

# DIGGING INTO THE DATAGRID CONTROL

Paul Litwin  
Deep Training  
paul@deeptraining.com  
www.deeptraining.com



# Paul Litwin

## ◆ Developer

- Focus: ASP.NET, ASP, VB, SQL Server, Access, ...
- MCSD
- Microsoft MVP

## ◆ Co-Founder and Senior Trainer

- Deep Training
  - [www.deeptraining.com](http://www.deeptraining.com)

## ◆ Conference Chair/Speaker

- Chair, Microsoft ASP.NET Connections
- Speaker at Tech\*Ed, VSLive, ...
- Member INETA Speakers Bureau
- Leader of Web Developer Meeting, .NET Dev Assn

## ◆ Author

- Author/co-author of a dozen books, including...
  - *ASP.NET for Developers*
  - *Access 2002 Desktop/Enterprise Dev Handbook*

# Samples, Slides & Plug

- ◆ You can download the samples and slides for this presentation from
  - [www.deeptraining.com/litwin](http://www.deeptraining.com/litwin)
- ◆ I'm teaching an 5-day ASP.NET class in Kirkland the week of Sept 22
  - Members of the group are eligible for a \$400 discount. Register online at [www.deeptraining.com](http://www.deeptraining.com) using **NETDA2003** sales code or call us at 206-282-5096 for more details

# Agenda

- ◆ DataGrid Basics
- ◆ Sorting it Out
- ◆ Pagination
- ◆ Updating Data using a DataGrid
- ◆ Template Columns

# DataGrid Basics

# DataGrid Control

- ◆ Columnar grid
- ◆ Support sorting, paging & in-place editing
- ◆ Has default interface (i.e., you don't have to supply templates)
- ◆ Templates
  - HeaderTemplate
  - ItemTemplate
  - AlternatingItemTemplate
  - EditItemTemplate
  - SelectedItemTemplate
  - FooterTemplate
  - PagerTemplate

# DataGrid Control

- ◆ AutogenerateColumns Attribute of DataGrid
  - True (default) – all DataSource columns included as `<asp:BoundColumn>` columns
  - False – you specify the columns
- ◆ Column Types
  - BoundColumn – label/textbox
  - ButtonColumn – button
  - EditCommandColumn – button with special editing capabilities
  - HyperlinkColumn – hyperlink
  - TemplateColumn – if none of the others apply; you will have to do binding using `<%# xxx %>` syntax

# DataGrid Control

## ◆ Events

- **OnEditCommand** – occurs when EditCommandColumn's Edit button is clicked
- **OnUpdateCommand** – occurs when EditCommandColumn's Update button is clicked
- **OnCancelCommand** – occurs when EditCommandColumn's Cancel button is clicked
- **OnDeleteCommand** – occurs when ButtonColumn with CommandName="Delete" is clicked
- **OnItemCommand** – occurs when any button within a DataGrid is clicked
- **OnPageIndexChanged** – occurs when page changes
- **OnSortCommand** – occurs when column sort is requested

# Basic DataGrid Example

## BasicGrid.aspx (HTML)

```
<form runat="server">
  <asp:DataGrid id="dgrCustomers" runat="server"
    AutoGenerateColumns="True"
    BorderColor="black"
    BorderWidth="1"
    GridLines="Both"
    CellPadding="3"
    CellSpacing="0"
    Font-Name="Verdana"
    Font-Size="10pt"
    HeaderStyle-BackColor="#99ccff"
    AlternatingItemStyle-BackColor="lightgray">
  </asp:DataGrid>
</form>
```

