**Lab 11 – 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**

The tutorial shows how to insert a new row into the database and update the GridView as shown in Figure 1 below. The GridView that ships with VS does not have a built-in insert feature. You can write code to embedded that ability. Instead, we take a simpler approach by coding the insert feature outside the GridView. We also show how to confirm a delete on the client and add a validator inside the GridView to restrict the Jersey number (PNumber) to be between 1 and 99.

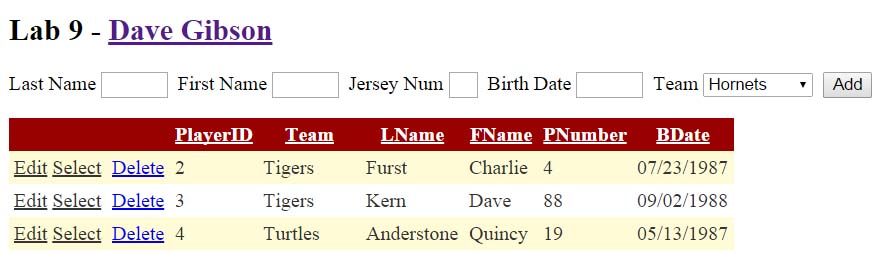


Figure 1 – Design to allow Insertion into a GridView.

**Note: I just did this Lab (4/9/2020) and most of it I had to do twice as it didn’t work the first time. Be extra slow, read everything carefully, and you won’t have a problem! If you do, just start over.**

**Lab Objectives**

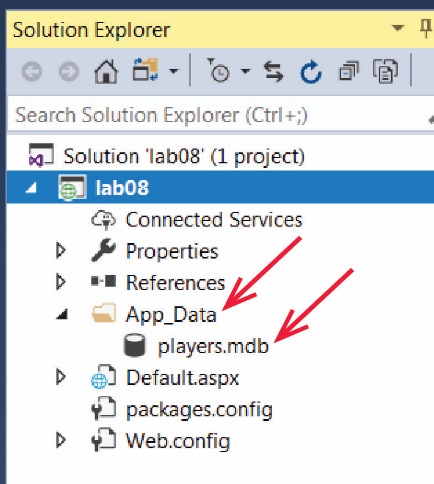
1. Allow the user to insert a row into a GridView.
2. Confirm a deletion from the GridView.
3. Utilize a validator inside the GridView to constrain the user’s input.

**Lab Organization**

There are 4 stages to complete Lab 11.

|  |  |
| --- | --- |
| Stage | Description |
| 1 | Add Insert Feature |
| 2 | Confirming a Delete on the Client |
| 3 | Adding a Validator |
| 4 | Add a Select Event |

**Stage 1 – Add Insert Feature**

1. Locate a copy of *players.zip* from Lab 8 (or 9 or 10)
2. Create your *lab11* project (solution folder must be named *lab11\_lastName*).
3. Do the following:
4. Right-click the project node in the SE and choose: Add, Add ASP.NET Folder, App\_Data

Note: Do not add a regular folder with this name. Make sure you add an *ASP.NET Folder.*

1. Drag the database, *players.mdb* into the *App\_Data* folder.
2. Add a web form named, *Default.aspx*.
3. Add this markup to the Source inside the *form* tags (you can delete the *div* tags or place this code inside them):

<h2>Lab 11</h2>

<p>Last Name

<asp:TextBox ID="txtLName" runat="server" Width="50px"></asp:TextBox>

&nbsp;First Name

<asp:TextBox ID="txtFName" runat="server" Width="50px"></asp:TextBox>

&nbsp;Jersey Num

<asp:TextBox ID="txtPNumber" runat="server" Width="20px"></asp:TextBox>

&nbsp;Birth Date

<asp:TextBox ID="txtBDate" runat="server" Width="50px"></asp:TextBox>

&nbsp;Team

<asp:DropDownList ID="ddTeams" runat="server">

</asp:DropDownList>

&nbsp;<asp:Button ID="btnAdd" runat="server" Text="Add" />

</p>

<asp:GridView ID="gvPlayers" runat="server">

</asp:GridView>

1. Configure the DropDownList above to be bound to a DataSource which displays the *Name* as the Text and *TeamID* as the Value (from the Teams table). Name the DataSource, *dsTeams.* See Lab 9 if you need help with this.

