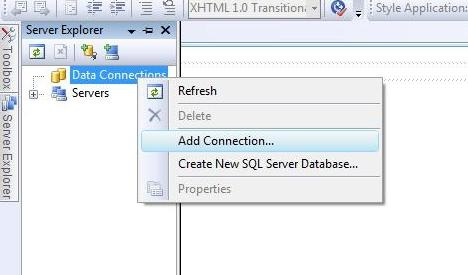
**1. GridView Control**

The GridView control is used to provide flexibility in working and display data from the database. The data in the GridView control is displayed in the tabular format. It has several properties assigned to the control. Some of the properties are as mentioned below:

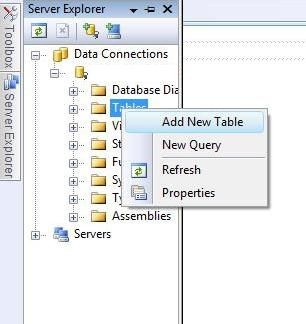
|  |  |
| --- | --- |
| **Property** | **Description** |
| AllowPaging | It is a Boolean value indicating the control supports paging |
| AllowSorting | It is a Boolean value indicating the control supports sorting |
| SortExpression | It accepts the current expression determining the order of the row |
| Datasource | It is used to get or set the data source object containing the data to populate the control |
| AutoGenerateEditButton | It is a Boolean value indicating that the user can edit the record in the control |
| DataSourceID | It indicates the data source control to be used |
| AutoGenerateDeleteButton | It is a Boolean value indicating that the user can delete the record in the control |
| AutoGenerateColumns | It is a Boolean value to indicate the columns are automatically created for each field of the data source |
| AutoGenerateSelectButton | It is a Boolean value to indicate the column should be added to select the particular record |
| SortDirection | It gets the sorting direction of the column for the control |
| EmptyDataText | It indicates the text to appear when there is no record in the data source |

The code sample of the GridView control is as shown below:

1) Add a new connection object to the ASP.NET web application as shown below:

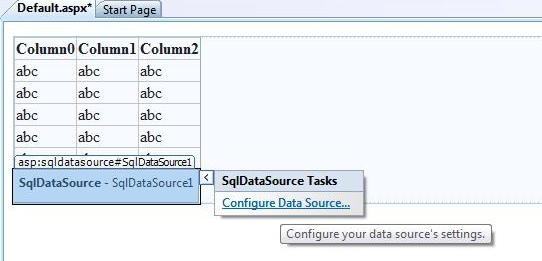


2) Next, add a new table for the connection object created above. The snippet for adding the table is as shown below:



3) Add the fields Sno, Name, Address in the table. Add values to the respective fields in the table

4) Add the GridView and SqlDataSource control to the design view of the web page.



5) The source code for the GridView control is as shown below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <%@Page Language=”C#” AutoEventWireup=”true” CodeFile=”binding.aspx.cs” Inherits=”binding” %>    <!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN”  “http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>  <html xmlns=”http://www.w3.org/1999/xhtml”>   <head runat=”server”>       <title></title>   </head>   <body>      <form id=”form1” runat=”server” >        <asp:Button ID=”Button1” runat=”server” onclick=”Button1\_Click” Text=”GetData” Width=”123px” />        <br/>        <div>       <asp:GridView ID=”GridView1” runat=”server”>       </asp:GridView>        </div>     </form>   </body>  </html> |

6) The code behind file contains the following code?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | protected void Button1\_Click(object sender, EventArgs e)  {    SqlConnection con = new SqlConnection();    con.ConnectionString = ConfigurationManager.ConnectionStrings    [ “ConnectionString” ].ToString();      con.Open();      SqlCommand cmd = new SqlCommand();    cmd.CommandText = “Select \* from deltable”;    cmd.Connection = con;    DataSet ds = new DataSet();    da.Fill( ds, “deltable”);    GridView1.DataSource= ds;    GridView1.DataBind();  } |

7) **The output is:**



**2. DetailsView control**

Details view control is used as a data bound control. It is used to render one record at a time. User can insert, update and delete the record from the control. Some of the properties of the DetailsView control is as shown below:

|  |  |
| --- | --- |
| **Property** | **Description** |
| AllowPaging | It is a Boolean value to indicate the control supports navigation |
| DataSource | It is used to populate the control with the data source object |
| DataSourceID | It indicates the bound data source control with corresponding adapters |
| AutoGenerateEditButton | It is a Boolean value to indicate the column can be edited by the user |
| AutoGenerateDeleteButton | It is a Boolean value to indicate the records can be deleted |
| DefaultMode | It is used to indicate the default display mode of the control |
| AutoGenerateRows | It is a Boolean value to indicate the rows are automatically created for each field in the data source |