# Basic DataGrid Example

## BasicGrid.aspx (Code)

```
Private Sub Page_Load(...)
    If Not Page.IsPostBack Then
        Call BindDataGrid()
    End If
End Sub

Sub BindDataGrid()
    Dim strCnx As String = "..."
    Dim strSQL As String = _
        "SELECT CustomerID, CompanyName, ContactName, " & _
        "Country FROM Customers ORDER BY CompanyName"

    Dim cnx As SqlConnection = New SqlConnection(strCnx)
    cnx.Open()

    Dim cmd As SqlCommand = New SqlCommand(strSQL, cnx)
    Dim sdr As SqlDataReader

    ' Fill DataSet with Customers query
    sdr = cmd.ExecuteReader(CommandBehavior.CloseConnection)

    ' Bind reader to grid
    dgrCustomers.DataSource = sdr
    dgrCustomers.DataBind()
End Sub
```

# Sorting it Out

# Sorting Grids

- ◆ Steps to provide sorting
  1. AllowSorting = "True"
  2. Provide sorting event handler that retrieves sort expression (name of sort column)
  3. Sort data based on sort expression

# Simple Sort Example

## SortedGrid1.aspx

```
Private Sub dgrCustomers_SortCommand(source As Object, _
    e As DataGridSortCommandEventArgs)
    Call BindDataGrid(e.SortExpression)
End Sub

Sub BindDataGrid(ByVal strSort As String)
    ...

    ' Customize sort order based on strSort
    Dim strSQL As String = _
        "SELECT CustomerID, CompanyName, ContactName, " & _
        "Country FROM Customers " & _
        "ORDER BY " & strSort

    ...

```

# Ascending/Descending Sorts

- ? Need to store away sort column and direction between postbacks
  - Use ViewState

# Ascending/Descending Sort Example

## SortedGrid2.aspx – rebuilds DataReader each time

```
Private Sub dgrCustomers_SortCommand(  
    If ViewState("SortCol") = e.SortExpression Then  
        If ViewState("SortDir") = "ASC" Then  
            ViewState("SortDir") = "DESC"  
        Else  
            ViewState("SortDir") = "ASC"  
        End If  
    Else  
        ViewState("SortDir") = "ASC"  
    End If  
    ViewState("SortCol") = e.SortExpression  
    Call BindDataGrid()  
End Sub  
Sub BindDataGrid()  
    ...  
    Dim strSQL As String = _  
        "SELECT CustomerID, CompanyName, ContactName, "  
        "Country FROM Customers "  
    If Not ViewState("SortCol") Is Nothing Then  
        strSQL &= "ORDER BY " & ViewState("SortCol") & " " & _  
            ViewState("SortDir")  
    End If  
    ...  
End Sub
```

# Ascending/Descending Sort Example

## SortedGrid3.aspx – caches DataView in Session

```
Private Sub dgrCustomers_SortCommand(...)
  If ViewState("SortCol") = e.SortExpression Then
    If ViewState("SortDir") = "ASC" Then
      ViewState("SortDir") = "DESC"
    Else
      ViewState("SortDir") = "ASC"
    End If
  Else
    ViewState("SortDir") = "ASC"
  End If
  ViewState("SortCol") = e.SortExpression

  CType(Session("dvCustomer"), DataView).Sort = _
    ViewState("SortCol") & " " & ViewState("SortDir")

  Call BindDataGrid()
End Sub
```

# Pagination

# Pagination

- ◆ Properties related to pagination
  - AllowPaging
  - AllowCustomPaging
  - PageSize
  - PagerStyle-Mode (NextPrev or NumericPages)
  - PagerStyle-Position (Top, Bottom, TopAndBottom)
  - PagerStyle-PageButtonCount
  - ...

# Automatic Pagination

- ◆ AllowPaging = "True"
- ◆ You provide DataGrid with data source and a PageSize and it takes care of chopping up data into pages and displaying the correct page
- ◆ Very little coding on your part
- ◆ Requires query to be re-run for every page unless you cache the data source somewhere

# Pagination Examples

CustomersPageAuto1.aspx – rebuilds DataReader each time

CustomersPageAuto2.aspx – caches DataSet in Session

```
<asp:DataGrid id="dgrCustomers" runat="server"
```

```
...
```

```
    AllowPaging="True"
```

```
    PageSize="10"
```

```
    PagerStyle-Mode="NumericPages"
```

```
    PagerStyle-HorizontalAlign="Left">
```

```
</asp:DataGrid>
```

```
Private Sub dgrCustomers_PageIndexChanged(source As Object, _  
e As DataGridPageChangedEventArgs)
```

```
    dgrCustomers.CurrentPageIndex = e.NewPageIndex
```

```
    Call BindDataGrid()
```

