**Lab 12 – CS 3340**

To make this document easier to read, I recommend that you turn off spell checking and grammar checking in Word:

1. Choose: File, Option, Proofing
2. At the very bottom, check: “Hide spelling errors…” and “Hide grammar errors…”

**Introduction**

This tutorial shows how to provide “master/detail” filtering. On the first page we develop (see Figure 1 below), we use a DropDownList to do the filtering: select a Team and the corresponding Players are displayed. On the second page (see Figure 2), we use a GridView to do the filtering: the Teams are displayed in GridView and when a Team is selected the corresponding Players are displayed below. Other enhancements (not shown in figures below) are: add a calculated field to the GridView, color code select rows based on a condition, and detecting an invalid delete.

|  |  |
| --- | --- |
| E:\Data-Classes\CS 3340 - Web Programming\Topics\05_Database2\Labs\sp15\bb13.jpg  Figure 1 – Master/Detail filtering with a drop down. | E:\Data-Classes\CS 3340 - Web Programming\Topics\05_Database2\Labs\sp15\bb12.jpg  Figure 2 – – Master/Detail filtering with a GridView. |

**Lab Objectives**

1. Implement master/detail filtering using a drop down and a grid view.
2. Implement master/detail filtering using two grid views.
3. Add a calculated field to the GridView.
4. Customize the display of the GridView.

**Lab Organization**

There are 7 stages to complete Lab 12.

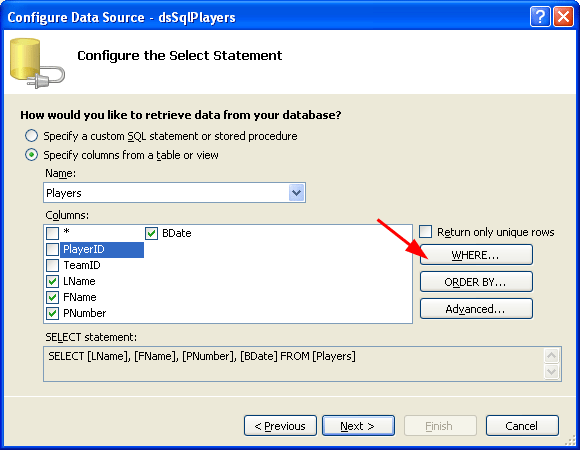
|  |  |
| --- | --- |
| Stage | Description |
| 1 | Build DropDownList to Display Teams |
| 2 | Build GridView to Display Players on Team Selected From DropDownList |
| 3 | Build GridView to Display Teams |
| 4 | Build GridView to Display Players on Team Selected From GridView |
| 5 | Add a Calculated Field to GridView |
| 6 | Customize the Display |
| 7 | Detecting an Invalid Delete |

**Stage 1 – Build DropDownList to Display Teams**

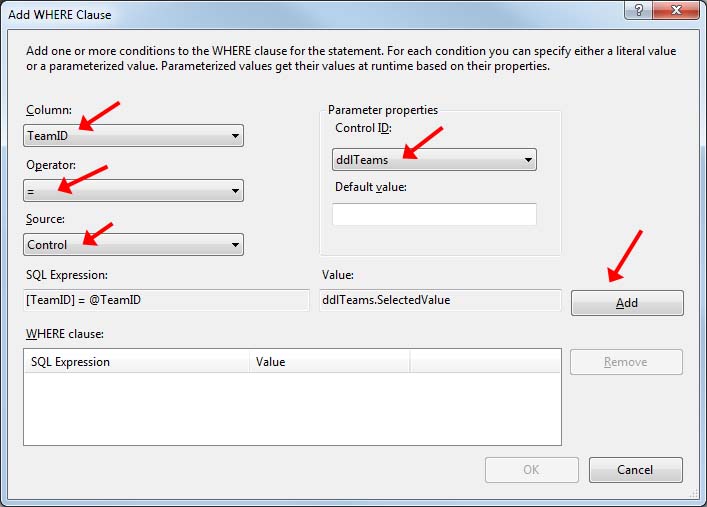
1. Locate a copy of *players.zip* from Lab 9 (or 10 or 11).
2. Create your *lab12* project (solution folder must be named *lab12\_lastName*).
3. Create an *App\_Data* ASP.NET folder (See Lab 10 if necessary) and drag the database into it.
4. Add a web form named, *Default.aspx*.
5. Ajaxify your page (See Lab 7, Stage 3 if necessary): add a *ScriptManager*, an *UpdatePanel*, and put all subsequent controls inside the *UpdatePanel*
6. Do the following:
7. Add a *DropDownList* to your page, set the ID to “ddlTeams”, and set *AutoPostBack* to *True*.
8. Configure the drop down to display *Name* from the *Teams* table and the *Value* to be *TeamID* (see Lab 10, Stage 6, Step 8 only, if necessary)
9. Build and run your page (Ctrl+Shift+B, Ctrl+F5). Make sure the teams are displayed in the drop down.