Note: remember that we had to change the path to the database in *Web.config* in Lab 9 from:

<connectionStrings>

<add name="playersConnectionString"

connectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=&quot;D:\eDataClasses\CS 3340 - Spring 2020\topics\04\_Database\_1\labs\lab08\_2\lab08\_gibson\App\_Data\players.mdb&quot;"

providerName="System.Data.OleDb" />

</connectionStrings>

To:

<connectionStrings>

<add name="playersConnectionString"

connectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=|DataDirectory|\players.mdb"

providerName="System.Data.OleDb" />

</connectionStrings>

**I’ve now found out there is an easier way to do this. On the dialog below, instead of choosing: New Connection, select the drop down and you should see *players.mdb*. If so, it will create the path properly.**

1. Build and run your page (Ctrl+Shift+B, Ctrl+F5). Make sure the teams are displayed in the drop down.
2. Configure the GridView above to be bound to a DataSource which displays all the fields of Players table just as we did in Lab 10. The GridView should have edit and delete features (you can omit the select feature). Notes:

* **Open the directions for Lab 10 and follow along to accomplish this.**
* This will probably take some trial and error. Don’t hesitate to completely start over if things go wrong. By start over, I mean, delete the GridView and any DataSources you have created for it.
* The GridView will need a DataSource which should be named *dsPlayers*.
* Don’t forget that the DataSource needs to have the Insert, Update, and Delete generated. Then, you go back through add use a custom Sql statement with an inner join.
* You can reuse the ConnectionString created in the step above.
* When you configure the DropDownList that is inside the GridView (for changing the Team when in Edit mode), you can reuse the DataSource (*dsTeams*) you created in the step above.
* **Don’t forget to add the DataKeyNames attribute in the GridView tag.**

1. Build and run your page (Ctrl+Shift+B, Ctrl+F5). Verify that Edit and Delete work properly.
2. (Read, no action required) The DataSource, *dsPlayers* that you created above has Insert, Update, and Delete statements. The GridView itself uses the Update and Delete statements to carry out those actions. However, the Insert statement is not used as the GridView does not have a mechanism to insert a new row in the database (third-party gridview-like controls do have this feature). However, we can use the DataSource’s INSERT statement programmatically to insert a row, which is what we do next.
3. Create an Add button event handler by double-clicking the Add button in Design mode and add this code:

// Connect parameters to data entry controls

dsPlayers.InsertParameters["TeamID"].DefaultValue = ddTeams.SelectedValue;

dsPlayers.InsertParameters["LName"].DefaultValue = txtLName.Text;

dsPlayers.InsertParameters["FName"].DefaultValue = txtFName.Text;

dsPlayers.InsertParameters["PNumber"].DefaultValue = txtPNumber.Text;

dsPlayers.InsertParameters["BDate"].DefaultValue = txtBDate.Text;

// Insert player

dsPlayers.Insert();

// Clear fields

txtLName.Text = string.Empty;

txtFName.Text = string.Empty;

txtPNumber.Text = string.Empty;

txtBDate.Text = string.Empty;

1. (Read, no action required) The page will bomb when trying to add a new Player. Try it if you like. We will need to do some delicate work to fix things.
2. Open the Source and find the SqlDataSource tag. Find the *InsertCommand* (partially shown below). I recommend putting line breaks in so that you can see the command clearly on the screen.

<asp:SqlDataSource ID="dsPlayers" runat="server"

...

InsertCommand="INSERT INTO [Players] ([PlayerID], [TeamID], [LName], [FName], [PNumber], [BDate]) VALUES (?, ?, ?, ?, ?, ?)"

...

</asp:SqlDataSource>

Remove the two items highlighted in yellow above. Remember that we do NOT ever try to add or change a primary key. The result should look as shown below:

InsertCommand="INSERT INTO [Players] ([TeamID], [LName], [FName], [PNumber], [BDate]) VALUES (?, ?, ?, ?, ?)"

1. Also in the SqlDataSource tag, find the *InsertParameters*. Delete the *PlayerID* parameter highlighted below.

<asp:SqlDataSource ID="dsPlayers" runat="server"

...

<InsertParameters>

<asp:Parameter Name="PlayerID" Type="Int32" />

<asp:Parameter Name="TeamID" Type="Int32" />

<asp:Parameter Name="LName" Type="String" />

<asp:Parameter Name="FName" Type="String" />

<asp:Parameter Name="PNumber" Type="Int32" />

<asp:Parameter Name="BDate" Type="DateTime" />

</InsertParameters>

...