# Custom Pagination

- ◆ AllowPaging = "True"
- ◆ AllowCustomPaging = "True"
- ◆ You provide DataGrid with each page of data
- ◆ More coding than automatic paging

# Custom Pagination, Solution #1

- ◆ How do you split up data into pages and grab correct page?
  - Solution #1: Use arbitrarily-divided evenly-sized pages
    - Use db-specific SQL (temp tables, cursors, etc.)
    - Example:
      - CustomersPageCustom1.aspx
      - Uses procGetCustomersPage stored proc & SQL Server temp tables

# Arbitrarily-Divided Pages Example

## CustomersPageCustom1.aspx – code

```
Function GetPage(intPageSize As Integer, intPageIndex As Integer) As SqlDataReader
    Dim strCnx As String = "server=localhost;uid=sa;pwd=;database=northwind;"
    Dim prm As SqlParameter

    Dim cnx As SqlConnection = New SqlConnection(strCnx)
    cnx.Open()

    Dim cmd As SqlCommand = New SqlCommand("procGetCustomersPage", cnx)
    cmd.CommandType = CommandType.StoredProcedure

    ' Create parameters
    ...

    Dim sdr As SqlDataReader
    sdr = cmd.ExecuteReader(CommandBehavior.CloseConnection)

    Return sdr
End Function

Private Sub dgrCustomers_PageIndexChanged(source As Object, e As _
    DataGridPageChangedEventArgs)
    dgrCustomers.CurrentPageIndex = e.NewPageIndex
    dgrCustomers.DataSource = GetPage(dgrCustomers.PageSize, e.NewPageIndex)
    dgrCustomers.DataBind()
End Sub
```

# Arbitrarily-Divided Pages Example

## CustomersPageCustom1.aspx – stored proc

```
ALTER PROCEDURE procGetCustomersPage
( @pageSize int, @pageIndex int )
AS
DECLARE @PageLowerBound int
DECLARE @PageUpperBound int
SET @PageLowerBound = @pageSize * @pageIndex
SET @PageUpperBound = @pageLowerBound + @pageSize + 1
-- Create a temp table to store the select results
CREATE TABLE #PageIndex ( IndexID int IDENTITY (1, 1) NOT NULL, PageID NCHAR(5) )
BEGIN
-- INSERT into the temp table
INSERT INTO #PageIndex (PageID)
SELECT CustomerID
FROM Customers
ORDER BY CustomerID

SELECT CustomerID, CompanyName, ContactName, Country
FROM Customers WITH (nolock)
JOIN #PageIndex WITH (nolock)
ON CustomerID = PageID
WHERE IndexID > @PageLowerBound AND
IndexID < @PageUpperBound
ORDER BY IndexID
```

# Custom Pagination, Solution #2

- ◆ How do you split up data into pages and grab correct page?
  - Solution #2: Use data-dependent unevenly-sized pages
    - Use an existing column to split up pages
    - Example:
      - CustomersPageCustom2.aspx
      - Uses `CompanyName LIKE "A*"` expression
      - Requires use of `ItemCreated` event handler to fix up pager