**Stage 2 – Build GridView to Display Players on Team Selected From DropDownList**

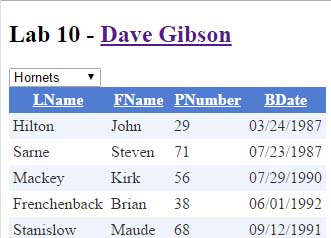
1. Add a *GridView* to your page just below the *DropDownList* and provide the ID, *gvPlayers*.



1. Configure the GridView to display players
2. Select *gvPlayers*, click the fly-out menu
3. Select, “Choose Data Source...”
4. Select “<New Data Source...>”.
5. Select, “SQL Database”
6. Provide the ID: “dsPlayers”
7. Select, “OK”
8. From the drop down, select the connection created above.
9. Select, “Next”
10. Select the “Players” table.
11. Select the fields shown on the right.
12. Choose, the “WHERE” button.
13. Set the values as shown below and then choose, “Add”. Note that we are setting the GridView to be synched with the DropDownList by specifying the “Column” (TeamID) and “Control ID” (ddTeams). This is adding a WHERE clause to the SQL statement defined in the DataSource. Note that the “SQL Expression” will appear differently than shown below. It will be: “[TeamID] = ?”.

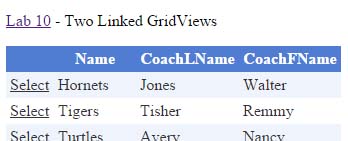


1. Choose “OK”, then “Next”, then “Finish”.
2. Style the GridView:



1. Format the BDate field to display as: {0:MM/dd/yyyy}. Hint: Choose: Edit Columns, Select BDate from Selected fields, and then set the *DataFormatString.* See a previous Lab if necessary.
2. Enable Paging and Sorting for the GridView (fly-out menu)
3. Apply a style to the GridView (fly-out, Auto-format)
4. Build and run your page (Ctrl+Shift+B, Ctrl+F5). Test thoroughly. You should be able to select a team from the drop down and the corresponding player be immediately displayed (using Ajax) in the grid view.

**Stage 3 – Build GridView to Display Teams**

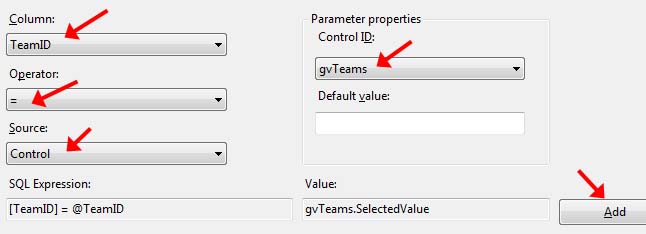
1. Add a web form named, *Page2.aspx* and add a link to *Default.aspx* to this new page.
2. Ajaxify your page. See a previous Lab if necessary.
3. Add a GridView to *Page2*.
4. Set the ID to *gvTeams*.
5. Configure it to show all four fields in the Teams table: TeamID, Name, CoachLName, CoachFName.
6. Enable row selection (Hint: fly-out)
7. Hide the *TeamID* field (Hint: fly-out, Edit Columns, select TeamID from Selected fields, and set Visible to False)
8. Apply a style to the GridView (fly-out, Auto-format)
9. Build and run your page (Ctrl+Shift+B, Ctrl+F5). Test thoroughly. The Select button will not work yet.