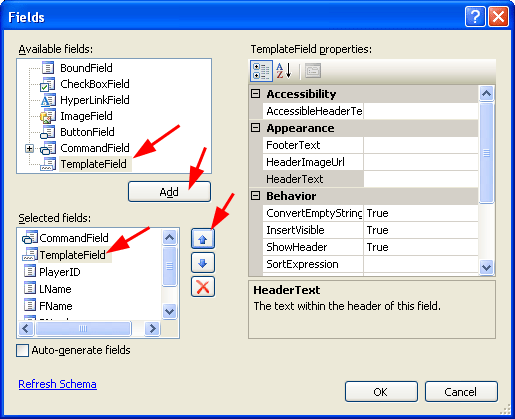
</asp:SqlDataSource>

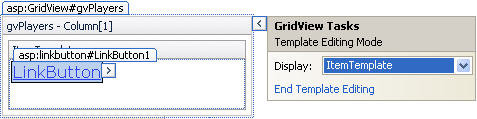
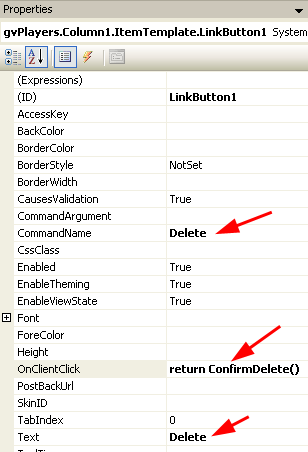
1. Run your page and verify that Insert, Update, and Delete work. Hint: add a Player with a last name, “aa” and some other random data, press Add, then sort on Last Name to see that it is there. Then, you can Edit it, Save, and then Delete it to check that functionality.

**Stage 2 – Confirming a Delete on the Client**

The next feature we add is to require the user to confirm a delete from the GridView. To do this we will remove the “automatic” Delete button and put in a new one that allows us to intercept the deletion and do a custom confirmation.

1. Display the GridView Tasks for the GridView and **uncheck** “Enable Delete”.



1. Display the GridView Tasks menu, choose “Edit Columns...” to display the Fields dialog, then:
2. Select *TemplateField* from the “Available fields” and choose “Add”. This will be our new Delete button.
3. Select *TemplateField* from the “Selected fields” and then click the up arrow to raise it to just below “Edit, Update, Cancel” (which in the figure on the rights says “*CommandField”.* This moves the *TemplateField* left in the GridView display).
4. Choose “OK”.
5. Display the GridView Tasks menu and choose “Edit Templates” and do the following:
6. Make sure the *ItemTemplate* is displayed. Put a *LinkButton* in the *ItemTemplate* (i.e. drag a *LinkButton* from the ToolBox).
7. Select the LinkButton and set the properties shown below:
8. Set the *CommandName* property to “Delete”. This synchs the LinkButton’s Click event with GridView’s *DeleteCommand*, which contains the SQL for a Delete statement.
9. Set the *OnClientClick* property to “return ConfirmDelete()”. This specifies a JavaScript function (that runs on the client) that returns *true* or *false*. If *true,* the page posts-back and the deletion is carried out on the server. If *false,* the page does not post back (*i.e.* the deletion is aborted). In this case, the JavaScript function name is *ConfirmDelete* which we will add to the Source in the next step.
10. Set the *Text* property to “Delete” which is the text that appears on the LinkButton.
11. Display the GridView Tasks menu and “End Template Editing.”
12. Add the JavaScript function below in the Source (in the head section). You could have a more elaborate function that displays the values of the row that is about to be deleted; however, that is a good bit more involved.

<script type="text/javascript">

function ConfirmDelete() {

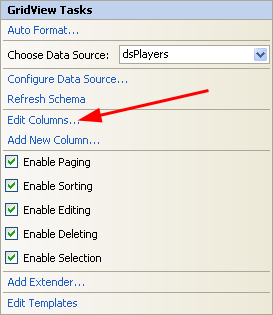
return window.confirm("Are you sure you want to delete this entry?");