# Data-Dependent Pages Example

## CustomersPageCustom2.aspx – code 1 of 2

```
Function GetPage(ByVal intPageIndex As Integer) As SqlDataReader
    Dim strCnx As String = "server=localhost;uid=sa;pwd=;database=northwind;"
    Dim strSQL = "SELECT CustomerID, CompanyName, ContactName, Country " & _
        "FROM Customers " & _
        "WHERE CompanyName LIKE '" & Chr(intPageIndex + Asc("A")) & "%' " & _
        "ORDER BY CompanyName "

    Dim cnx As SqlConnection = New SqlConnection(strCnx)
    cnx.Open()

    Dim cmd As SqlCommand = New SqlCommand(strSQL, cnx)
    cmd.CommandType = CommandType.Text

    Dim sdr As SqlDataReader
    sdr = cmd.ExecuteReader()

    Return sdr
End Function

Private Sub dgrCustomers_PageIndexChanged(source As Object, e As _
    DataGridPageChangedEventArgs)
    dgrCustomers.CurrentPageIndex = e.NewPageIndex
    dgrCustomers.DataSource = GetPage(e.NewPageIndex)
    dgrCustomers.DataBind()
End Sub
```

# Data-Dependent Pages Example

## CustomersPageCustom2.aspx – code 2 of 2

```
Private Sub dgrCustomers_ItemCreated(sender As Object, e As DataGridItemEventArgs)
    Dim lit As ListItemType = e.Item.ItemType
    Dim intl As Integer

    If lit = ListItemType.Pager Then
        Dim pgr As TableCell = e.Item.Controls(0)

        For intl = 0 To pgr.Controls.Count - 1
            Dim ctl As Object = pgr.Controls(intl)

            If TypeOf ctl Is LinkButton Then
                ' LinkButton control represents other pages
                Dim lbt As LinkButton = CType(pgr.Controls(intl), LinkButton)

                If CType(ctl, LinkButton).Text <> "..." Then
                    lbt.Text = "|" & Chr(Convert.ToInt32(lbt.Text) + Asc("A") - 1) & "|"
                End If
            ElseIf TypeOf ctl Is Label Then
                ' Label control represents current page
                Dim lbl As Label = CType(pgr.Controls(intl), Label)
                lbl.Text = "|" & Chr(Convert.ToInt32(lbl.Text) + Asc("A") - 1) & "|"
                lbl.BackColor = Color.SkyBlue
                lbl.Font.Bold = True
            End If
        Next intl
    End If
End Sub
```

# Updating Data using a DataGrid

# Grid Update Example

- ◆ Data Lives in Three Places
  - DataGrid – markup on the page
  - DataSet – cached in Session
  - Database – SQL Server

# Update Example -- HTML SqlDataSetUpdate.aspx

```
<asp:datagrid id="dgrEmployees" runat="server"
...
autogeneratecolumns="False" >
<columns>
<asp:editcommandcolumn
edittext="Edit" updatetext="Save" canceltext="Cancel"
itemstyle-wrap="false" headertext="Edit" headerstyle-wrap="false" />
<asp:buttoncolumn headertext="Delete" text="Delete"
commandname="Delete" />
<asp:boundcolumn headertext="Employee Id"
datafield="EmployeeId" readonly="True" />
<asp:boundcolumn headertext="First Name" datafield="FirstName" />
<asp:boundcolumn headertext="Last Name" datafield="LastName" />
<asp:boundcolumn headertext="Phone" datafield="HomePhone" />
</Columns>
</asp:datagrid>
<asp:linkbutton id="cmdAddRow" runat="server"
text="Add New Record" />
<asp:button id="cmdDbCommit" runat="server"
text="Commit Changes" />
<asp:button id="cmdDbAbandon" runat="server"
text="Abandon Changes" />
```

## Update Example – Switching to Edit Mode

SqlDataSetUpdate.aspx – EditCommand handler

```
Sub dgrEmployees_EditCommand(source As Object, _
    e As DataGridCommandEventArgs)
    ' This code is executed when the Edit button
    ' associated with the EditCommandColumn is clicked.
    ' Set the EditItemIndex to the index of the row
    ' that triggered the Edit event
    dgrEmployees.EditItemIndex = e.Item.ItemIndex
    Call BindDataGrid()
    ' Disable buttons and clear message during editing
    Call EnableDbButtons(False)
    lblMsg.Text = ""
End Sub
```