**Stage 4 – Build GridView to Display Players on Team Selected From GridView**

1. Add another GridView to Page2 below the first one and provide the ID, *gvPlayers*.
2. Configure the GridView to display players:
3. Display fly-out menu for GridView, and choose “new data source…”
4. Select the Database icon (accept the default ID).
5. Choose *“OK”*
6. From the drop down, select the connection created above.
7. Choose “Next”
8. Select the Players table from the drop down.
9. Select the LName, FName, PNumber, and BDate fields
10. Choose the “Where” button



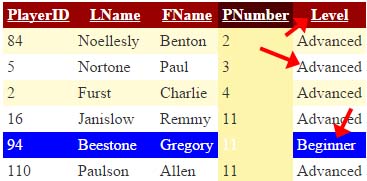
1. Choose the values shown below, then choose the “Add” button, then “OK”, “Next”, “Finish”



1. Apply a style to the GridView (fly-out, Auto-format)
2. Build and run your page (Ctrl+Shift+B, Ctrl+F5). Test the Select feature.

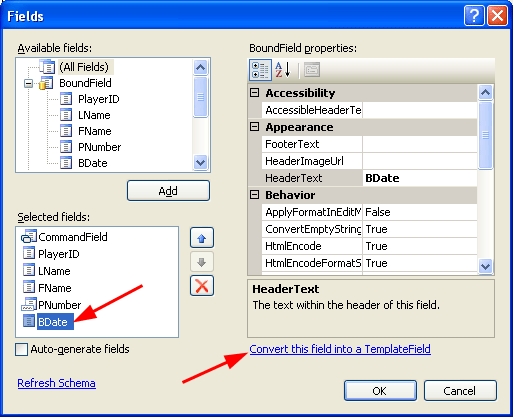
****

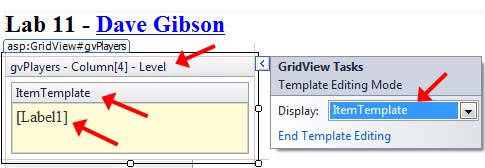
**Stage 5 – Add a Calculated Field to GridView**

In this Stage we customize the display of one of the fields from the Players table. Instead of showing the actual birth date (BDate), we will code it to display a label (Beginner, Intermediate, or Advanced) according to the criteria shown in the table below. The figure on the right shows the result.

|  |  |
| --- | --- |
| **Criteria** | **Display** |
| Less than 18 years old | Beginner |
| Between 18 and 22 years old | Intermediate |
| More than 22 years old | Advanced |

In Stage 6 we will further customize by highlighting in blue Players that are *Beginners* as shown in the figure above.

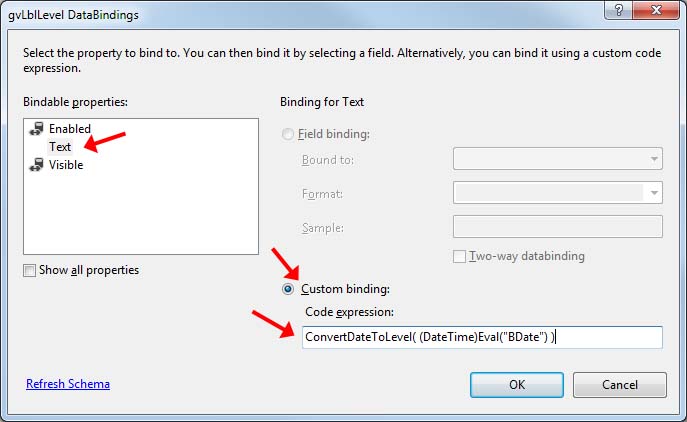
1. Select *gvPlayers* (the lower GridView) and display the fly-out menu and do the following:
2. Choose “Edit Columns.”
3. Select “BDate” and “Convert this field into a Template Field” (lower right).
4. Select “BDate” and change the *Header Text* property to “Level” (instead of “BDate”).
5. Choose “OK”
6. Again, display the “GridView Tasks” fly-out menu
7. Choose “Edit Templates.”
8. From the drop-down, choose “Item Template” under “Level”.



