Hands On ASP.NET GridView

By: Abhimanyu Kumar Vatsa

Hands on ASP.NET GridView

Abhimanyu Kumar Vatsa
Microsoft & C# Corner MVP,
Blogs at ITORIAN.COM

Sam Hobbs
Editor, C# Corner

This book is a basic introduction to **ASP.NET GridView** basically for beginners who want to learn complete basic with example of ASP.NET GridView.

Hands on ASP.NET GridView

Table of Contents

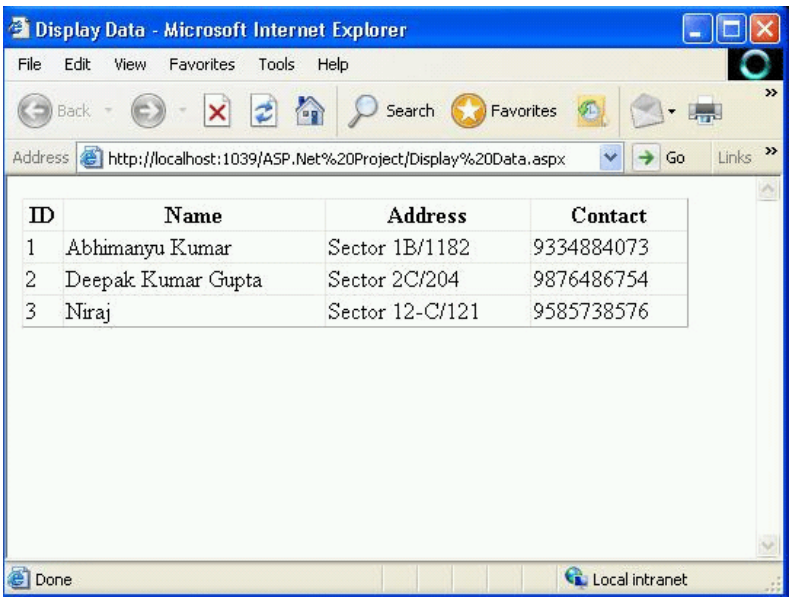
1 Fundamentals of GridView Control

If we have data in database we can display it in Data Display controls and also can edit it. There are couples of Data Display and Edit Controls exist in ASP.Net but we will discuss on GridView here.

The GridView control is one of the most useful controls in ASP.NET. The GridView control enables us to display, select, sort, page, and edit data items such as database records.

1.1 Displaying Data

GridView control renders data items in an HTML table. Each data item is rendered in a distinct HTML table row. For example, given code will demonstrates how we use the GridView to display the contents of the database table.




```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Display Data.aspx.vb"Inherits="Display_Data" %>

<!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>Display Data</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
        DataSourceID="SqlDataSource1"
        EmptyDataText="There are no data records to display." Width="458px">
        <Columns>
          <asp:BoundField DataField="ID" HeaderText="ID" SortExpression="ID" />
          <asp:BoundField DataField="Name" HeaderText="Name" SortExpression="Name" />
          <asp:BoundField DataField="Address" HeaderText="Address"
            SortExpression="Address" />
          <asp:BoundField DataField="Contact" HeaderText="Contact"
            SortExpression="Contact" />
```

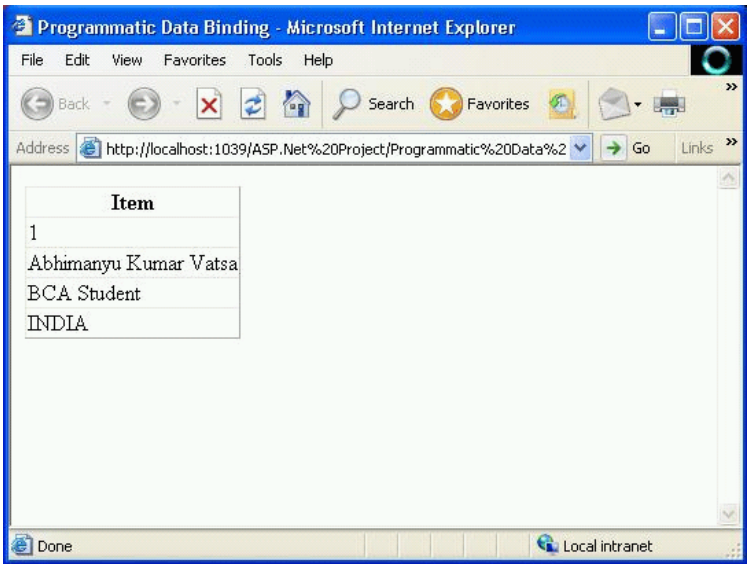
```
        </Columns>
      </asp:GridView>
      <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
        ProviderName="<%$ ConnectionStrings:DatabaseConnectionString1.ProviderName %>"
        SelectCommand="SELECT [ID], [Name], [Address], [Contact] FROM [MyTB]">
      </asp:SqlDataSource>

    </div>
  </form>
</body>
</html>
```

In above code, the GridView control is bound to a SqlDataSource control, which represents the database table named MyTB. The GridView is associated with its data source through its DataSourceID property.

We can add a GridView and SqlDataSource control to a page quickly by dragging a database table from the Database Server Explorer onto a page in Design view. It automatically creates SqlDataSource, which retrieves all the rows and all the columns from database table.

1.2 Programmatic Data Binding

GridView control also supports programmatic data-binding. In programmatic data-binding it lists Generic collections. Notice that the GridView is bound to my info list in the Page Load () method. Its DataSource property points to the list collection, and its DataBind () method is called to load the items from the list collection and display them.

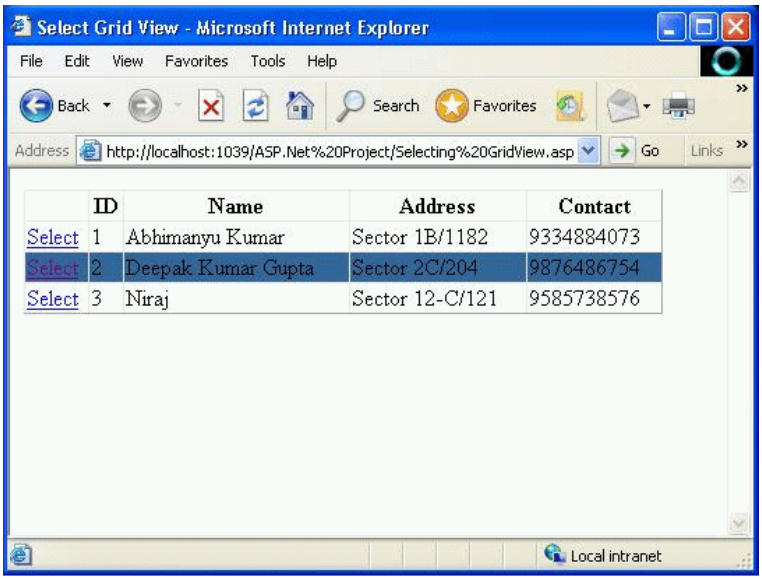



```
<%@ Page Language="VB" %>

<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
  Sub Page Load()
    ' Building list item
    Dim myinfo As New List(Of String)()
    myinfo.Add("1")
```

```
        myinfo.Add("Abhimanyu Kumar Vatsa")
        myinfo.Add("BCA Student")
        myinfo.Add("INDIA")
        ' Binding to GridView
        GridView1.DataSource = myinfo
        GridView1.DataBind()
    End Sub
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Programmatic Data Binding</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

        <asp:GridView ID="GridView1" runat="server">
        </asp:GridView>

      </div>
    </form>
</body>
</html>
```

1.3 Selecting GridView

We can enable a user to select a particular row in a GridView control. This is useful when we want to build single page form for huge data. We can apply some different property to selected row to look different than other when selected.




```
<%@ Page Language="VB" %>

<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
</script>
```

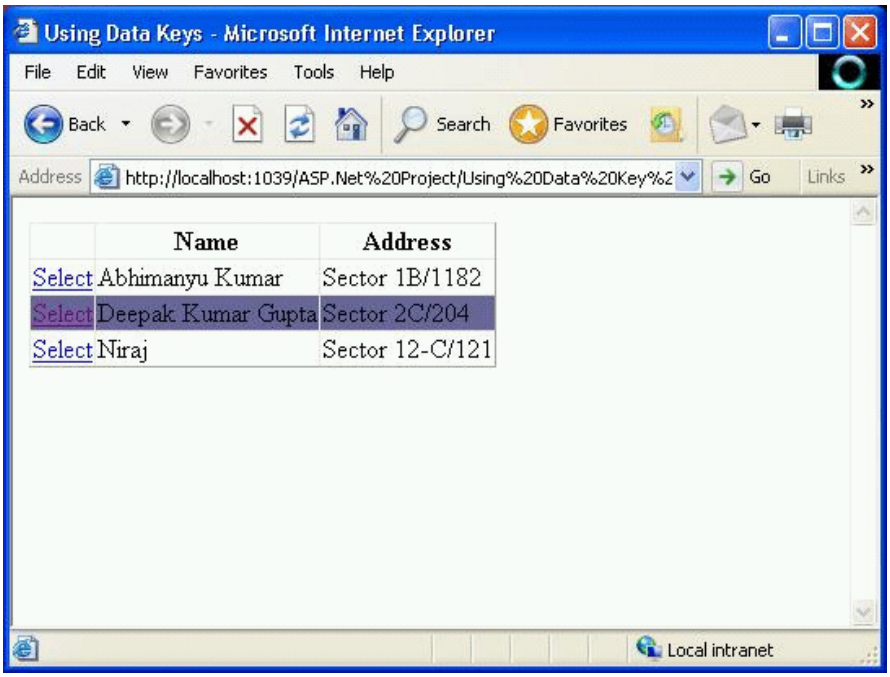
```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
   <title>Select GridView</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>

        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
        DataSourceID="SqlDataSource1"
        EmptyDataText="There are no data records to display." Width="458px">
        <Columns>
        <asp:CommandField ShowSelectButton="True"/>
        <asp:BoundField DataField="ID" HeaderText="ID" SortExpression="ID" />
        <asp:BoundField DataField="Name" HeaderText="Name" SortExpression="Name" />
        <asp:BoundField DataField="Address" HeaderText="Address"
             SortExpression="Address" />
        <asp:BoundField DataField="Contact" HeaderText="Contact"
             SortExpression="Contact" />
        </Columns>
        <SelectedRowStyle BackColor="#336699" BorderColor="Blue" />
      </asp:GridView>
      <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
        ProviderName="<%$ ConnectionStrings:DatabaseConnectionString1.ProviderName %>"
        SelectCommand="SELECT [ID], [Name], [Address], [Contact] FROM [MyTB]">
      </asp:SqlDataSource>
     </div>
   </form>
</body>
</html>
```

We can also enable the GridView selection using the following properties:

- **SelectedDataKey:** Returns the DataKey object associated with the selected row (it is useful when there are multiple data key).
- **SelectedIndex:** Returns the index of the selected row.
- **SelectedValue:** Returns the data key associated with the selected row.
- **SelectedRow:** Returns the actual row that is GridViewRow object associated with the selected row.