The sample code for the Details View control is as shown below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | <%@ Page Language=”C#” %>    <!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional //EN”  “http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd” >    <script runat=”server”>  </script>    <html xmlns=”http://www.w3.org/1999/xhtml”>   <head runat=”server”>      <title>asp.net DetailsView example: how to use DetailsView</title>   </head>   <body>      <form id=”form1” runat=”server” >      <div>      <h2 style=”color:Navy” >DetailsView Example</h2>      <asp:SqlDataSource ID=”SqlDataSource1” runat=”server” ConnectionString=”<%$ ConnectionStrings:  NorthwindConnectionString %>”      Select Command = “SELECT ProductID, ProductName, UnitPrice FROM Products”;      </asp:SqlDataSource>      <asp:DetailsView ID=”DetailsView” runat=”server” DataSourceID=”SqlDataSource1” AllowPaging=”true”  ForeColor=”DarkGreen” BackColor=”Snow” BorderColor=”Tomato”>      </asp:DetailsView>      </div>      </form>   </body>  </html> |

**The output is:**



**3. Data List Control**

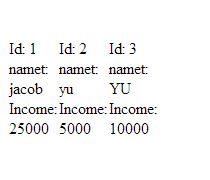
The DataList control is used to display a repeated list of items that are bound to the control. There are different templates using which user can design the layout of the control. The template properties are mentioned below:

|  |  |
| --- | --- |
| **Template Property** | **Description** |
| ItemTemplate | It contains the HTML elements and controls to render for each row in the data source |
| AlternatingItemTemplate | It contains the HTML elements and controls to render once for every other row in the data source |
| SelectedItemTemplate | It contains the elements to render when the user selects an item in the DataList control |
| EditItemTemplate | It specifies the layout of an item when the edit mode is working |
| HeaderTemplate | It contains all the text and controls to be rendered at the beginning of the list |
| FooterTemplate | It contains all the text and controls to be rendered at the end of the list |
| SeperatorTemplate | It contains all elements to render between each item |

A sample code to demonstrate the DataList control is as shown below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | <%@Page Language=”C#” %>  <! DOCTYPE html>  <script runat=”server”>  </script>  <html xmlns=”http://www.w3.org/1999/xhtml”>   <head runat=”server”>     <title></title>   </head>   <body>     <form id=”form1” runat=”server”>       <div>       <asp:DataList ID=”DataList1” runat=”server” DataKeyField=”Id” DataSourceID=”SqlDataSource1” Height=”285px”  RepeatColumns=”3” RepeatDirection=”Horizontal” Width=”134px”>      <ItemTemplate>      Id:      <asp:Label ID=”IdLabel” runat=”server” Text=’<%# Eval ( “Id” )%>’ />      <br/>      name:      <asp:Label ID=”nameLabel” runat=”server” Text=’<%# Eval ( “name” )%>’ />      <br/>      Income:      <asp:Label ID=”IncomeLabel” runat=”server” Text=’<%# Eval ( “Income” )%>’ />      <br/>      <br/>         </ItemTemplate>         </asp:DataList>         <asp:SqlDataSource ID=”SqlDataSource1” runat=”server” ConnectionString=’<%$ConnectionStrings:ConnectionString %>”         SelectCommand=”SELECT \* FROM [footerex]”>         </asp:SqlDataSource>       </div>     </form>   </body>  </html> |
|  |  |

**Output is:**



**4. Repeater Control**

The Repeater control is a data bound control created by using the templates to display items. The control does not support editing, paging or sorting of data rendered through the control. The list of templates supported by the Repeater control is as mentioned below:

|  |  |
| --- | --- |
| **Templates** | **Description** |
| HeaderTemplate | It contains all the text and controls to be rendered at the beginning of the list |
| FooterTemplate | It contains all the text and controls to be rendered at the end of the list |
| AlternatingItemTemplate | It contains the HTML elements and controls to render once for every other row in the data source |
| SeperatorTemplate | It contains all elements to render between each item |
| ItemTemplate | It contains the HTML elements and controls to render for each row in the data source |

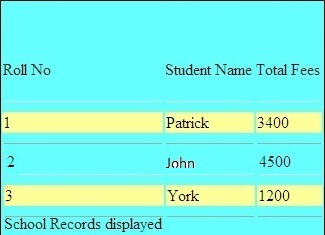
The sample code for the Repeater control is as shown below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | <body>    <form id=”form1” runat=”server”>      <div>        <asp:Repeater ID=”RepeaterInformation” runat=”server”>      <HeaderTemplate>          <table class=”tblcolor”>          <tr>              <b>              <td>              Roll No              </td>              <td>              StudentName              </td>              <td>              Total Fees              </td>              </b>          </HeaderTemplate>          </ItemTemplate>          <tr class=”tblrowcolor”>          <td>              <%#DataBinder.Eval ( Contiane.”DataItem.RollNo”)%>          </td>          <td>              <%#DataBinder.Eval(Contianer,”DataItem.Name”)%>          </td>          <td>              <%#DataBinder.Eval(Contianer.”DataItem.Fees”)%>          </td>          </tr>          </ItemTemplate>          <SeperatorTemplate>          <tr>          <td>              <hr/>          </td>          <td>              <hr/>          </td>          <td>              <hr/>          </td>          </tr>          </SeperatorTemplate>          <FooterTemplate>          <tr>          <td>              School Records displayed          </td>          </tr>          </table>          </FooterTemplate>      </asp:Repeater>       </div>    </form>  </body> |