# Update Example – Canceling Edit Mode

## SqlDataSetUpdate.aspx – CancelCommand handler

```
Sub dgrEmployees_CancelCommand(source As Object, _  
    e As DataGridCommandEventArgs)  
    ' This code is executed when the Cancel button  
    ' associated with the EditCommandColumn is clicked.  
    ' Reset EditItemIndex to -1 which takes out of edit mode  
    dgrEmployees.EditItemIndex = -1  
    Call BindDataGrid()  
End Sub
```

# Update Example – Deleting a Grid Row

## SqlDataSetUpdate.aspx – DeleteCommand handler

```
Sub dgrEmployees_DeleteCommand(source As Object, _
    e As DataGridCommandEventArgs)
    Dim drFound As DataRow
    ' Find row in DataSet
    drFound = ds.Tables("Employees").Rows.Find(e.Item.Cells(2).Text)
    If Not drFound Is Nothing Then
        ' Grab controls from DataGrid
        drFound.Delete()
    Else
        lblMsg.Text = "Error: Row not found"
    End If
    ' Reset EditItemIndex to -1 which takes us out of edit mode
    dgrEmployees.EditItemIndex = -1
    Session("ds") = ds
    Call BindDataGrid()

    ...
End Sub
```

# Update Example – Updating DataSet

## SqlDataSetUpdate.aspx – UpdateCommand handler

```
Sub dgrEmployees_UpdateCommand(source As Object, e As DataGridCommandEventArgs)
    Dim drFound As DataRow
    ' Find row in DataSet
    drFound = ds.Tables("Employees").Rows.Find(e.Item.Cells(2).Text)
    If Not drFound Is Nothing Then
        ' Grab controls from DataGrid
        Dim ctlFirstName As TextBox = e.Item.Cells(3).Controls(0)
        Dim ctlLastName As TextBox = e.Item.Cells(4).Controls(0)
        Dim ctlPhone As TextBox = e.Item.Cells(5).Controls(0)
        Try
            ' Update DataRow values with DataGrid control values
            drFound("FirstName") = ctlFirstName.Text
            drFound("LastName") = ctlLastName.Text
            drFound("HomePhone") = ctlPhone.Text
            ' Turn on constraints that may have been disabled for insert
            ds.EnforceConstraints = True

            ' Reset EditItemIndex to -1 which takes us out of edit mode
            dgrEmployees.EditItemIndex = -1
            Session("ds") = ds
            Call BindDataGrid()
        ...
```

## Update Example – Adding Row to DataSet & Grid SqlDataSetUpdate.aspx – AddRow\_Click handler

```
Sub cmdAddRow_Click(sender As System.Object, _  
    e As EventArgs)  
    Dim dr As DataRow  
    ' Turn off constraints because empty row will  
    ' not be valid with constraints on  
    ds.EnforceConstraints = False  
  
    ' Add new empty row and store away in DataSet  
    dr = ds.Tables("Employees").NewRow()  
    ds.Tables("Employees").Rows.Add(dr)  
    Session("ds") = ds  
  
    ' Set EditItemIndex to new row and rebind  
    dgrEmployees.EditItemIndex = dgrEmployees.Items.Count  
    Call BindDataGrid()  
End Sub
```

# Update Example – Updating Database

## SqlDataSetUpdate.aspx – cmdDbCommit\_Click handler

```
Sub cmdDbCommit_Click(sender As Object, e As EventArgs)
    ...
    Dim sda As SqlDataAdapter = New _
        SqlDataAdapter(strSQL, cnx)
    ...
    ' Create SqlCommandBuilder to handle updating
    Dim scb As SqlCommandBuilder = New _
        SqlCommandBuilder(sda)
    sda.UpdateCommand = scb.GetUpdateCommand()
    sda.DeleteCommand = scb.GetDeleteCommand()
    sda.InsertCommand = scb.GetInsertCommand()

    ' Call Update method on Employees DataTable.
    intRows = sda.Update(ds, "Employees")
    ' Reset state of edited rows.
    ds.AcceptChanges()
    ...
End Sub
```