1.4 Using Data Keys Name

We associate a value with each row in a GridView by providing a value for the GridView control's DataKeyName property. We can assign the name of a single database column to this property or we can assign a comma-separated list of column names to this property.

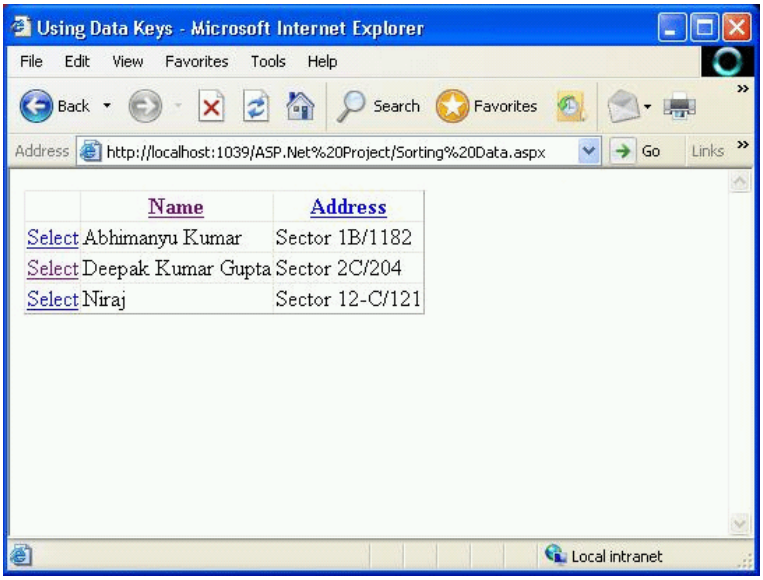



```
<%@ Page Language="VB" %>

<! 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>Using Data Keys</title>
   <style type="text/css">

   </style>
</head>
<body>
   <form id="form1" runat="server">
   <div>

      <asp:GridView ID="GridView1"
      runat="server"
      DataSourceID="SqlDataSource1"
      DataKeyNames="Name,Address"
      AutoGenerateSelectButton="true">
      <SelectedRowStyle CssClass="mycss" BackColor="#666699" BorderColor="Red">
      </SelectedRowStyle>
      </asp:GridView>
      <br />

      <br />
      <asp:SqlDataSource ID="SqlDataSource1" runat="server"
```

```
        ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
        SelectCommand="SELECT Name, Address FROM MyTB">
      </asp:SqlDataSource>
      <br />

    </div>
  </form>
</body>
</html>
```

1.5 Shorting Data

We can sort the rows rendered by a GridView control by enabling the AllowSorting property.



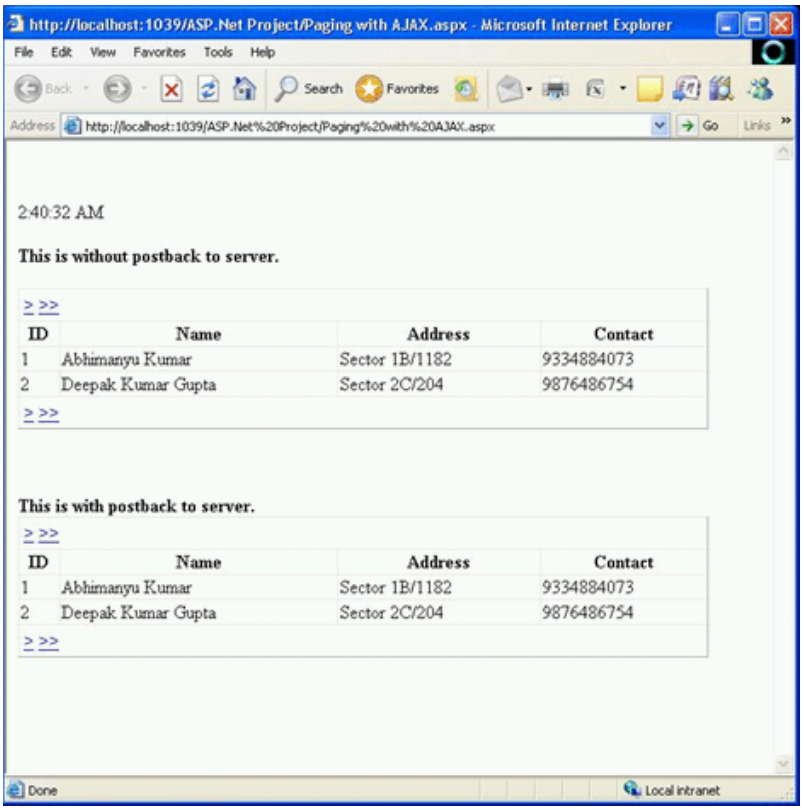

```
<%@ Page Language="VB" %>

<!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 id="Head1" runat="server">
  <title></title>
  <style type="text/css">

  </style>
</head>
<body>
  <form id="form1" runat="server">
  <div>
```

```
    <asp:GridView ID="GridView1"
    runat="server"
    DataSourceID="SqlDataSource1"
    DataKeyNames="Name,Address"
    AutoGenerateSelectButton="true"
    AllowSorting="true">
    <SelectedRowStyle CssClass="mycss" BackColor="#666699" BorderColor="Red">
    </SelectedRowStyle>
    </asp:GridView>
    <br />

    <br />
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
      SelectCommand="SELECT Name, Address FROM MyTB">
    </asp:SqlDataSource>
    <br />

  </div>
 </form>
</body>
</html>
```

Note: We can do this manually using property 'AllowSorting' to true.

1.6 Shorting with Ajax

By default, whenever we click column header to sort the rows contained in a GridView, the page containing the GridView is posted back to the server. When sorting records with the GridView control, we can avoid posting the entire page back to the server by taking advantage of AJAX (Asynchronous JavaScript and XML).




```
<%@ Page Language="VB" %>

<!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 id="Head1" runat="server">
   <title>Using Data Keys</title>
   <style type="text/css">

   </style>
</head>
<body>
   <form id="form1" runat="server">
   <div>
   <div><b>Check here the time while clicking to sort.</b></div>
   <b><%= DateTime.Now.ToString("T") %></b>
   <br />
   <br />
   </div>
   <div>
   <div>
   <br />
      <b>This is with POST BACK</b>
   </div>
```

```
<br />
    <asp:GridView ID="GridView1"
    runat="server"
    DataSourceID="SqlDataSource1"
    DataKeyNames="Name,Address"
    AutoGenerateSelectButton="true"
    AllowSorting="true">
    </asp:GridView>
    <br />
    <div>
    <br />
    <br />
      <b>This is without POST BACK</b>
    </div>
    <br />
    <asp:GridView ID="GridView2"
    runat="server"
    DataSourceID="SqlDataSource1"
    DataKeyNames="Name,Address"
    EnableSortingAndPagingCallbacks="true"
    AllowSorting="true">
    </asp:GridView>
    <br />
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
      SelectCommand="SELECT Name, Address FROM MyTB">
    </asp:SqlDataSource>
    <br />

  </div>
 </form>
</body>
</html>
```

This page also displays the current time at the top of the page. Notice that the time is not updated when we sort the records in the GridView. The entire page is not posted back to the server; only the content of the GridView control is updated. When using AJAX with the GridView control, you cannot use Template Fields. Furthermore, you cannot display a Select button when AJAX is enabled.

1.7 Using Image in Sorting

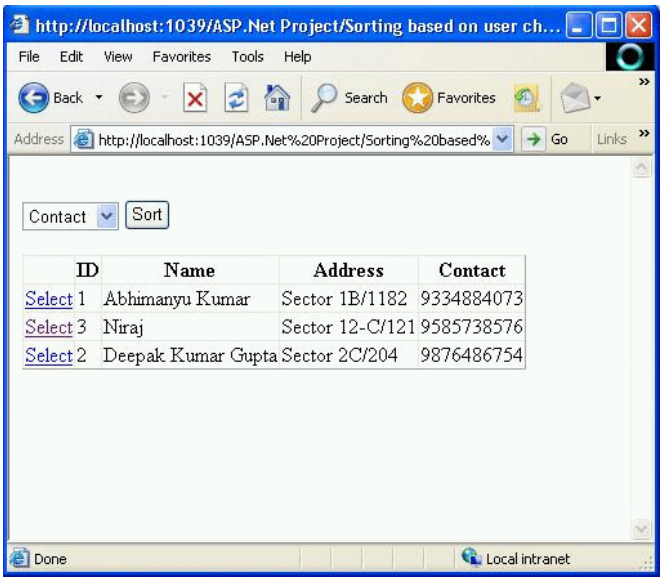
We can customize the appearance of the sort links by handling the GridView control's RowDataBound event. This event is raised for each row rendered by the GridView after the GridView is bound to its data source. In the example given below I have displayed an image that represent whether a column is sorted in ascending or descending order.

```
<%@ Page Language="VB" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
  Protected Sub image_RowDataBound(ByVal sender As Object, ByVal e AsGridViewRowEventArgs)
    If e.Row.RowType = DataControlRowType.Header Then
      For Each cell As TableCell In e.Row.Cells
        Dim sortLink As LinkButton = CType(cell.Controls(0), LinkButton)
        If sortLink.Text = GridView1.SortExpression Then
```

```
            If GridView1.SortDirection = SortDirection.Ascending Then
               sortLink.Text += " <img src='asc.GIF' title='Sort ascending' />"
            Else
               sortLink.Text += " <img src='desc.GIF' title='Sort descending' />"
            End If
         End If
      Next
   End If
End Sub
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>

      <asp:GridView ID="GridView1"
      runat="server"
      DataSourceID="SqlDataSource1"
      DataKeyNames="Name,Address"
      AutoGenerateSelectButton="true"
      AllowSorting="true"
      OnRowDataBound="image_RowDataBound">
      <SelectedRowStyle
      CssClass="mycss"
      BackColor="#666699"
      BorderColor="Red">
      </SelectedRowStyle>
      </asp:GridView>
      <br />
      <asp:SqlDataSource ID="SqlDataSource1" runat="server"
         ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
         SelectCommand="SELECT Name, Address FROM MyTB">
      </asp:SqlDataSource>
      <br />

   </div>
   </form>
</body>
</html>
```

1.8 Sorting based on user choice

If we need to completely customize the appearance of the sorting user interface, then we can call the GridView control's Sort() method programmatically.



