is as below:

```

public class clsDepartmentDAO
{

```

```
SqlConnection Conn;
SqlCommand Cmd;

///Perfrom UInsert operations on Department
///Input Parameter 'clsDepartmentVO'
///Output Parameter integer
public int InsertDepartment( clsDepartmentVO objDepartmentVO)
{
    Conn = new SqlConnection();
    Cmd = new SqlCommand();
    //Create and Open Connection

    Conn = clsConnectionManager.CreateAndOpenConnection(Conn);

    string strInsertDepartment = "Insert into Department
(DeptNo,Dname,Location) Values (@DeptNo,@Dname,@Location)";
    Cmd.CommandText = strInsertDepartment;

    Cmd.Parameters.AddWithValue("@DeptNo",objDepartmentVO.DeptNo );
    Cmd.Parameters.AddWithValue("@Dname",
objDepartmentVO.Dname);
    Cmd.Parameters.AddWithValue("@Location",
objDepartmentVO.Location);

    //Call Executor
    int Res = clsConnectionManager.ExecutedMLOperations(Conn,
Cmd);

    //Close Connection
    clsConnectionManager.CloseConnection(Conn);

    return Res;
}

public DataTable GetAllDepartments()
{
    Conn = new SqlConnection();
    Cmd = new SqlCommand();

    Conn = clsConnectionManager.CreateAndOpenConnection(Conn);

    string strSelectAllRows = "Select * from Department";
```

```

        Cmd.CommandText = strSelectAllRows;

        DataTable DtDept =
clsConnectionManager.GetAllRowsFromTable(Conn, Cmd);
        clsConnectionManager.CloseConnection(Conn);

        return DtDept;
    }
}

```

(Note: Write, Update, Delete and Search operations yourself. It is **Mandatory**)

Task 8: Build the project.

Task 9: Now in the solution created in Task 1, add a new ASP.NET web application or Web Site. Rename 'Default.aspx' to 'frmDepartment.aspx'.

Task 10: Design the page as below:

Task 11: In the web site created add the reference of following assemblies:

- 'Com.App.VO.dll'- Created in Task 2.
- 'Com.App.DataAccessObject.dll' – Create in Task 5.

Task 12: Open 'frmDepartment.aspx.cs' and use the following namespace:

- using Com.App.VO;
- using Com.App.DataAccessObject.
- using System.Data.

At class level define the following object reference:

```

clsDepartmentDAO objDepartmentDAO;
clsDepartmentVO objDepartmentVO;

```

Write the following code in Page.Load event. This will call 'GetAllDepartments()' method from the 'clsDepartmentDAO' class as below:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (this.IsPostBack == false)
    {
        objDepartmentDAO = new clsDepartmentDAO();
        DataTable DtDept = objDepartmentDAO.GetAllDepartments();
        lstdname.DataSource = DtDept;
        lstdname.DataValueField = "DeptNo";
        lstdname.DataTextField = "Dname";
        lstdname.DataBind();
    }
}
```

Write the following code in 'Insert' button click, this will 'InsertDepartment()' method from 'clsDepartmentDAO' as below:

```
protected void btnInsert_Click(object sender, EventArgs e)
{
    try
    {
        objDepartmentVO = new clsDepartmentVO();
        objDepartmentVO.DeptNo = Convert.ToInt32(txtldno.Text);
        objDepartmentVO.Dname = txtldname.Text;
        objDepartmentVO.Location = txtldloc.Text;
        objDepartmentDAO = new clsDepartmentDAO();
        objDepartmentDAO.InsertDepartment(objDepartmentVO);
    }
    catch (Exception ex)
    {
        lblerr.Text = ex.Message;
    }
}
```

Task 13: Run the application and test the functionality.

Task 14: Implement Update, Delete and Find functionality on the page yourself.