# Template Columns

# When to Use Template Column?

- ◆ If you need to do something that is not provided by standard DataGrid columns
  - Problem: Need to link to another page but querystring may contain special characters
    - Try using CustomerMain.aspx with Netscape and a CustomerId containing a space
  - Solution: Use TemplateColumn that calls Server.UrlEncode to encode querystring
    - CustomerMain\_Template.aspx

# TemplateColumn Example – Encoding querystring CustomerMain\_Template.aspx

```
<asp:TemplateColumn HeaderText="Customer">
  <itemtemplate>
    <asp:Label
      Text='<%# CreateLinkCol(Container.DataItem("CustomerId"),
        Container.DataItem("CompanyName")) %>'
      runat="server"/>
    </itemtemplate>
  </asp:TemplateColumn>
```

```
Function CreateLinkCol(ByVal strId, ByVal strName) As String
  ' Create hyperlink with encoded querystring
  ' to handle special chars in Id
  Return "<a href=""CustomerEdit_TemplateCol.aspx?Customer=" & _
    Server.UrlEncode(strId) & """">" & strName & "</a>"
End Function
```

# When to Use Template Column?

- ◆ If you need to do something that is not provided by standard DataGrid columns
  - Problem: Would like user to be able to use a dropdownlist control when editing within a grid
  - Solution: Use TemplateColumn
    - `DataGridWithDropDown.aspx`

# DropDownList Example

## DataGridWithDropDown.aspx– HTML

```
<asp:templatecolumn headertext="Title">
  <itemtemplate>
    <%# Container.DataItem("Title")%>
  </itemtemplate>
  <edititemtemplate>
    <asp:dropdownlist
      id="drpTitle"
      datatextfield="Title"
      datavaluefield="Title"
      datasource="<%# GetTitlesDataSource()%>"
      selectedindex='<%# GetTitlesSelectedIndex(Container.DataItem("Title"))%>'
      runat="server"/>
    </edititemtemplate>
  </asp:templatecolumn>
```

# DropDownList Example

DataGridWithDropDown.aspx– GetTitlesDataSource function

```
Public Function GetTitlesDataSource() As DataView
    'Provide DataSource to Title column's EditItemTemplate
    Return ds.Tables("EmployeeTitles").DefaultView
End Function
```

# DropDownList Example

DataGridWithDropDown.aspx– GetTitlesSelectedIndex function

```
Public Function GetTitlesSelectedIndex(ByVal objItem As Object) _
    As Integer
    'Provide Initial SelectedIndex to Title column's EditItemTemplate
    Dim intI = 0
    Dim dtEmpTitles As DataTable = ds.Tables("EmployeeTitles")
    Dim drEmpTitles As DataRow
    ' Handle null values
    If objItem Is System.DBNull.Value Then
        objItem = "(none)"
    End If
    ' Iterate through Title rows to locate matching item.
    ' When a match is found, intI will be SelectedIndex.
    For Each drEmpTitles In dtEmpTitles.Rows
        If objItem = drEmpTitles("Title") Then
            Return intI
        End If
        intI = intI + 1
    Next
End Function
```

# Summary

- ◆ We explored various DataGrid properties and events
- ◆ We looked at how to sort columns in a grid in both ascending and descending order
- ◆ We learned how to use automatic pagination
- ◆ We learned how to employ custom pagination, both with arbitrarily-divided and data-divided pages
- ◆ We looked at editing, inserting, and deleting of grid rows data in-place
- ◆ Learned how to use templated columns to provide encoding of querystrings and editable dropdownlist controls

# Samples, Slides & Plug

- ◆ You can download the samples and slides for this presentation from
  - [www.deeptraining.com/litwin](http://www.deeptraining.com/litwin)
- ◆ I'm teaching an 5-day ASP.NET class in Kirkland the week of Sept 22
  - Members of the group are eligible for a \$400 discount. Register online at [www.deeptraining.com](http://www.deeptraining.com) using NETDA2003 sales code or call us at 206-282-5096 for more details