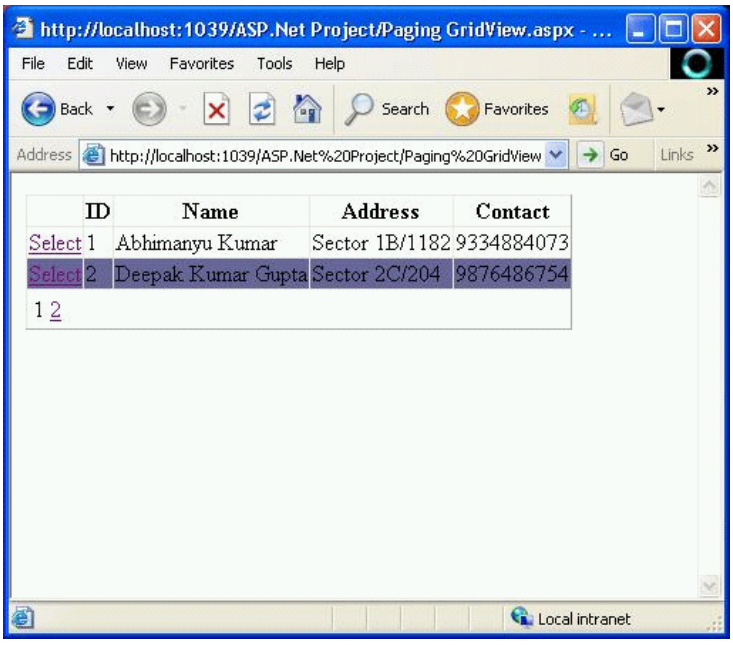

```
<%@ Page Language="VB" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

    Protected Sub btnSort_Click(ByVal sender As Object, ByVal e As System.EventArgs)
        GridView1.Sort(ddlSort.Text, SortDirection.Ascending)
    End Sub
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title></title>
    <style type="text/css">

    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <div>
    <br />
    <asp:DropDownList
        id="ddlSort"
        Runat="server">
        <asp:ListItem Text="ID" />
        <asp:ListItem Text="Name" />
        <asp:ListItem Text="Address" />
        <asp:ListItem Text="Contact" />
    </asp:DropDownList>
    <asp:Button
        id="btnSort"
        Text="Sort"
        Runat="server" onclick="btnSort_Click" />
```

```
<br />
<br />
</div>
    <asp:GridView ID="GridView1"
    runat="server"
    DataSourceID="SqlDataSource1"
    AutoGenerateSelectButton="true">
    <SelectedRowStyle
    CssClass="mycss"
    BackColor="#666699"
    BorderColor="Red">
    </SelectedRowStyle>
    </asp:GridView>
    <br />
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
      SelectCommand="SELECT * FROM MyTB">
    </asp:SqlDataSource>
    <br />

</div>
</form>
</body>
</html>
```

Above code includes a DropDownList control, which we can use to sort the contents of the GridView. When a list item is selected from the DropDownList control and the Sort button is clicked and the btnSort_Click() method executes. This method calls the Sort() method of the GridView control to sort the contents of the GridView.

2 Paging in GridView

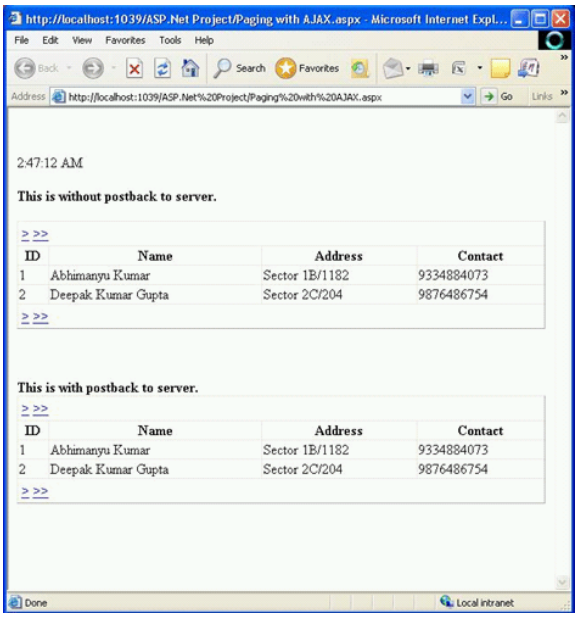
If we have large amount of data in GridView then we need to display the rows in different pages. We can enable paging with the GridView control by enabling its AllowPaging property. We can also determine how many rows want to see in one page by setting the PageSize property.



```
<%@ Page Language="VB" %>

<!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 id="Head1" runat="server">
   <title></title>
   <style type="text/css">

   </style>
</head>
<body>
   <form id="form1" runat="server" >
   <div>
      <asp:GridView ID="GridView1"
      runat="server"
      DataSourceID="SqlDataSource1"
      AutoGenerateSelectButton="true"
      PageSize="2"
      AllowPaging="true">

      <SelectedRowStyle
      CssClass="mycss"
      BackColor="#666699"
      BorderColor="Red">
      </SelectedRowStyle>

      </asp:GridView>

      <br />
      <asp:SqlDataSource ID="SqlDataSource1" runat="server"
         ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
         SelectCommand="SELECT * FROM MyTB">
      </asp:SqlDataSource>
      <br />

   </div>
   </form>
</body>
</html>
```

2.1 Paging with AJAX

The default behavior of the GridView control is to post back to the server each and every time we navigate to a new page of records. However, there is an alternative of AJAX (Asynchronous JavaScript and XML) when paging through records with the GridView control.




```
<%@ Page Language="VB" %>

<!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 id="Head1" runat="server">
   <title>Using Data Keys</title>
   <style type="text/css">

   </style>
</head>
<body>
      <form id="form1" runat="server" >
      <div>
      <div>
      <br />
      <br />
      <%= DateTime.Now.ToString("T") %>
      <br />
      <br />
      <b>This is without postback to server.</b>
      <br />
      <br />
      </div>
      <asp:GridView ID="GridView1"
      runat="server"
      DataSourceID="SqlDataSource1"
      PageSize="2"
      AllowPaging="true"
```

```
    PagerSettings-Mode="NextPreviousFirstLast"
    PagerSettings-Position="TopAndBottom"
    PagerStyle-HorizontalAlign="Left"
    EnableSortingAndPagingCallbacks="true"
    Width="603px">
    </asp:GridView>
    <br />
    <br />
    <br />
    <b>This is with postback to server.</b>
    <br />
    <asp:GridView ID="GridView2"
    runat="server"
    DataSourceID="SqlDataSource1"
    PageSize="2"
    AllowPaging="true"
    PagerSettings-Mode="NextPreviousFirstLast"
    PagerSettings-Position="TopAndBottom"
    PagerStyle-HorizontalAlign="Left"
    Width="603px">
    </asp:GridView>
    <br />
    <br />
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
      SelectCommand="SELECT * FROM MyTB">
    </asp:SqlDataSource>
    <br />

  </div>
  </form>
</body>
</html>
```

In above example, I have used two GridView controls but first one is with AJAX and second one is without AJAX. I have also used some different navigation item in above example.

2.2 Using Pager Templates

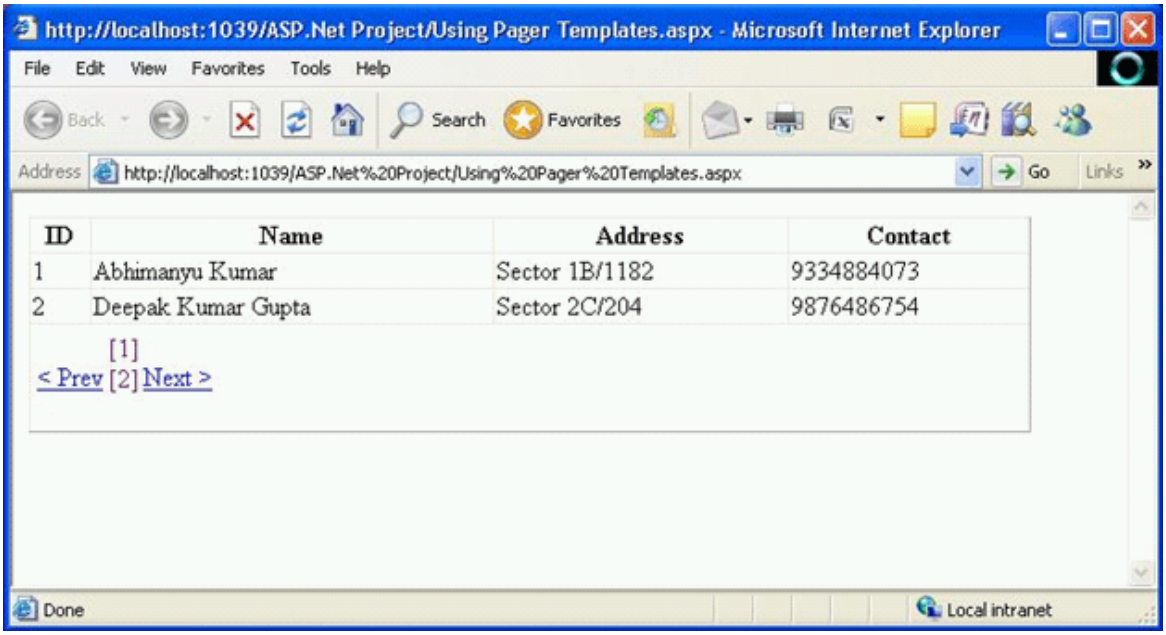
By default, when paging is enabled, the GridView renders a list of page numbers at the bottom of the grid. We can modify the user interface for paging through records by modifying the GridView control's PagerSettings property. The PagerSettings class supports the following properties:

- FirstPageImageUrl Enables us to display an image for the first page link.
- FirstPageText Enables us to specify the text for the first page link.
- LastPageImageUrl Enables us to display an image for the last page link.
- LastPageText Enables us to specify the text for the last page link.
- Mode Enables us to select a display mode for the pager user interface. Possible values are NextPrevious, NextPreviousFirstLast, Numeric, and NumericFirstLast.
- NextPageImageUrl Enables us to display an image for the next page link.
- NextPageText Enables us to specify the text for the next page link.
- PageButtonCount Enables us to specify the number of page number links to display.
- Position Enables us to specify the position of the paging user interface. Possible values are Bottom, Top, TopAndBottom.
- PreviousPageImageUrl Enables us to display an image for the previous page link.
- PreviousPageText Enables us to specify the text for the previous page link.

- Visible Enables us to hide the paging user interface.

The PageButtonCount requires more explanation. Imagine that we are displaying the contents of a database table that contains 3 billion records and we are displaying two records per page. In that case, we would need to render an overwhelming number of page numbers. The PageButtonCount property enables we to limit the number of page numbers displayed at once. When PageButtonCount has a value less than the number of page numbers, the GridView renders ellipsis, which enables a user to move between ranges of page numbers.

The GridView control includes a PagerTemplate, which enables us to completely customize the appearance of the paging user interface. For example, the PagerTemplate also includes two LinkButton controls, which represent a Previous and Next link.



