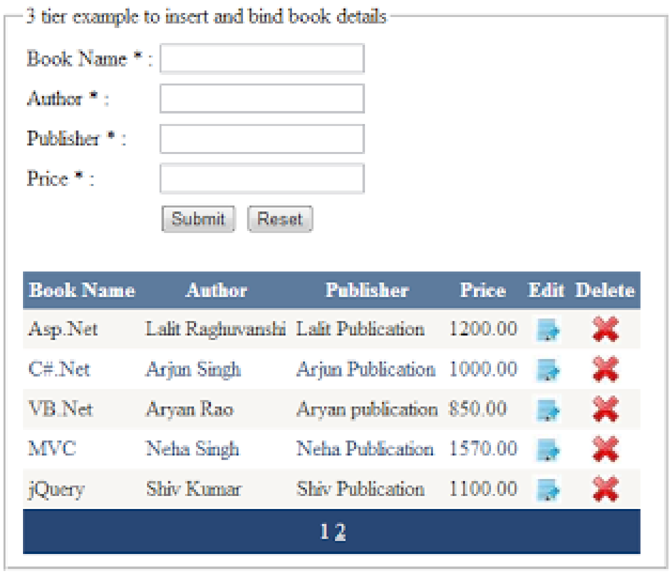
HOW TO CREATE 3 TIER APPLICATION IN ASP.NET C# TO INSERT,EDIT,UPDATE,BIND AND DELETE DATA

Lalit Raghuvanshi 09:16:00


3 tier example to insert and bind book details

Book Name * :	
Author * :	
Publisher * :	
Price * :	

Book Name	Author	Publisher	Price	Edit	Delete
Asp.Net	Lalit Raghuvanshi	Lalit Publication	1200.00		
C#.Net	Arjun Singh	Arjun Publication	1000.00		
VB.Net	Aryan Rao	Aryan publication	850.00		
MVC	Neha Singh	Neha Publication	1570.00		
jQuery	Shiv Kumar	Shiv Publication	1100.00		

Click on image to enlarge

Introduction: In this article you will learn how to create 3 tier project with example to Save, Bind, Edit, Update and Delete Book Details using asp.net and Sql server as a back end database as shown in image.

Note : Before reading this article you must read the article How to setup 3 tier architecture project in asp.net C# to set up the 3 tier architecture for this application because this article is the continued part of the previous article where you learned how to set up 3 tier project.

I am assuming that you have setup the 3-tier project after reading the above mentioned article.

So now our next step is to create the database.

- So create a Sql server database e.g. "BookDb" and in that create a table using the script below:

```
CREATE TABLE [dbo].[BookDetails]
(
        [BookId] [int] IDENTITY(1,1) NOT NULL,
        [BookName] [varchar](100),
        [Author] [varchar](100),
        [Publisher] [varchar](200),
        [Price] [decimal](18, 2) NOT NULL
)
```

- Now create the stored procedure to insert the book details in the table.

```
CREATE PROCEDURE [dbo].[InsertBookDetails_SP]
            @BookName                VARCHAR(100),
            @Author                  VARCHAR(100),
            @Publisher               VARCHAR(200),
            @Price                   DECIMAL(18,2)
AS
BEGIN
            INSERT INTO BookDetails
             (
                        BookName,Author,Publisher,Price
              )
            VALUES
             (
                        @BookName,@Author,@Publisher,@Price
             )
END
```

- Create a stored procedure to update the book detail

```
CREATE PROCEDURE [dbo].[UpdateBookRecord_SP]
            @BookId                  INT,
            @BookName                VARCHAR(100),
            @Author                  VARCHAR(100),
            @Publisher               VARCHAR(200),
            @Price                   DECIMAL(18,2)
AS
BEGIN
            UPDATE BookDetails SET
                        BookName=@BookName,
                        Author=@Author,
                        Publisher=@Publisher,
                        Price=@Price
            WHERE BookId=@BookId
END
```

- Create a stored procedure to delete book record

```
CREATE PROCEDURE [dbo].[DeleteBookRecords_Sp]
            @BookId          INT
AS
```

```
BEGIN
            DELETE FROM BookDetails WHERE BookId=@BookId
END
```