1. Select the label, change the ID to, *gvLblLevel*.
2. Select the label and click the arrow icon on the right. Choose “Edit DataBindings”.
3. Choose “Custom binding:” and type this in for the “Code expression”:

**ConvertDateToLevel( (DateTime)Eval("BDate") )**

This is calling a page method we will write in the next step. Choose, “OK”.

****

1. Display the “GridView Tasks” menu and choose “End Template Editing.”
2. Go to the code-behind page and add this method shown below. Note that this method is called when the Label’s data binding is taking. This method accepts a DateTime and returns a string stating whether the player is a “Beginner”, “Intermediate”, or “Advanced”. Thus, the returned string is bound to the Text property of the Label.

protected string ConvertDateToLevel(DateTime dt) {

TimeSpan ts = DateTime.Now - dt;

int days = ts.Days;

if (days < 6570) // 18 years old

return "Beginner";

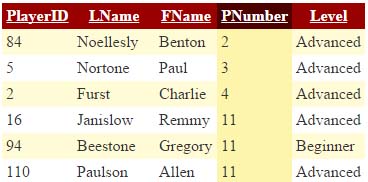
else if (days < 8030) // 22 years old

return "Intermediate";

else

return "Advanced";

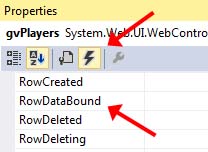
}



1. Build/Run (Ctrl+Shift+B, Ctrl+F5). Verify that the “Level” is displaying properly. (There may not be many “Intermediate” or “Beginner” players. If not, you could open the database and change a few birth dates. Alternately, you could change some of the values in the code above.).

**Stage 6 – Customize the Display**

Next, we are going to customize the display by highlighting the rows where we have a “Beginner.”

1. Select *gvPlayers* (the lower GridView) and do the following:
2. Choose the “Events” icon in the Property Window.
3. Double-click the “RowDataBound” event. An event stub will be created in the code-behind file.
4. Add the code below (just replace the event stub from previous step) and read the comments.

/// <summary>

/// This event is fired after each row in the GridView is created. Thus, the

/// default row creation is complete. So, this event allows us to intercept this

/// just completed row and modify it. Here, on each row, we check the Level

/// (Beg, Intermed, Adv), and then color the entire row if the Level is Beginner.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void gvPlayers\_RowDataBound(object sender, GridViewRowEventArgs e) {

// Only color code if NOT in edit mode. The EditIndex property is the index of

// the row being edited, or -1 if not being editted.

if (gvPlayers.EditIndex == -1) {

// Only attempt to color code if the row contains data. In other words,

// the top row is the column headers and we are not interested in that.

// Nor are we interested in the footer row.

// Note that the event arg, "e" contains a reference to the current row.

if (e.Row.RowType == DataControlRowType.DataRow) {

// Obtain a reference to the label that contains the Level

Label l = (Label)e.Row.FindControl("gvLblLevel");

// If the Level is Beginner, color the row.

if (l.Text.Equals("Beginner")) {

//l.ForeColor = System.Drawing.Color.White;

e.Row.BackColor = System.Drawing.Color.Blue;

e.Row.ForeColor = System.Drawing.Color.White;

}

}

}

}

1. Build/Run (Ctrl+Shift+B, Ctrl+F5). Thoroughly test. Note that “Beginner” rows are highlighted blue.



1. (No action required) Note, if you wanted the actual birth date (BDate) when a row is selected (we haven’t added Selection for the Player’s Gridview), then you would have to put the statement below in the ItemTemplate for BDate/Level in the Source:

**<asp:HiddenField ID="BDateHidden" runat="server" Visible="false"**

**Value='<%# Eval("BDate") %>' />**

And then to access the birth date we find the hidden field using its ID:

**// Access birth data from HiddenField**

**HiddenField BDateHidden = (HiddenField)gvPlayers.Rows[**

**gvPlayers.SelectedIndex ].FindControl("BDateHidden");**

**DateTime bDate = Convert.ToDateTime( BDateHidden.Value );**

**Stage 7 – Detecting an Invalid Delete**

1. Add a web form named, *Page3.aspx*.
2. Add a GridView to *Page3*.
3. Set the ID to *gvTeams*.
4. Configure it to show all four fields in the Teams table: TeamID, Name, CoachLName, CoachFName. DO NOT press, “Next”.
5. Generate INSERT, UPDATE, and DELETE commands (choose: Advanced)
6. Choose: OK, Next, Finish
7. Enable row deletion and sorting (Hint: fly-out)
8. Hide the *TeamID* field (Hint: fly-out, Edit Columns, select TeamID from Selected fields, and set Visible to False)
9. Apply a style them to the GridView (fly-out, Auto-format)
10. Build and run your page (Ctrl+Shift+B, Ctrl+F5). Do the following:
11. Press Delete on a team that has players (probably any of these teams have players: Hornets, Tigers, Turtles, Cougars). Notice that you get a server error and the page crashes. This is because the team has players. You cannot delete a row in a table (Teams) whose primary key (TeamID) exists as a foreign key in another table (Players). We will detect this in a subsequent step and stop the page from crashing.
12. Press Delete on a team with no players (probably the Jurks). The delete should succeed. If necessary, open the database in windows (not in VS) and add some teams (with no players), or drag a new copy of the database into VS if necessary so that you have some teams with no players.
13. Add a Label below the GridView and set the following properties:

|  |  |
| --- | --- |
| **Property** | **Value** |
| ID | lblDeleteStatus |
| Font/Bold | True |
| ForeColor | Red |

1. Select the GridView and display Events (select the lightning bolt icon in the Properties window). Double click the *RowDeleted* event. This creates the *gvTeams\_RowDeleted* event handler. Add the following code there:

/// <summary>

/// This event is fired after an attempt has been made to delete a row from the

/// GridView (and database). If the delete fails, an exception is thrown, which

/// is contained in the event arg, "e".

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void gvTeams\_RowDeleted(object sender, GridViewDeletedEventArgs e) {

// If delete was successful (i.e. no exception was thrown)

if (e.Exception == null) {

lblDeleteStatus.Text = "Team deleted";

}

// If delete failed (i.e. an exception was thrown), then display a message

// and set a flag indicating that the exception was handled.

else {

lblDeleteStatus.Text = "Team can't be deleted, Players exist.";

e.ExceptionHandled = true;

}

}

1. Build and run your page (Ctrl+Shift+B, Ctrl+F5). Try to delete a team with players. Then delete a team with no players. You should see the appropriate message displayed in the label. If necessary, open the database in windows (not in VS) and add some teams (with no players), or drag a new copy of the database into VS if necessary so that you have some teams with no players.
2. (Read, no action required) This is a slightly more detailed explanation of the comments in the code above.

When you try to delete a Team with Players, an exception is throw causing the page to crash. We can’t catch the exception, but we can intercept it. The GridView has a RowDeleted event that fires after an attempt to delete. The GridViewDeletedEventArgs that is passed into the event handler

protected void gvTeams\_RowDeleted(object sender, GridViewDeletedEventArgs e)

contains a reference to the Exception:

e.Exception

that is *null* if the deletion was successful. Otherwise, we can set the status of the exception to be “handled” allowing the page to not crash:

e.ExceptionHandled = true

**Stage 8 – Package Assignment for Submission**

1. Close VS and zip your *lab12\_lastName* solution folder and submit on Blazeview in the *Lab 12* dropbox.

**You’re done!**