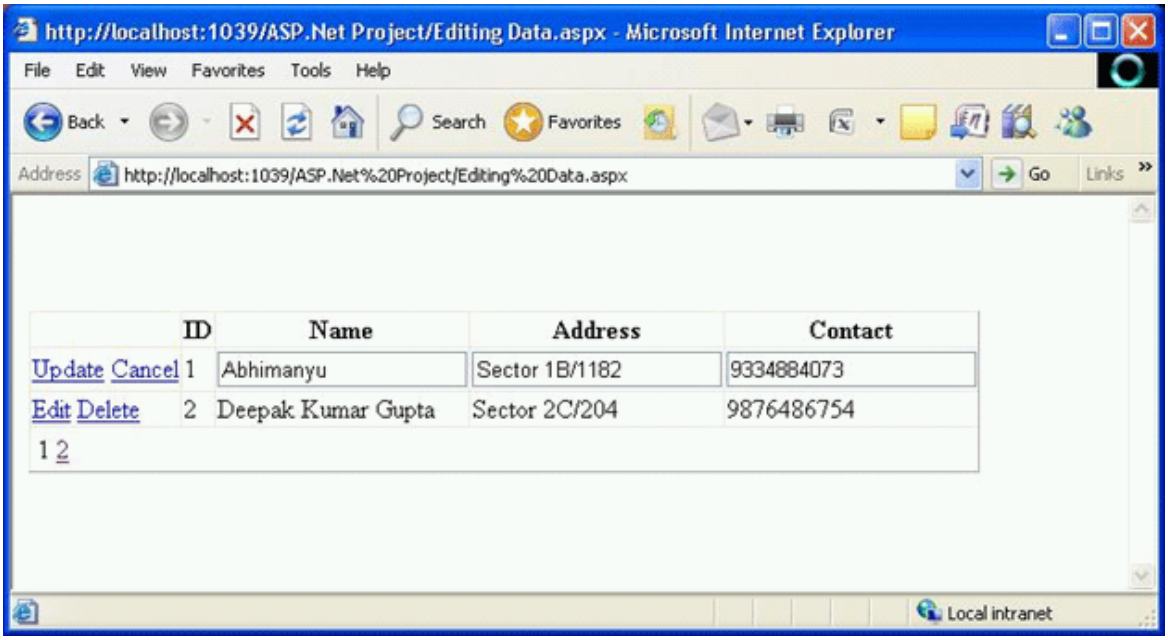

```
<%@ Page Language="VB" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
   Protected Sub paging_DataBound(ByVal sender As Object, ByVal e As EventArgs)
      Dim menuPager As Menu = CType(GridView1.BottomPagerRow.FindControl("menuPager"),Menu)
      For i As Integer = 0 To GridView1.PageCount - 1
         Dim item As New MenuItem()
         item.Text = String.Format("[{0}]", i + 1)
         item.Value = i.ToString()
         If GridView1.PageIndex = i Then
            item.Selected = True
         End If
         menuPager.Items.Add(item)
      Next
   End Sub

   Protected Sub menuPager_MenuItemClick(ByVal sender As Object, ByVal e As MenuEventArgs)
      GridView1.PageIndex = Int32.Parse(e.Item.Value)
   End Sub
</script>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title></title>
    <style type="text/css">
    .menu td
        {
            padding:5px 0px;
        }
    .selectedPage a
        {
            font-weight:bold;
            color:red;
        }

    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    <asp:GridView
        id="GridView1"
        DataSourceID="SqlDataSource1"
        AllowPaging="true"
        PageSize="2"
        Runat="server"
        OnDataBound="paging_DataBound" Width="622px">
        <PagerTemplate>
        <table>
        <tr><td>
        <asp:LinkButton
            id="lnkPrevious"
            Text="&lt; Prev"
            CommandName="Page"
            CommandArgument="Prev"
            ToolTip="Previous Page"
            Runat="server" />
        </td><td>
        <asp:Menu
            id="menuPager"
            Orientation="Vertical"
            OnMenuItemClick="menuPager_MenuItemClick"
            StaticSelectedStyle-CssClass="selectedPage"
            CssClass="menu"
            Runat="server" />
        </td><td>
        <asp:LinkButton
            id="lnkNext"
            Text="Next &gt;"
            CommandName="Page"
            CommandArgument="Next"
            ToolTip="Next Page"
            Runat="server" />
        </td></tr>
```

```
    </table>
    </PagerTemplate>
  </asp:GridView>

  <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
      SelectCommand="SELECT * FROM MyTB">
  </asp:SqlDataSource>

  </div>
  </form>
</body>

</html>
```

2.3 Editing Data

The GridView control also enables us to edit database data. The amazing thing is that we can use the GridView to edit the content of a database table row without writing a single line of code. Here is an example which illustrates how we can update and delete record in the database table using GridView only.



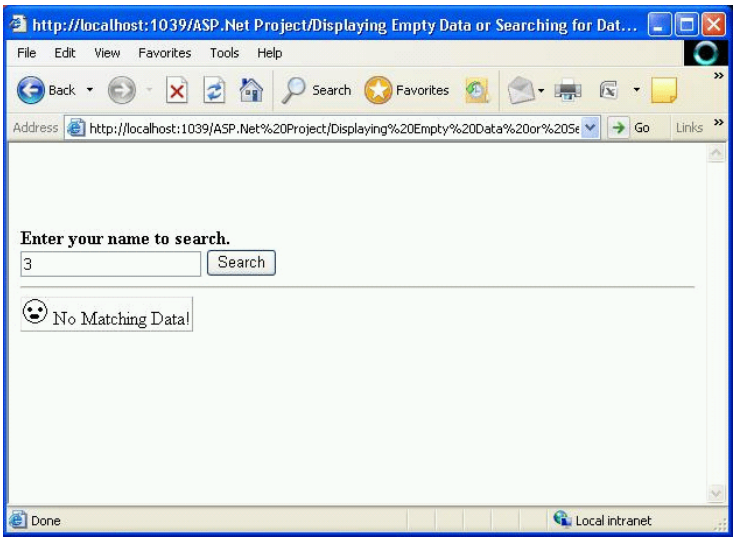

```
<%@ Page Language="VB" %>

<!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 id="Head1" runat="server">
```

```
    <title></title>
    <style type="text/css">

    </style>
</head>
<body>
    <form id="form1" runat="server" >
    <div>
    <br /><br /><br />
        <asp:GridView
        ID="GridView1"
        runat="server"
        DataSourceID="SqlDataSource1"
        PageSize="2"
        AutoGenerateEditButton="true"
        AutoGenerateDeleteButton="true"
        DataKeyNames="ID"
        AllowPaging="true">

        <SelectedRowStyle
        CssClass="mycss"
        BackColor="#666699"
        BorderColor="Red">
        </SelectedRowStyle>

        </asp:GridView>

        <br />
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
            SelectCommand="SELECT * FROM MyTB"
            UpdateCommand="UPDATE MyTB SET Name=@Name, Address=@Address WHERE ID=@ID"
            DeleteCommand="DELETE MyTB WHERE ID=@ID">
        </asp:SqlDataSource>
        <br />

    </div>
    </form>
</body>
</html>
```

Notice that the GridView control has both its AutoGenerateEditButton and AutoGenerateDeleteButton properties enabled. When these properties are enabled, Edit and Delete links are automatically rendered next to each row in the GridView.

2.4 Displaying Empty Data/Searching for Data

The GridView includes two properties that enable you to display content when no results are returned from the GridView control's data source. You can use either the EmptyDataText property or the EmptyDataTemplate property to handle empty data. The example given below contains a search form. If we enter a search string that does not match the start of any name, then the contents of the EmptyDataText property will be displayed.




```
<%@ Page Language="VB" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

   Protected Sub btnSubmit_Click(ByVal sender As Object, ByVal e As System.EventArgs)
      GridView1.Visible = True
   End Sub
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
   <title></title>
   <style type="text/css">

   </style>
</head>
<body>
   <form id="form1" runat="server" >
   <div>
   <br /><br /><br />
      <b>Enter your name to search.</b>
      <br />
       <asp:TextBox
       id="txtName"
       Runat="server" />

       <asp:Button
       id="btnSubmit"
       Text="Search"
       OnClick="btnSubmit_Click"
       Runat="server" />
       <hr />
```

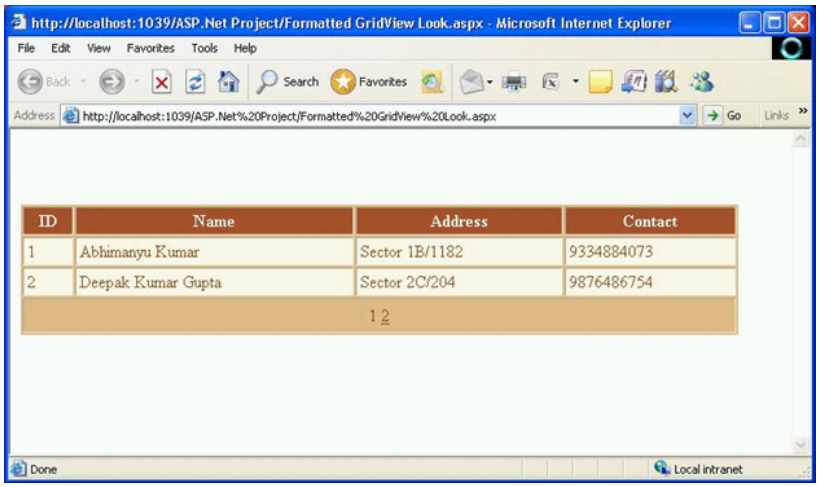
```
    <asp:GridView
    id="GridView1"
    DataSourceID="SqlDataSource1"
    EmptyDataText="<img src='sad.GIF'/> No Matching Data!"
    Visible="false"
    Runat="server" />

    <br />
    <asp:SqlDataSource
    id="SqlDataSource1"
    ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
    SelectCommand="SELECT iD, Name, Address, Contact FROM MyTB WHERE Name LIKE +'%'+ @Name+'%'"
    Runat="server">
    <SelectParameters>
    <asp:ControlParameter
      Name="Name"
      ControlID="txtName"
      PropertyName="Text" />
    </SelectParameters>
    </asp:SqlDataSource>
    <br />

  </div>
  </form>
</body>
</html>
```

2.5 Formatted GridView Look

The GridView control includes a rich set of formatting properties that you can use to modify its appearance. I recommend that you don't use most of these properties because using these properties results in bloated pages. Instead, I recommend that you use Cascading Style Sheets to format the GridView control.