- Create a stored procedure to delete book record

```
CREATE PROCEDURE [dbo].[FetchBookRecords_Sp]
AS
BEGIN
            SELECT * FROM BookDetails
END
```

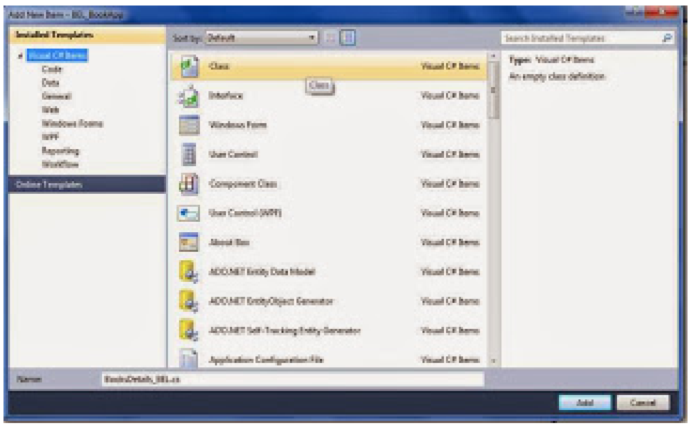
- Now we need to connect our asp.net application with the sql server database. So in the web.config file create the connection string under the <configuration> tag as:

```
<connectionStrings>
   <add name="conStr" connectionString="Data Source=localhost;Initial
Catalog=BooksDb;Integrated Security=True"/>
  </connectionStrings>
```

Now database part is done. Now it's time to write the code for each layer.

Create Class In BEL

- So let's create an entity/ property class in BEL.
- Right click on the "BEL_BookApp" in the solution explorer -> Add -> New Item -> Select "Class" and name it "BooksDetails_BEL.cs" as shown in image below.



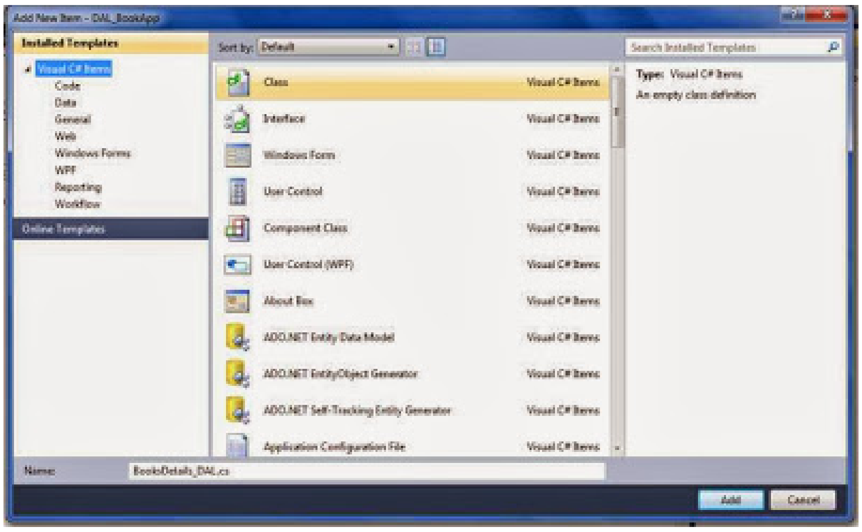
Click on image to enlarge

- Create the property for each column in the table "BookDetails". So write the code in BooksDetails_BEL.cs as:

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

namespace BEL_BookApp
{
  public class BooksDetails_BEL
   {
     public int BookId { get; set; }
     public string BookName { get; set; }
     public string Author { get; set; }
     public string Publisher { get; set; }
     public decimal Price { get; set; }
   }
}
```

Create Class in DAL

- Now we need to create a class in DAL to perform database operations.
- So right click on the "DAL_BookApp" in the solution explorer -> Add -> New Item -> Select "Class" and name it "BooksDetails_DAL.cs" as shown in image below.


Add New Item - DAL_BookApp
Installed Templates
Visual C# Items
Code
Data
General
Web
Windows Forms
WPF
Reporting
Workflow
Online Templates
Sort by: Default
Search Installed Templates
Class Visual C# Items
Interface Visual C# Items
Windows Form Visual C# Items
User Control Visual C# Items
Component Class Visual C# Items
User Control (WPF) Visual C# Items
About Box Visual C# Items
ADO.NET Entity Data Model Visual C# Items
ADO.NET EntityObject Generator Visual C# Items
ADO.NET Self-Tracking Entity Generator Visual C# Items
Application Configuration File Visual C# Items
Type: Visual C# Items
An empty class definition
Add
Cancel

Name: BooksDetails_DAL.cs

Click on image to enlarge

Write the following code in **BooksDetails_DAL.cs** :

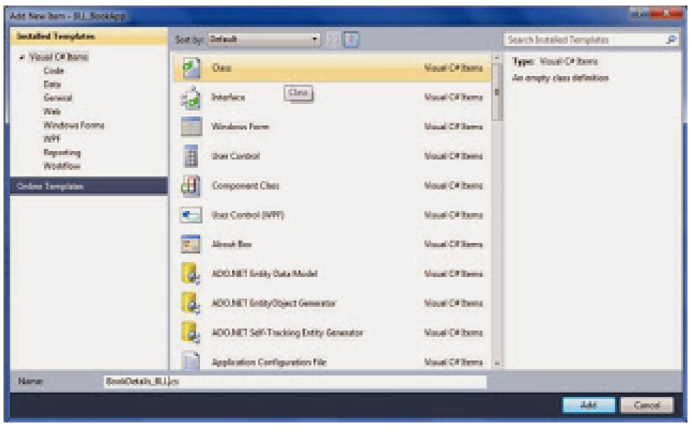
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using BEL_BookApp;
```

```
namespace DAL_BookApp
{
  public class BooksDetails_DAL
   {
     SqlConnection con
= new SqlConnection(ConfigurationManager.ConnectionStrings["conStr"].Connec
tionString);
     public Int32 SaveBookDetails(BooksDetails_BEL objBEL)
     {
        int result;
        try
        {
          SqlCommand cmd = new SqlCommand("InsertBookDetails_SP", con);
          cmd.CommandType = CommandType.StoredProcedure;
          cmd.Parameters.AddWithValue("@BookName", objBEL.BookName);
          cmd.Parameters.AddWithValue("@Author", objBEL.Author);
          cmd.Parameters.AddWithValue("@Publisher", objBEL.Publisher);
          cmd.Parameters.AddWithValue("@Price", objBEL.Price);
          if (con.State == ConnectionState.Closed)
          {
             con.Open();
          }
          result = cmd.ExecuteNonQuery();
          cmd.Dispose();
          if (result > 0)
          {
             return result;
          }
          else
          {
             return 0;
          }
        }
        catch (Exception ex)
        {
          throw;
        }
        finally
        {
          if (con.State != ConnectionState.Closed)
          {
             con.Close();
          }
        }
    }

    public DataSet GetBookRecords()
    {
        DataSet ds = new DataSet();
        try
        {
            SqlCommand cmd = new SqlCommand("FetchBookRecords_Sp", con);
            cmd.CommandType = CommandType.StoredProcedure;
            SqlDataAdapter adp = new SqlDataAdapter(cmd);
            adp.Fill(ds);
            cmd.Dispose();
        }
        catch (Exception ex)
        {
            throw;
        }
        finally
        {
            ds.Dispose();
        }
        return ds;
    }

    public Int32 DeleteBookRecord(BooksDetails_BEL objBEL)
    {
        int result;
        try
        {
            SqlCommand cmd = new SqlCommand("DeleteBookRecords_Sp", con);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@BookId", objBEL.BookId);
            if (con.State == ConnectionState.Closed)
            {
                con.Open();
            }
            result = cmd.ExecuteNonQuery();
            cmd.Dispose();
            if (result > 0)
            {
                return result;
            }
            else
            {
                return 0;
            }
        }
        catch (Exception ex)
        {
            throw;
        }
        finally
        {
            if (con.State != ConnectionState.Closed)
            {
                con.Close();
            }
        }
    }

    public Int32 UpdateBookRecord(BooksDetails_BEL objBEL)
    {
        int result;
        try
        {
            SqlCommand cmd = new SqlCommand("UpdateBookRecord_SP", con);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@BookId", objBEL.BookId);
            cmd.Parameters.AddWithValue("@BookName", objBEL.BookName);
            cmd.Parameters.AddWithValue("@Author", objBEL.Author);
            cmd.Parameters.AddWithValue("@Publisher", objBEL.Publisher);
            cmd.Parameters.AddWithValue("@Price", objBEL.Price);
            if (con.State == ConnectionState.Closed)
            {
                con.Open();
            }
            result = cmd.ExecuteNonQuery();
            cmd.Dispose();
            if (result > 0)
            {
                return result;
            }
            else
            {
                return 0;
            }
        }
```

```
            catch (Exception ex)
            {
                throw;
            }
            finally
            {
                if (con.State != ConnectionState.Closed)
                {
                    con.Close();
                }
            }
        }
    }
}
```