The code behind file is as shown below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public partial class \_Default: System.Web.UI.Page  {    SqlConnection con;    SqlCommand cmd = new SqlCommand();    protected void Page\_Load( object sender, EventArgs e)    {      con=new SqlConnection( ConfigurationManager.ConnectionStrings[“constr”].ConnectionString);      cmd.Connection=con;      com.Open();      RepeatInformation.DataSource = cmd.ExecuteReader();      RepeatInformation.DataBind();      con.Close();    }  } |

Output is :



**5. Formatting DataList Control**

The benefit of the [DataList](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datalist(v=vs.100).aspx)control is that you can create a free-form layout for the data. In this section, you will work with a template and configure it with text and controls to customize the data display.

### To format the layout in the DataList control

1. Right-click the DataList control, click Edit Template, and then click Item Templates.

The DataList control switches to template editing mode and displays template boxes for these templates:

* + ItemTemplate, which contains the text and controls that are displayed by default in the DataList control.
  + AlternatingItemTemplate, which is an optional template in which you can create a layout that is used for every other data record. Typically, the [AlternatingItemTemplate](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datalist.alternatingitemtemplate(v=vs.100).aspx) property is similar to the [ItemTemplate](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datalist.itemtemplate(v=vs.100).aspx) property, but the [AlternatingItemTemplate](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datalist.alternatingitemtemplate(v=vs.100).aspx) property uses a different background color for a banded effect.
  + SelectedItemTemplate, which defines the layout for a data record that is explicitly selected using a button click or other gesture. Typical uses for this template are to provide an expanded view of a data record or to serve as the master record for a master/detail relationship. You must write code to support putting a record into selected mode. (You will not do so in this walkthrough. For more information, see [How to: Allow Users to Select Items in DataList Web Server Controls](https://msdn.microsoft.com/en-us/library/75670ez0(v=vs.100).aspx) and [Walkthrough: Creating Master/Detail Web Pages in Visual Studio](https://msdn.microsoft.com/en-us/library/stc0szde(v=vs.100).aspx).)
  + EditItemTemplate, which defines the layout for edit mode for a data record. Typically, the [EditItemTemplate](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datalist.edititemtemplate(v=vs.100).aspx) property includes editable controls, such as the [TextBox](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.textbox(v=vs.100).aspx) and [CheckBox](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.checkbox(v=vs.100).aspx) controls in which users can modify the data record.

By default, Visual Web Developer populates the item template with a data-bound [Label](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.label(v=vs.100).aspx) control for each data column in the data source. In addition, Visual Web Developer generates static text for each label to act as a caption.

1. Drag the right-hand resize handle to widen the DataList control so it takes up most of the page width.
2. Edit the item template to rearrange the [Label](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.label(v=vs.100).aspx) controls and create a new caption so that the template contents look similar to the following code example.
3. Name: [CategoryNameLabel] (ID: [CategoryIDLabel])
4. Click the CategoryNameLabel control. In Properties, expand the Font node, and then set Bold to true.
5. In Design view, on the DataList control, right-click the title bar, click Edit Template, and then click Separator Template.

The separator template allows you to specify what text or other elements are displayed between data records.

1. In the Toolbox, from the HTML group, drag a Horizontal Rule element into the separator template.
2. On the DataList control, right-click the title bar, and then click End Template Editing.

**6. how to create and call stored procedures in .net using ADO.Net**

**Connection String**

<connectionStrings>

      <add name="conString"

        connectionString="Data Source=.\SQLEXPRESS;

                          database=Northwind;Integrated Security=true"/>

connectionStrings>

**Namespaces**

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

**Select Stored Procedure**

A simple Select Stored Procedure that brings all the records from the Employees table

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[GetAllEmployeeDetails]

AS

BEGIN

      SET NOCOUNT ON;

      SELECT FirstName, LastName, BirthDate, City, Country

      FROM Employees

END

Now we ll call the stored procedure using ADO.Net in my ASP.Net website and bind the results to a GridView. Since we need to fetch multiple rows using ExecuteReader method of SQL Command object:

String strConnString =  ConfigurationManager.ConnectionStrings["conString"].ConnectionString;

SqlConnection con =  new SqlConnection(strConnString);

SqlCommand cmd = new SqlCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "GetAllEmployeeDetails";

cmd.Connection = con;

try

{

    con.Open();

    GridView1.EmptyDataText = "No Records Found";

    GridView1.DataSource = cmd.ExecuteReader() ;

    GridView1.DataBind();

}

catch (Exception ex)

{

    throw ex;

}

finally

{

    con.Close();

    con.Dispose();

}