The GridView control includes a CssClass property. The control also exposes several Style objects that include the CssClass property:

- AlternatingRowStyle Enables you to format every other row.
- FooterStyle Enables you to format the footer row.
- HeaderStyle Enables you to format the header row.
- PagerStyle Enables you to format the pager row.
- RowStyle Enables you to format each row.
- SelectedRowStyle Enables you to format the selected row.




```
<%@ Page Language="VB" %>

<!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 id="Head1" runat="server">
   <title></title>
   <style type="text/css">

   </style>
</head>
<body>
   <form id="form1" runat="server" >
   <div>
   <br /><br /><br />
      <asp:GridView
      ID="GridView1"
      runat="server"
      DataSourceID="SqlDataSource1"
      PageSize="2"
      AllowPaging="True" BackColor="#DEBA84" BorderColor="#DEBA84"
         BorderStyle="None" BorderWidth="1px" CellPadding="3" CellSpacing="2"
         Width="683px">
         <FooterStyle BackColor="#F7DFB5" ForeColor="#8C4510" />
         <HeaderStyle BackColor="#A55129" Font-Bold="True" ForeColor="White" />
         <PagerStyle ForeColor="#8C4510" HorizontalAlign="Center" />
         <RowStyle BackColor="#FFF7E7" ForeColor="#8C4510" />
         <SelectedRowStyle BackColor="#738A9C" Font-Bold="True" ForeColor="White" />
         <SortedAscendingCellStyle BackColor="#FFF1D4" />
         <SortedAscendingHeaderStyle BackColor="#B95C30" />
         <SortedDescendingCellStyle BackColor="#F1E5CE" />
```

```
        <SortedDescendingHeaderStyle BackColor="#93451F" />
    </asp:GridView>

    <br />
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
      SelectCommand="SELECT * FROM MyTB"
      UpdateCommand="UPDATE MyTB SET Name=@Name, Address=@Address WHERE ID=@ID"
      DeleteCommand="DELETE MyTB WHERE ID=@ID">
    </asp:SqlDataSource>
    <br />

  </div>
  </form>
</body>
</html>
```

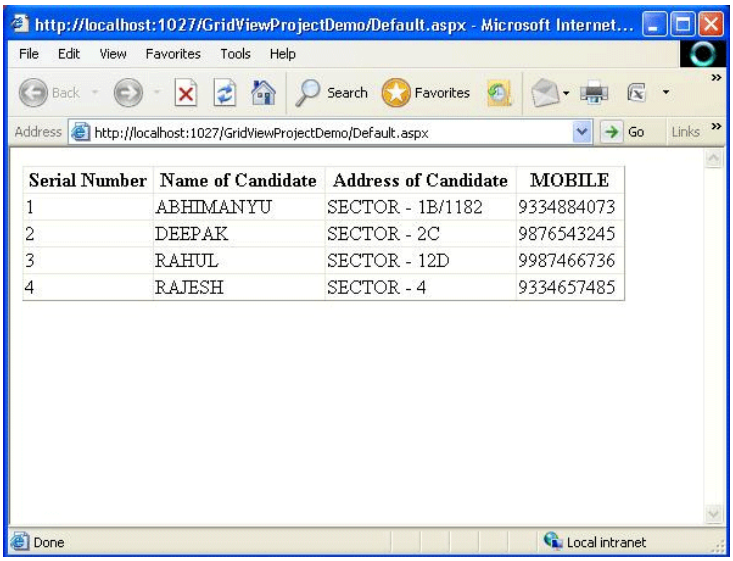
3 Using Fields with GridView Control

As to solve some of problems like enabling the GridView to render its columns automatically is that we give up any control over column formatting. For example, the BoxOfficeTotals column is displayed as a decimal amount without any currency formatting. The EnTRyDate column always displays in short-date and long-time format. The solution to such problems is to specify explicitly the fields that a GridView displays. The GridView control supports the following types of fields:

- BoundField Enables us to display the value of a data item as text.
- CheckBoxField Enables us to display the value of a data item as a check box.
- CommandField Enables us to display links for editing, deleting, and selecting rows.
- ButtonField Enables us to display the value of a data item as a button (image button, link button, or push button).
- HyperLinkField Enables us to display the value of a data item as a link.
- ImageField Enables us to display the valueof a data item as an image.
- TemplateField Enables us to customize the appearance of a data item.

3.1 Using BoundField

A BoundField always displays the value of a data item as text when a row is in normal display mode. When a row is selected for editing, a BoundField displays the value of a data item in a single line text field. The most important three properties of the BoundField class are the DataField, DataFormatString, and HeaderText properties.




```
<%@ Page Language="VB" %>

<!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></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
   <div>

      <asp:GridView ID="GridView1"
      runat="server"
      AutoGenerateColumns="False"
      DataSourceID="SqlDataSource1"
      EmptyDataText="There are no data records to display."
      Width="495px">
         <Columns>
            <asp:BoundField
            DataField="ID"
            HeaderText="Serial Number"
            SortExpression="ID"/>
            <asp:BoundField
            DataField="NAME"
            HeaderText="Name of Candidate"
            SortExpression="NAME" />
            <asp:BoundField
            DataField="ADDRESS"
            HeaderText="Address of Candidate"
```

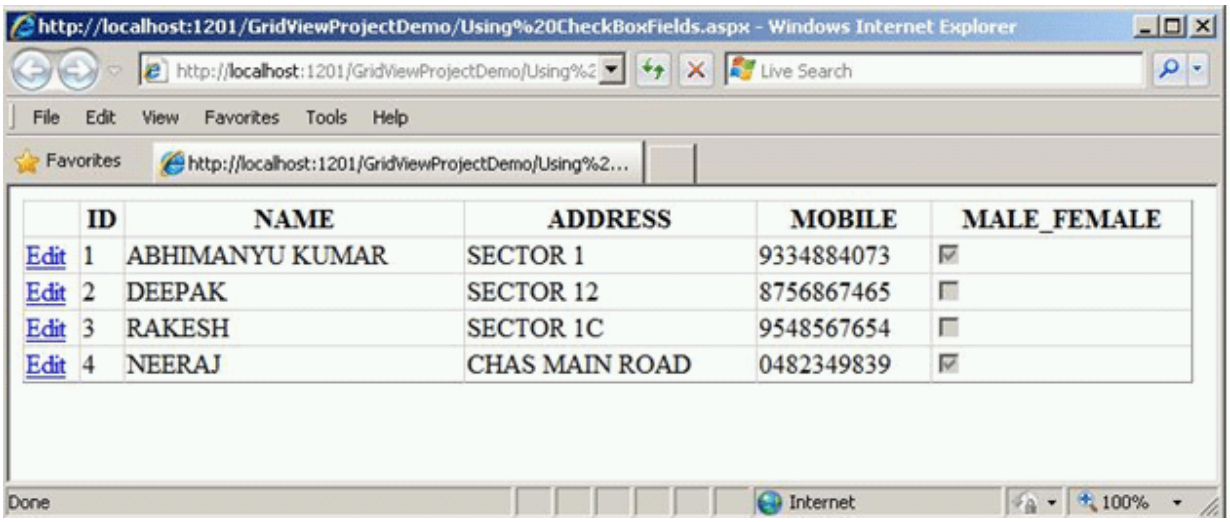
```
            SortExpression="ADDRESS" />
            <asp:BoundField
            DataField="MOBILE"
            HeaderText="MOBILE"
            SortExpression="MOBILE" />
          </Columns>
        </asp:GridView>
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
          ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
          ProviderName="<%$ ConnectionStrings:DatabaseConnectionString1.ProviderName %>"
          SelectCommand="SELECT [ID], [NAME], [ADDRESS], [MOBILE] FROM [MYTB]">
        </asp:SqlDataSource>

      </div>
    </div>
    </form>
</body>
</html>
```

Notice that the GridView control includes an AutoGenerateColumns property that is assigned the value False. If we don't disable automatically generated columns, then both columns represented by the BoundFields and all the columns from the data source are displayed redundantly. BoundFields also supports several useful properties as listed below.

- AccessibleHeaderText Enables we to add an HTML abbr attribute to the column header.
- ApplyFormatInEditMode Enables us to apply the DataFormatString to the field when the row is in edit display mode.
- ConvertEmptyStringToNull Enables we to convert an empty string "" into the value Nothing (null) when editing a column.
- DataField Enables we to specify the name of the field that the BoundField displays.
- DataFormatString Enables we to use a format string to format a data item.
- FooterStyle Enables we to format the column footer.
- FooterText Enables we to display text in the column footer.
- HeaderImageUrl Enables we to display an image in the column header.
- HeaderStyle Enables we to format the column header.
- HeaderText Enables we to display text in the column header.
- HtmlEncode Enables we to HTML-encode the value of a data item, which enables you to avoid script injection attacks.
- InsertVisible Enables we to not display a column when inserting a new record (does not apply to the GridView control).
- ItemStyle Enables we to format a data item.
- NullDisplayText Enables we to specify textthat is displayed when a data item has the value Nothing (null).
- ReadOnly Enables we to prevent the data item from being edited in edit mode.
- ShowHeader Enables we to display the column header.
- SortExpression Enables we to associate a sort expression with the column.
- Visible Enables we to hide a column.

3.2 Using Checkbox Fields

A Checkbox Field, as we can probably guess, displays a check box. When a row is not in edit mode, the check box is displayed but it is disabled.