Create Class in BLL

- Now we need to create a class that act as a bridge between Presentation tier and Data access layer whose work is to pass the data from the presentation layer to data access layer for processing and after that getting sending the results back to the presentation layer.
- So right click on the "BLL_BookApp" in the solution explorer -> Add -> New Item -> Select "Class" and name it "BooksDetails_BLL.cs" as shown in image below.



Click on image to enlarge

Write the following code in **BooksDetails_BLL.cs** as:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using BEL_BookApp;
using DAL_BookApp;
```

```
namespace BLL_BookApp
{
  public class BookDetails_BLL
   {
      public Int32 SaveBookDetails(BooksDetails_BEL objBel)
      {
         BooksDetails_DAL objDal = new BooksDetails_DAL();
         try
         {
            return objDal.SaveBookDetails(objBel);
         }
         catch (Exception ex)
         {
            throw;
         }
         finally
         {
            objDal = null;
         }
      }

      public DataSet GetBookRecords()
      {
         BooksDetails_DAL objDal = new BooksDetails_DAL();
         try
         {
            return objDal.GetBookRecords();
         }
         catch (Exception ex)
         {
            throw;
         }
         finally
         {
            objDal = null;
         }
      }

      public Int32 DeleteBookRecord(BooksDetails_BEL objBel)
      {
         BooksDetails_DAL objDal = new BooksDetails_DAL();
         try
         {
                return objDal.DeleteBookRecord(objBel);
            }
            catch (Exception ex)
            {
                throw;
            }
            finally
            {
                objDal = null;
            }
        }

        public Int32 UpdateBookRecord(BooksDetails_BEL objBel)
        {
            BooksDetails_DAL objDal = new BooksDetails_DAL();
            try
            {
                return objDal.UpdateBookRecord(objBel);
            }
            catch (Exception ex)
            {
                throw;
            }
            finally
            {
                objDal = null;
            }
        }
    }
}
```

Create page in Presentation Tier

- Now let's design the front end i.e. the page in presentation tier from where user will input the book details.
- So right click on the "Presentation_BookApp" > Add -> Add New Item -> select Web Form and name it "bookdetails.aspx"
- In the <Form> tag of the Design page (.aspx) design the page using Html source code as:

```
<div>
    <fieldset style="width:470px">
    <legend>3 tier example to insert and bind book details</legend>
    <table>
        <tr><td>Book Name * : </td><td>
```

```
            <asp:TextBox ID="txtBookName" runat="server"></asp:TextBox><br />
            <asp:RequiredFieldValidator ID="rfvBookName" runat="server"
              ErrorMessage="Book Name can't be left
blank" ControlToValidate="txtBookName"
              Display="Dynamic" ForeColor="Red" SetFocusOnError="True">
</asp:RequiredFieldValidator>
        </td></tr>
        <tr><td>Author * : </td><td>
        <asp:TextBox ID="txtAuthor" runat="server"></asp:TextBox><br />
        <asp:RequiredFieldValidator ID="rfvAuthor" runat="server"
              ErrorMessage="Author Name can't be left
blank" ControlToValidate="txtAuthor"
              Display="Dynamic" ForeColor="Red" SetFocusOnError="True">
</asp:RequiredFieldValidator>
        </td></tr>
        <tr><td>Publisher * : </td><td>
        <asp:TextBox ID="txtPublisher" runat="server"></asp:TextBox><br />
        <asp:RequiredFieldValidator ID="rfvPublisher" runat="server"
              ErrorMessage="Publisher Name can't be left
blank" ControlToValidate="txtPublisher"
              Display="Dynamic" ForeColor="Red" SetFocusOnError="True">
</asp:RequiredFieldValidator>
        </td></tr>
        <tr><td>Price * : </td><td>
        <asp:TextBox ID="txtPrice" runat="server"></asp:TextBox><br />
        <asp:RequiredFieldValidator ID="rfvPrice" runat="server"
              ErrorMessage="Price can't be left blank" ControlToValidate="txtPrice"
              Display="Dynamic" ForeColor="Red" SetFocusOnError="True">
</asp:RequiredFieldValidator>
            <asp:RegularExpressionValidator ID="rgePrice" runat="server"
              ControlToValidate="txtPrice" Display="Dynamic"
              ErrorMessage="Enter Numeric
only" ForeColor="Red" SetFocusOnError="True"
              ValidationExpression="^\d*[0-9](|.\d*[0-9]|)*$">
</asp:RegularExpressionValidator>
        </td></tr>
        <tr><td></td><td>
            <asp:Button ID="btnSubmit" runat="server" Text="Submit"
              onclick="btnSubmit_Click" /></td></tr>
              <tr><td colspan="2">
<asp:Label ID="lblStatus" runat="server" Text=""></asp:Label></td></tr>
     </table>
        <br />
     <asp:GridView ID="grdBookDetails" runat="server" DataKeyNames="BookId"
```

```
        AutoGenerateColumns="False"
        onpageindexchanging="grdBookDetails_PageIndexChanging"
        onrowcancelingedit="grdBookDetails_RowCancelingEdit"
        onrowdeleting="grdBookDetails_RowDeleting"
        onrowediting="grdBookDetails_RowEditing"
        onrowupdating="grdBookDetails_RowUpdating" AllowPaging="True" PageSi
ze="5"
        CellPadding="4" ForeColor="#333333" GridLines="None">
        <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
    <Columns>
    <asp:TemplateField HeaderText="Book Name">
    <ItemTemplate>
        <asp:Label ID="lblBookName" runat="server" Text='<%#Eval("BookName"
)%>'></asp:Label>
    </ItemTemplate>
    <EditItemTemplate>
        <asp:TextBox ID="txtBookNameEdit" runat="server" Text='<%#Eval("Book
Name")%>'></asp:TextBox></EditItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Author">
    <ItemTemplate>
        <asp:Label ID="lblAuthor" runat="server" Text='<%#Eval("Author")%>'>
</asp:Label>
    </ItemTemplate>
    <EditItemTemplate>
        <asp:TextBox ID="txtAuthorEdit" runat="server" Text='<%#Eval("Author")
%>'></asp:TextBox>
    </EditItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Publisher">
    <ItemTemplate>
        <asp:Label ID="lblPublisher" runat="server" Text='<%#Eval("Publisher")%
>'></asp:Label>
    </ItemTemplate>
    <EditItemTemplate>
        <asp:TextBox ID="txtPublisherEdit" runat="server" Text='<%#Eval("Publis
her")%>'></asp:TextBox>
     </EditItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Price">
    <ItemTemplate>
        <asp:Label ID="lblPrice" runat="server" Text='<%#Eval("Price")%>'>
</asp:Label>
    </ItemTemplate>
    <EditItemTemplate>
       <asp:TextBox ID="txtPriceEdit" runat="server" Text='<%#Eval("Price")%>'
></asp:TextBox>
    </EditItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Edit" ItemStyle-HorizontalAlign="Center">
    <ItemTemplate>
       <asp:ImageButton ID="imgEdit" runat="server" ImageUrl="~/Images/edit.j
pg" CommandName="Edit" CausesValidation="false"/>
    </ItemTemplate>
    <EditItemTemplate>
    <asp:LinkButton ID="lnkUpdate" runat="server" Text="Update" CommandNa
me="Update" CausesValidation="false"></asp:LinkButton>
    <asp:LinkButton ID="lnkCancel" runat="server" Text="Cancel" CommandNam
e="Cancel" CausesValidation="false"></asp:LinkButton>
    </EditItemTemplate>
<ItemStyle HorizontalAlign="Center"></ItemStyle>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Delete" ItemStyle-
HorizontalAlign="Center">
    <ItemTemplate>
       <asp:ImageButton ID="imgDelete" runat="server" ImageUrl="~/Images/del
ete.jpg" CommandName="Delete" CausesValidation="false" onclientclick="retur
n confirm('Are you sure you want to delete?')" />
    </ItemTemplate>
    <EditItemTemplate>
    </EditItemTemplate>
<ItemStyle HorizontalAlign="Center"></ItemStyle>
    </asp:TemplateField>
    </Columns>
       <EditRowStyle BackColor="#999999" />
       <FooterStyle BackColor="#5D7B9D" Font-
Bold="True" ForeColor="White" />
       <HeaderStyle BackColor="#5D7B9D" Font-
Bold="True" ForeColor="White" />
       <PagerStyle BackColor="#284775" ForeColor="White" HorizontalAlign="C
enter" />
       <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
       <SelectedRowStyle BackColor="#E2DED6" Font-
Bold="True" ForeColor="#333333" />
       <SortedAscendingCellStyle BackColor="#E9E7E2" />
       <SortedAscendingHeaderStyle BackColor="#506C8C" />
       <SortedDescendingCellStyle BackColor="#FFFDF8" />
       <SortedDescendingHeaderStyle BackColor="#6F8DAE" />
    </asp:GridView>
    </fieldset>
    </div>
```

Note: Create a folder in root directory of the project " **Presentation_BookApp** " and name it " **Images** " and search icon images for edit and delete from Google and paste in this folder. These images will be used to display the edit and delete option in gridview.

Asp.Net C# Code

- In the code behind file (**bookdetails.cs**) write the code as:

```
#region "Namespaces"
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using BEL_BookApp;
using BLL_BookApp;
#endregion

namespace Presentation_BookApp
{
    public partial class bookdetails : System.Web.UI.Page
    {
        #region "Create and Initialize objects "
        BooksDetails_BEL objBookDetailsBEL = new BooksDetails_BEL();
        BookDetails_BLL objBookDetailsBLL = new BookDetails_BLL();
        #endregion

        #region "Bind Book Records on Page load Event"
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                BindBookRecordsGridView();
            }
        }
        #endregion

        #region "Save Book Record"
        protected void btnSubmit_Click(object sender, EventArgs e)
```

```
{
   objBookDetailsBEL.BookName = txtBookName.Text.Trim();
   objBookDetailsBEL.Author = txtAuthor.Text.Trim();
   objBookDetailsBEL.Publisher = txtPublisher.Text.Trim();
   objBookDetailsBEL.Price = Convert.ToDecimal(txtPrice.Text);
   try
   {
      int retVal = objBookDetailsBLL.SaveBookDetails(objBookDetailsBEL);
      if (retVal > 0)
      {
         lblStatus.Text = "Book detail saved successfully";
         lblStatus.ForeColor = System.Drawing.Color.Green;
         ClearControls();
         BindBookRecordsGridView();
      }
      else
      {
         lblStatus.Text = "Book details couldn't be saved";
         lblStatus.ForeColor = System.Drawing.Color.Red;
      }
   }
   catch (Exception ex)
   {
      Response.Write("Oops! error occured :" + ex.Message.ToString());
   }
   finally
   {
      objBookDetailsBEL = null;
      objBookDetailsBLL = null;
   }
}
#endregion

#region "Bind Book Records in GridView"
private void BindBookRecordsGridView()
{
   DataSet ds = new DataSet();
   try
   {
      ds = objBookDetailsBLL.GetBookRecords();
      if (ds.Tables[0].Rows.Count > 0)
      {
         grdBookDetails.DataSource = ds;
         grdBookDetails.DataBind();
```

```
                }
                else
                {
                    grdBookDetails.DataSource = null;
                    grdBookDetails.DataBind();
                }
            }
            catch (Exception ex)
            {
                Response.Write("Oops! error occured :" + ex.Message.ToString());
            }
            finally
            {
                objBookDetailsBEL = null;
                objBookDetailsBLL = null;
            }
        }
        #endregion