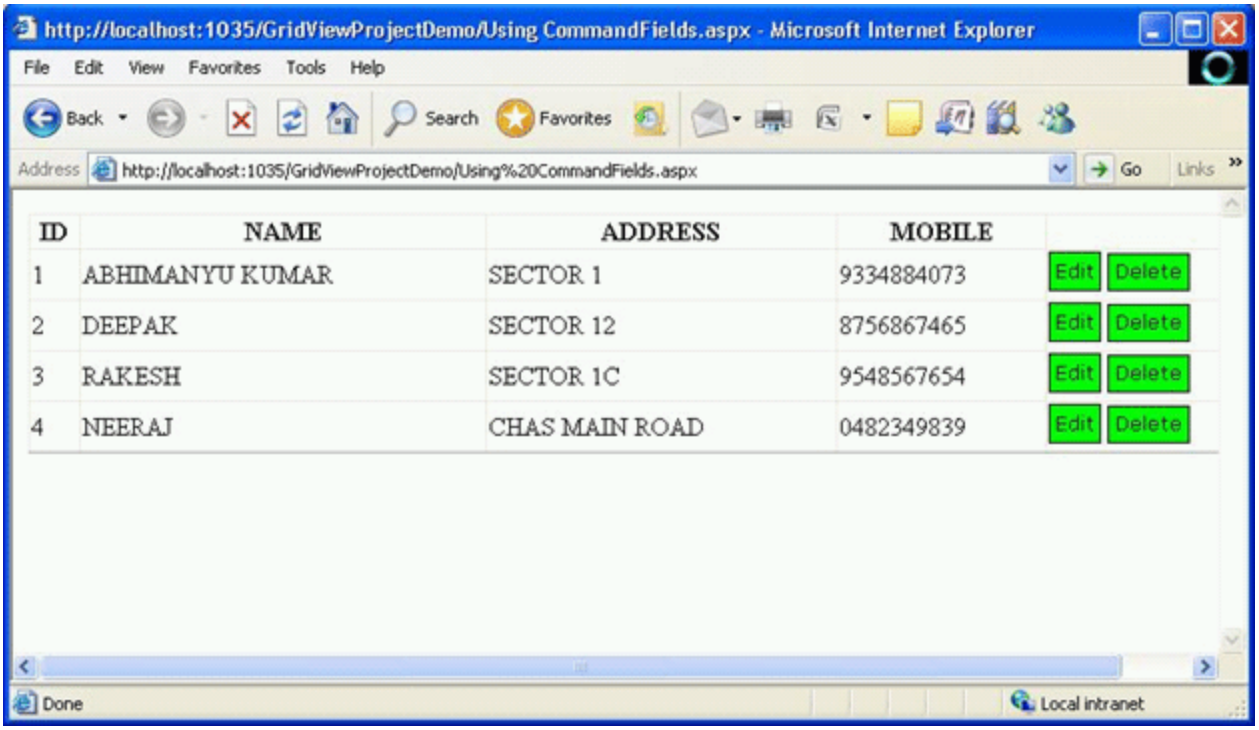
	ID	NAME	ADDRESS	MOBILE	MALE_FEMALE
Edit	1	ABHIMANYU KUMAR	SECTOR 1	9334884073	☑
Edit	2	DEEPAK	SECTOR 12	8756867465	☐
Edit	3	RAKESH	SECTOR 1C	9548567654	☐
Edit	4	NEERAJ	CHAS MAIN ROAD	0482349839	☑

```
<%@ Page Language="VB" %>

<!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 id="Head1" runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
   <div>

      <asp:GridView ID="GridView1"
      runat="server"
      AutoGenerateColumns="False"
      DataSourceID="SqlDataSource1"
      EmptyDataText="There are no data records to display."
      Width="709px">
         <Columns>
            <asp:CommandField ShowEditButton="True" />
            <asp:BoundField DataField="ID" HeaderText="ID" SortExpression="ID" />
            <asp:BoundField DataField="NAME" HeaderText="NAME" SortExpression="NAME" />
            <asp:BoundField DataField="ADDRESS" HeaderText="ADDRESS"
               SortExpression="ADDRESS" />
            <asp:BoundField DataField="MOBILE" HeaderText="MOBILE"
               SortExpression="MOBILE" />
            <asp:CheckBoxField DataField="MALE_FEMALE" HeaderText="MALE_FEMALE"
               SortExpression="MALE_FEMALE" />
         </Columns>
      </asp:GridView>
      <asp:SqlDataSource ID="SqlDataSource1" runat="server"
```

```
        ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
        ProviderName="<%$ ConnectionStrings:DatabaseConnectionString1.ProviderName %>"
        SelectCommand="SELECT [ID], [NAME], [ADDRESS], [MOBILE], [MALE_FEMALE] FROM [MYTB]"
        UpdateCommand="UPDATE MYTB SET ID =@ID, NAME =@NAME, ADDRESS =@ADDRESS, MOBILE =@MOBILE,
MALE_FEMALE =@MALE_FEMALE WHERE ID=@ID">
      </asp:SqlDataSource>
    </div>
   </div>
   </form>
</body>
</html>
```

3.3 Using Command Fields

We use CommandField to customize the appearance of the Edit, Delete, Update, Cancel and Select buttons displayed by the GridView control by default. We can change these all control by own. Let's take a look.




```
<%@ Page Language="VB" %>
<!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></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
```

```
    <asp:GridView
    ID="GridView1"
    runat="server"
    AutoGenerateColumns="False"
    DataSourceID="SqlDataSource1"
    EmptyDataText="There are no data records to display."
    Width="788px"
    DataKeyNames="ID">
      <Columns>
        <asp:BoundField
        DataField="ID"
        HeaderText="ID"
        SortExpression="ID" />
        <asp:BoundField
        DataField="NAME"
        HeaderText="NAME"
        SortExpression="NAME" />
        <asp:BoundField
        DataField="ADDRESS"
        HeaderText="ADDRESS"
        SortExpression="ADDRESS" />
        <asp:BoundField
        DataField="MOBILE"
        HeaderText="MOBILE"
        SortExpression="MOBILE" />
        <asp:CommandField
      ButtonType="Image"
      ShowEditButton="true"
      EditText="Edit Movie"
      EditImageUrl="~/images/edit.GIF"
      UpdateText="Update Movie"
      updateimageurl="~/images/update.GIF"
      ShowCancelButton="true"
      CancelText="Cancel Edit"
      cancelimageurl="~/images/cancel.GIF"
      ShowDeleteButton="true"
      DeleteText="Delete Movie"
      Deleteimageurl="~/images/delete.GIF"/>
      </Columns>
    </asp:GridView>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
      DeleteCommand="DELETE FROM MYTB WHERE ID=@ID"
      ProviderName="<%$ ConnectionStrings:DatabaseConnectionString1.ProviderName %>"
      SelectCommand="SELECT [ID], [NAME], [ADDRESS], [MOBILE] FROM [MYTB]"
      UpdateCommand="UPDATE MYTB SET ID =@ID, NAME =@NAME, ADDRESS =@ADDRESS, MOBILE =@MOBILE WHERE
ID=@ID">
    </asp:SqlDataSource>

    </div>
  </form>
</body>
</html>
```

Notice that we do not enable the AutoGenerateEditButton or AutoGenerateDeleteButton properties when using a CommandField. Instead, we use the CommandField to set up the standard editing buttons explicitly.

The CommandField supports the following properties:

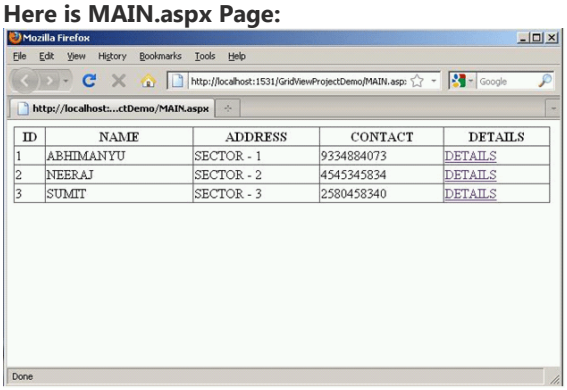
- ButtonType Enables we to specify the type of button displayed by the CommandField. Possible values are Button, Image, and Link.
- CancelImageUrl Enables we to specify an image to display for the Cancel button.
- CancelText Enables we to specify the text to display for the Cancel button.
- CausesValidation Enables we to disable validation when an edit button is clicked.
- DeleteImageUrl Enables we to specify an image to display for the Delete button.
- DeleteText Enables we to specify the text to display for the Delete button.
- EditImageUrl Enables we to specify an image to display for the Edit button.
- EditText Enables we to specify the text to display for the Edit button.
- InsertImageUrl Enables we to specify an image to display for the Insert button.
- InsertText Enables we to specify the text to display for the Insert button.
- NewImageUrl Enables we to specify an image to display for the New button (does not apply to GridView).
- NewText Enables we to specify the text to display for the New button.
- SelectImageUrl Enables we to specify the image to display for the Select button.
- SelectText Enables we to specify the text to display for the Select button.
- ShowCancelButton Enables we to display theCancel button.
- ShowDeleteButton Enables we to display theDelete button.
- ShowEditButton Enables we to display the Edit button.
- ShowInsertButton Enables we to display theInsert button (does not apply to GridView).
- ShowSelectButton Enables we to display theSelect button.
- UpdateImageUrl Enables we to specify the image to display for the Update button.
- UpdateText Enables we to specify the text to display for the Update button.
- ValidationGroup Enables we to associate the edit buttons with a validation group.

3.4 Using HyperLink Field

Introduction & Demonstration

We use a HyperLink Field to create a link to another page. Assume an example; we have two database tables named MAIN and DETAILS. MAIN table consist only fields like ID, NAME, ADDRESS, CONTACT and DETAILS. DETAILS table consist fields like ID, MONEY and SALARY. Now we have two forms named MAIN.aspx and DETAILS.aspx. MAIN.aspx will display MAIN table's data and DETAILS.aspx will display DETAILS table's data. Now we want a HyperLink in each record in MAIN.aspx form and when use will click on that link it will redirect to DETAILS.aspx page and it will only consist the data based on the ID what you have clicked on MAIN.aspx page. If you run directly DETAILS.aspx page it will not show you any data because it has no parameter passed. Take a look on development.

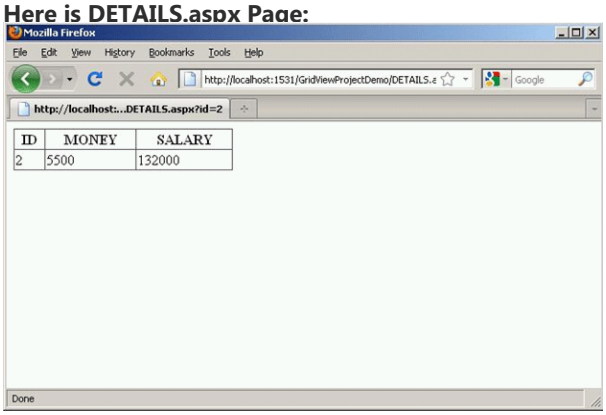
Here is MAIN.aspx Page:



```
<%@ Page Language="VB" %>

<!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></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>

      <asp:GridView
      ID="GridView1"
      runat="server"
      AutoGenerateColumns="False"
      DataSourceID="SqlDataSource1"
      EmptyDataText="There are no data records to display."
      Width="653px">
         <Columns>
            <asp:BoundField
            DataField="ID"
            HeaderText="ID"
            SortExpression="ID" />
            <asp:BoundField
            DataField="NAME"
            HeaderText="NAME"
            SortExpression="NAME" />
            <asp:BoundField
            DataField="ADDRESS"
            HeaderText="ADDRESS"
            SortExpression="ADDRESS" />
            <asp:BoundField
            DataField="CONTACT"
            HeaderText="CONTACT"
            SortExpression="CONTACT" />
            <asp: HyperLinkField
            HeaderText="DETAILS"
            DataTextField="DETAILS"
            DataNavigateUrlFields="ID"
            DataNavigateUrlFormatString="DETAILS.aspx?id={0}" />
      </Columns>
      </asp:GridView>
      <asp:SqlDataSource ID="SqlDataSource1" runat="server"
         ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
         ProviderName="<%$ ConnectionStrings:DatabaseConnectionString1.ProviderName %>"
         SelectCommand="SELECT [ID], [NAME], [ADDRESS], [CONTACT], [DETAILS] FROM [MAIN]">
      </asp:SqlDataSource>
```

```
    </div>
  </form>
</body>
</html>
```

Here is DETAILS.aspx Page:



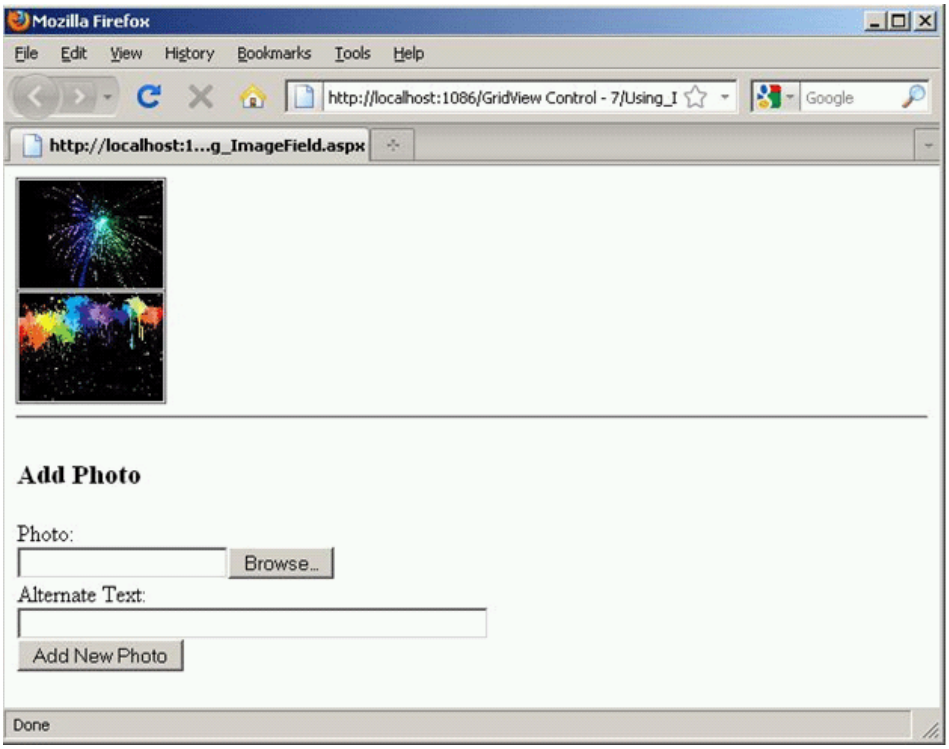

```
<%@ Page Language="VB" %>

<!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></title>
</head>
<body>
  <form id="form1" runat="server">
  <div>

    <asp:GridView
    ID="GridView1"
    runat="server"
    DataSourceID="SqlDataSource1"
    EmptyDataText="There are no data records to display."
    Width="248px">
    </asp:GridView>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
      ProviderName="<%$ ConnectionStrings:DatabaseConnectionString1.ProviderName %>"
      SelectCommand="SELECT [ID], [MONEY], [SALARY] FROM [DETAILS] WHERE ID=@ID">
      <SelectParameters>
      <asp:QueryStringParameter
      Name="ID"
      QueryStringField="ID" />
      </SelectParameters>
    </asp:SqlDataSource>
```

```
    </div>
  </form>
</body>
</html>
```

3.5 Using ImageField

Introduction & Demonstration

We use an ImageField to display an image stored on the server's hard drive. We can't use an ImageField to display images stored in a database table.



In above picture, we have a GridView control in which two pictures are displaying. We have also used a horizontal row to separate upload section and display section. For upload we have used lots of controls. Here is the code which I have used.

```
<%@ Page Language="VB" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
  Protected Sub Inserting(ByVal sender As Object, ByVal e As FormViewInsertEventArgs)
    ' Get the FileUpload control
    Dim Images As FileUpload = CType(frmPhoto.FindControl("Images"), FileUpload)
    SqlDataSource1.InsertParameters("FileName").DefaultValue = Images.FileName
    ' Save contents to file system
    Dim savePath As String = MapPath("~/Images/" + Images.FileName)
    Images.SaveAs(savePath)
  End Sub
</script>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>

         <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
          DataSourceID="SqlDataSource1"
          EmptyDataText="There are no data records to display." Width="96px"
          ShowHeader="false" Height="41px">
      <Columns>
          <asp:ImageField
          DataImageUrlField="FileName"
          DataImageUrlFormatString="~/Images/{0}"
          DataAlternateTextField="AltText"
          ControlStyle-Width="100px" />
      </Columns>
      </asp:GridView>
      <asp:SqlDataSource ID="SqlDataSource1" runat="server"
          ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
          ProviderName="<%$ ConnectionStrings:DatabaseConnectionString1.ProviderName %>"
          SelectCommand="SELECT [FILENAME], [ALTTEXT] FROM [IMAGES_DETAILS]"
          InsertCommand="INSERT INTO IMAGES_DETAILS(FILENAME, ALTTEXT) VALUES (@FILENAME, @ALTTEXT)">
          <InsertParameters>
             <asp:Parameter Name="FILENAME" />
          </InsertParameters>
      </asp:SqlDataSource>
   <hr />

   <asp:FormView
      id="frmPhoto"
      DefaultMode="Insert"
      DataSourceID="SqlDataSource1"
      OnItemInserting="Inserting"
      Runat="server">
      <InsertItemTemplate>
      <h3>Add Photo</h3>
      <asp:Label
         id="lblPhoto"
         Text="Photo:"
         AssociatedControlID="Images"
         Runat="server" />
      <br />
      <asp:FileUpload
         id="Images"
         Runat="server" />
      <br />
      <asp:Label
         id="lblAltText"
         Text="Alternate Text:"
         AssociatedControlID="txtAltText"
         Runat="server" />
```

```
        <br />
        <asp:TextBox
          id="txtAltText"
          Text='<%# Bind("AltText") %>'
          Columns="50"
          Runat="server" />
        <br />
        <asp:Button
          id="btnInsert"
          Text="Add New Photo"
          CommandName="Insert"
          Runat="server" />
      </InsertItemTemplate>
    </asp:FormView>

    </div>
    </form>
</body>
</html>
```

3.6 Using Template Field

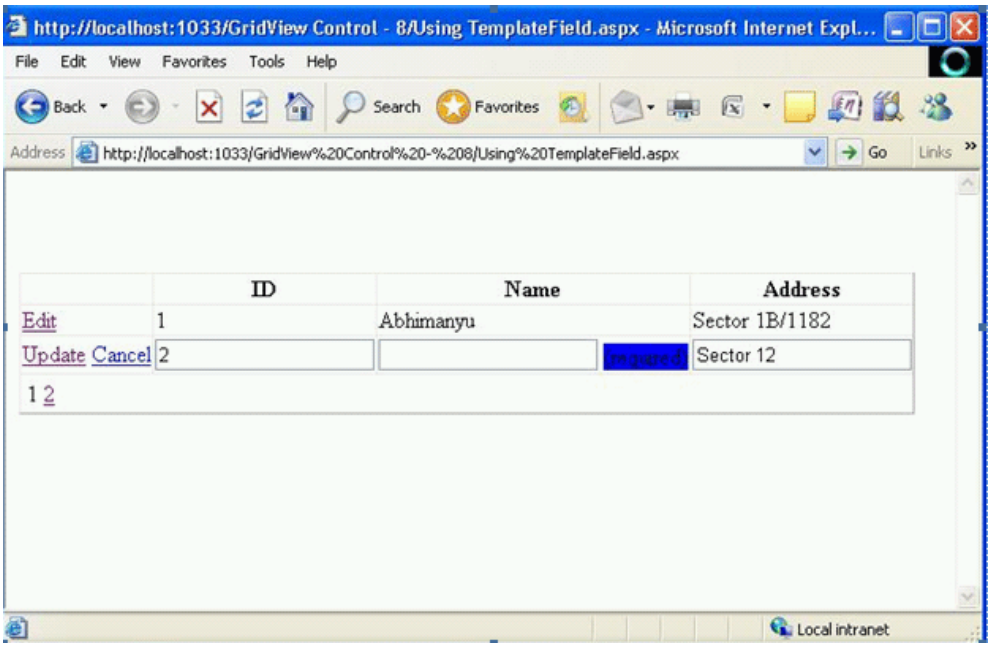
Introduction & Demonstration

A Template Field enables we to add any content to a GridView column that we need. A TemplateField can contain HTML,

DataBinding expressions, or ASP.NET controls.

TemplateFields are particularly useful when we are using a GridView to edit database records. You can use a TemplateField to customize the user interface and add validation to the fields being edited.

Assume an example, we want a required field validation or list box when we edit any row. These all features can be enabled using TemplateField. Let's take a look, here we are using required field validation.



```
<%@ Page Language="VB" %>

<!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 id="Head1" runat="server">
   <title></title>
   <style type="text/css">

   </style>
</head>
<body>
   <form id="form1" runat="server" >
   <div>
   <br /><br /><br />
      <asp:GridView
      ID="GridView1"
      runat="server"
      DataSourceID="SqlDataSource1"
      PageSize="2"
      AutoGenerateEditButton="true"
      DataKeyNames="ID"
      AllowPaging="true" AutoGenerateColumns="false">
      <Columns>
      <asp:TemplateField HeaderText="ID">
         <ItemTemplate>
         <%# Eval("ID")%>
         </ItemTemplate>
         <EditItemTemplate>
         <asp:TextBox
            id="txtID"
            Text='<%# Bind("ID") %>'
            Runat="server" />
         </EditItemTemplate>
      </asp:TemplateField>

      <asp:TemplateField HeaderText="Name">
         <ItemTemplate>
         <%# Eval("Name")%>
         </ItemTemplate>
         <EditItemTemplate>
         <asp:TextBox
            id="txtName"
            Text='<%# Bind("Name") %>'
            Runat="server" />
            <asp:RequiredFieldValidator
            id="valName"
            ControlToValidate="txtName"
```

```
            Text="(required)"
            Runat="server"
            BackColor="Blue"/>
        </EditItemTemplate>
      </asp:TemplateField>

      <asp:TemplateField HeaderText="Address">
        <ItemTemplate>
        <%# Eval("Address")%>
        </ItemTemplate>
        <EditItemTemplate>
        <asp:TextBox
          id="txtAddress"
          Text='<%# Bind("Address") %>'
          Runat="server" />
        </EditItemTemplate>
      </asp:TemplateField>
      </Columns>
      </asp:GridView>
      <br />
      <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
        SelectCommand="SELECT * FROM MyTB"
        UpdateCommand="UPDATE MyTB SET Name=@Name, Address=@Address WHERE ID=@ID">
      </asp:SqlDataSource>
      <br />
    </div>
    </form>
</body>
</html>
```

4 Working with GridView Control Events

The GridView control includes a rich set of events that we can handle to customize the control's behavior and appearance. These events can be divided into three groups.

First, the GridView control supports the following set of events that are raised when the control displays its rows:

- DataBinding Raised immediately before the GridView is bound to its data source.
- DataBound Raised immediately after a GridView is bound to its data source.
- RowCreated Raised when each row in the GridView is created.
- RowDataBound Raised when each row in the GridView is bound to data.