        #region "Edit and update Book Records"
        protected void grdBookDetails_RowEditing(object sender, GridViewEditEven
tArgs e)
        {
            grdBookDetails.EditIndex = e.NewEditIndex;
            BindBookRecordsGridView();
        }


    protected void grdBookDetails_RowCancelingEdit(object sender, GridViewCanc
elEditEventArgs e)
        {
            grdBookDetails.EditIndex = -1;
            BindBookRecordsGridView();
        }

        protected void grdBookDetails_RowUpdating(object sender, GridViewUpdate
EventArgs e)
        {
            objBookDetailsBEL.BookId
= Convert.ToInt32(grdBookDetails.DataKeys[e.RowIndex].Value);
            objBookDetailsBEL.BookName = ((TextBox)
(grdBookDetails.Rows[e.RowIndex].FindControl("txtBookNameEdit"))).Text.Trim()
;
            objBookDetailsBEL.Author = ((TextBox)
(grdBookDetails.Rows[e.RowIndex].FindControl("txtAuthorEdit"))).Text.Trim();
            objBookDetailsBEL.Publisher = ((TextBox)
(grdBookDetails.Rows[e.RowIndex].FindControl("txtPublisherEdit"))).Text.Trim();
            objBookDetailsBEL.Price = Convert.ToDecimal(((TextBox)
(grdBookDetails.Rows[e.RowIndex].FindControl("txtPriceEdit"))).Text.Trim());
            try
            {
                int retVal =
objBookDetailsBLL.UpdateBookRecord(objBookDetailsBEL);
                if (retVal > 0)
                {
                    lblStatus.Text = "Book detail updated successfully";
                    lblStatus.ForeColor = System.Drawing.Color.Green;
                    ClearControls();
                    grdBookDetails.EditIndex = -1;
                    BindBookRecordsGridView();
                }
                else
                {
                    lblStatus.Text = "Book details couldn't be updated";
                    lblStatus.ForeColor = System.Drawing.Color.Red;
                }
            }
            catch (Exception ex)
            {
                Response.Write("Oops! error occured :" + ex.Message.ToString());
            }
            finally
            {
                objBookDetailsBEL = null;
                objBookDetailsBLL = null;
            }
        }
        #endregion

        #region "Delete Book Record"
        protected void grdBookDetails_RowDeleting(object sender, GridViewDeleteE
ventArgs e)
        {
            int Book_Id
= Convert.ToInt32(grdBookDetails.DataKeys[e.RowIndex].Value);
            objBookDetailsBEL.BookId = Book_Id;
            try
            {
                int retVal = objBookDetailsBLL.DeleteBookRecord(objBookDetailsBEL);

                if (retVal > 0)
                {
                    lblStatus.Text = "Book detail deleted successfully";
                    lblStatus.ForeColor = System.Drawing.Color.Green;
                    ClearControls();
                    BindBookRecordsGridView();
                }
                else
                {
                    lblStatus.Text = "Book details couldn't be deleted";
                    lblStatus.ForeColor = System.Drawing.Color.Red;
                }
            }
            catch (Exception ex)
            {
                Response.Write("Oops! error occured :" + ex.Message.ToString());
            }
            finally
            {
                objBookDetailsBEL = null;
                objBookDetailsBLL = null;
            }
        }
        #endregion

        #region "Paging in GridView"
        protected void grdBookDetails_PageIndexChanging(object sender, GridView
PageEventArgs e)
        {
            grdBookDetails.PageIndex = e.NewPageIndex;
            BindBookRecordsGridView();
        }
        #endregion

        #region "Clear/Reset controls "
        protected void btnReset_Click(object sender, EventArgs e)
        {
            ClearControls();
        }
        private void ClearControls()
        {
            txtBookName.Text = string.Empty;
```

```
                txtAuthor.Text = string.Empty;
                txtPublisher.Text = string.Empty;
                txtPrice.Text = string.Empty;
                txtBookName.Focus();
            }
            #endregion
        }
    }
```

Download the complete example

Now over to you:

> *I hope you have learned how to create 3-tier architecture project with this example and If you like my work; you can appreciate by leaving your comments, hitting Facebook like button, following on Google+, Twitter, Linked in and Pinterest, stumbling my posts on stumble upon and subscribing for receiving free updates directly to your inbox . Stay tuned and stay connected for more technical updates.*