Second, the GridView control includes the following set of events that are raised when we are editing records:

- RowCommand Raised when an event is raised by a control contained in the GridView.
- RowUpdating Raised immediately before a GridView updates a record.
- RowUpdated Raised immediately after a GridView updates a record.
- RowDeleting Raised immediately before a GridView deletes a record.
- RowDeleted Raised immediately after a GridView deletes a record.
- RowCancelingEdit Raised when we cancel updating a record.

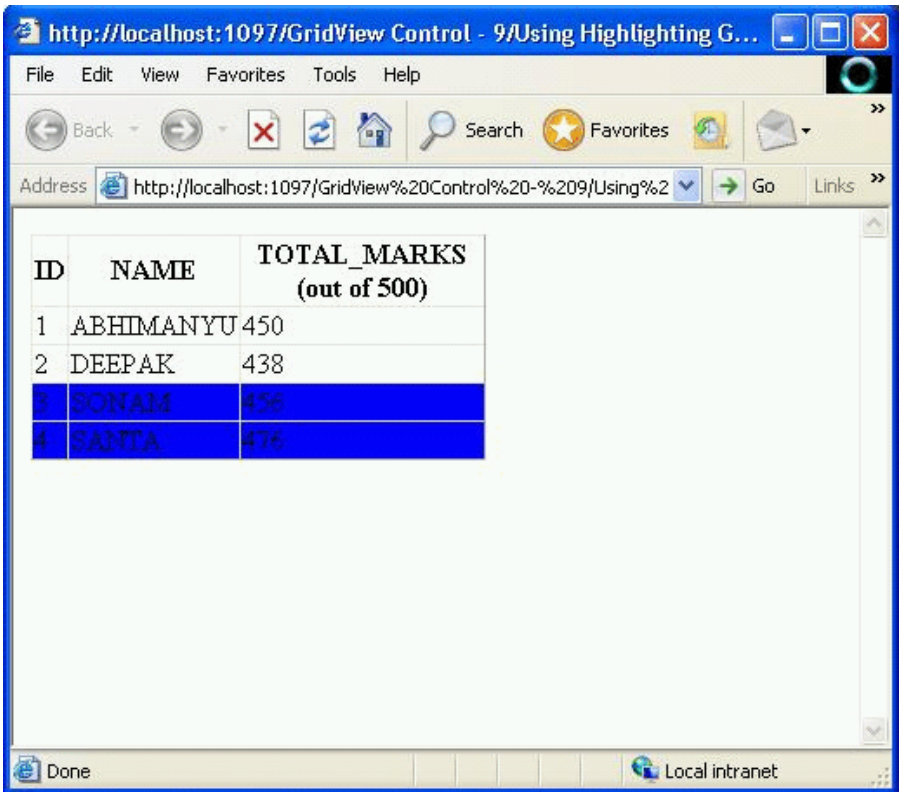
Finally, the GridView control supports the following events related to sorting, selecting, and paging:

- PageIndexChanging Raised immediately before the current page is changed.
- PageIndexChanged Raised immediately after the current page is changed.
- Sorting Raised immediately before sorting.
- Sorted Raised immediately after sorting.
- SelectedIndexChanging Raised immediately before a row is selected.
- SelectedIndexChanged Raised immediately after a row is selected.

4.1 Using Highlighting GridView Rows

Introduction & Demonstration

Imagine that we want to highlight particular rows in a GridView. For example, when displaying a table of marks totals, we might want to highlight the rows in which the marks are greater than a certain amount.




```
<%@ Page Language="VB" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
   Protected Sub GridView1_RowDataBound(ByVal sender As Object, ByVal e AsGridViewRowEventArgs)
      If e.Row.RowType = DataControlRowType.DataRow Then
         Dim TOTAL_MARKS As Decimal = CType(DataBinder.Eval(e.Row.DataItem,"TOTAL_MARKS"), Decimal)
         If TOTAL_MARKS > 450 Then
            e.Row.BackColor = System.Drawing.Color.Blue
         End If
      End If
   End Sub
```

```
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
   <title></title>
   <style type="text/css">

   </style>
</head>
<body>
   <form id="form1" runat="server" >
   <div>

        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
        DataSourceID="SqlDataSource1"
        EmptyDataText="There are no data records to display." Width="262px"OnRowDataBound="GridView1_RowDataBound">
        <Columns>
          <asp:BoundField DataField="ID" HeaderText="ID" SortExpression="ID" />
          <asp:BoundField DataField="NAME" HeaderText="NAME" SortExpression="NAME" />
          <asp:BoundField DataField="TOTAL_MARKS" HeaderText="TOTAL_MARKS (out of 500)"
            SortExpression="TOTAL_MARKS" />
        </Columns>
      </asp:GridView>
      <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
        ProviderName="<%$ ConnectionStrings:DatabaseConnectionString1.ProviderName %>"
        SelectCommand="SELECT [ID], [NAME], [TOTAL_MARKS] FROM [MARKSHEET]">
      </asp:SqlDataSource>

    </div>
   </form>
</body>
</html>
```

In above figure, the GridView1_RowDataBound() method is executed when the GridView renders each of its rows. The second parameter passed to this event handler is an instance of the GridViewRowEventArgs class. This class exposes a GridViewRow object that represents the row being bound.

The GridViewRow object supports several useful properties, here is some of them:

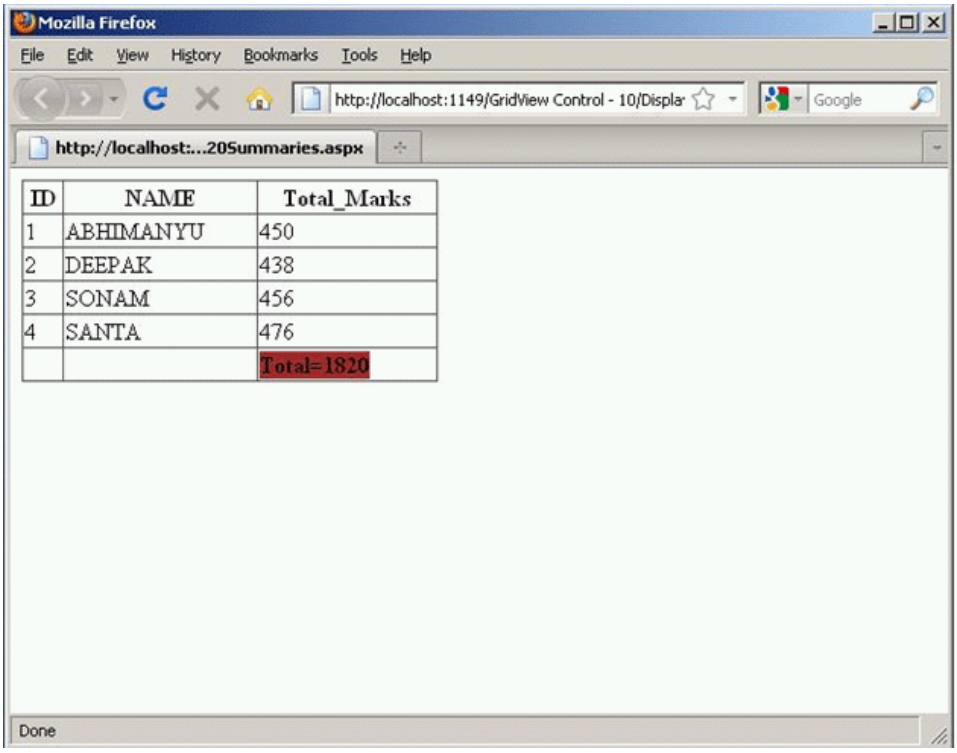
- Cells Represents the collection of table row cells associated with the row being bound.
- DataItem Represents the data item associated with the row being bound.
- DataItemIndex Represents the index of the data item in its DataSet associated with the row being bound.
- RowIndex Represents the index of the row being bound.
- RowState Represents the state of the row being bound. Possible values are Alternate, Normal, Selected, Edit. Because these values can be combined (for example, the RowState can be Alternate Edit), use a bitwise comparison with RowState.
- RowType Represents the type of row being bound. Possible values are DataRow, Footer, Header, NullRow, Pager, Separator.

In above list, the RowType property we have used to verify that the row is a DataRow. The DataItem property is used to retrieve the database record associated with the row. Notice that the DataBinder.Eval() method is used to retrieve the value of the TOTAL_MARKS.

5 Handle GridView RowDataBound Events

Introduction & Demonstration

Imagine that we want to display a column total at the bottom of a column. In that case, we can handle the GridView RowDataBound event to sum the values in a column and display the summary in the column footer. For example, in the page given below contains a GridView control that displays a summary column representing the total marks obtained by all students.




```
<%@ Page Language="VB" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

    Private TotalMarks As Decimal = 0

    Protected Sub TotalMarks_RowDataBound(ByVal sender As Object, ByVal e AsGridViewRowEventArgs)
        If e.Row.RowType = DataControlRowType.DataRow Then
            Dim Totals As Decimal = CType(DataBinder.Eval(e.Row.DataItem, "TOTAL_MARKS"),Decimal)
            TotalMarks += Totals
        End If
        If e.Row.RowType = DataControlRowType.Footer Then
            Dim lblTotalMarks As Label = CType(e.Row.FindControl("lblTotal"), Label)
            lblTotalMarks.Text = String.Format("Total=" & TotalMarks)
        End If
    End Sub

</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
```

```
    <title></title>
    <style type="text/css">

    </style>
</head>
<body>
    <form id="form1" runat="server" >

    <asp:GridView
      id="GridView1"
      DataSourceID="SqlDataSource1"
      OnRowDataBound="TotalMarks_RowDataBound"
      AutoGenerateColumns="false"
      ShowFooter="true"
      Runat="server" Width="287px">
      <Columns>
      <asp:BoundField
        DataField="ID"
        HeaderText="ID" />
      <asp:BoundField
        DataField="NAME"
        HeaderText="NAME" />
      <asp:TemplateField HeaderText="Total_Marks">
      <ItemTemplate>
        <%# Eval("TOTAL_MARKS")%>
      </ItemTemplate>
      <FooterTemplate>
        <asp:Label
          id="lblTotal"
          Runat="server" Font-Bold="true" BackColor="Brown"/>
      </FooterTemplate>
      </asp:TemplateField>
      </Columns>
    </asp:GridView>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1 %>"
      ProviderName="<%$ ConnectionStrings:DatabaseConnectionString1.ProviderName %>"
      SelectCommand="SELECT [ID], [NAME], [TOTAL_MARKS] FROM [MARKSHEET]">
    </asp:SqlDataSource>
    </form>
</body>
</html>
```

Notice that the GridView control uses a TemplateField to represent the MARKS_TOTAL column. The TemplateField includes a <FooterTemplate> that contains a Label control. The TotalMarks_RowDataBound() method displays the total of the marks in this Label control.