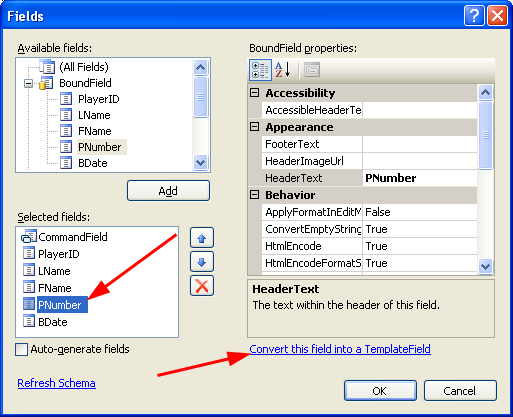
}

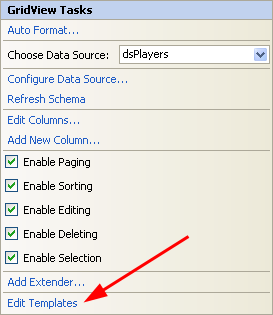
</script>

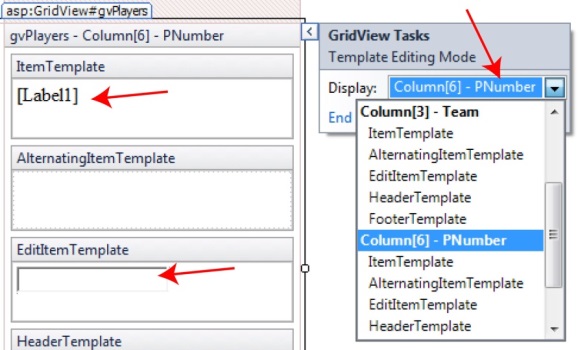
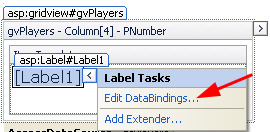
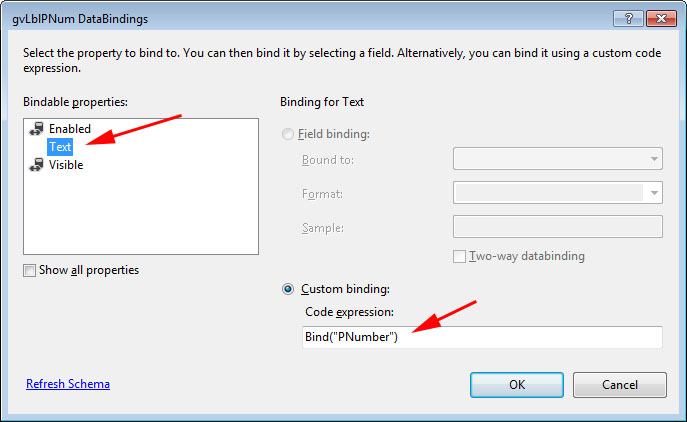
1. Build/Run (Ctrl+Shift+B, Ctrl+F5). Test the confirmation of deletion.

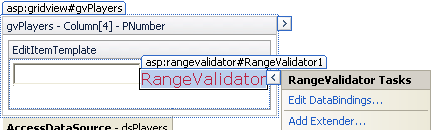
**Stage 3 – Adding a Validator**

We will add a RangeValidator to the Jersey Number (PNumber) field in the GridView when in Edit mode so that a Jersey Number can only be between 1 and 99.

1. Display the “GridView Tasks” menu and do the following:
2. Choose “Edit Columns.”
3. Select “PNumber” and “Convert this field into a Template Field” (lower right). Then choose OK.
4. Display the “GridView Tasks” menu and do the following:

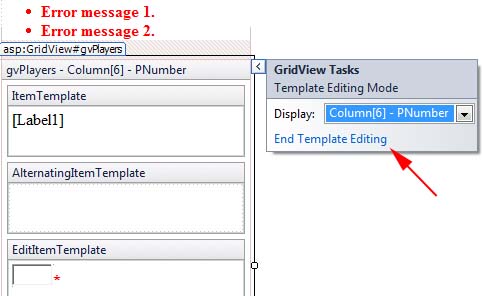


1. Choose “Edit Templates.”
2. Select “Column[6] – PNumber” from the *Display* drop down.
3. Select the Label in the ItemTemplate and supply it the ID: *lblGvPNum*. We will use this later.
4. This step is for information only.
5. Select the label, then click the arrow icon, then select “Edit DataBindings.”
6. The display will look like this. We will not change anything here. Notice how the *Text* property is bound to the *PNumber* field under “Custom binding”. Choose Cancel.
7. Select the TextBox in the EditItemTemplate (see Figure 2c above) and provide the ID, *txtGvPNum*.
8. (This step is for information only). Select the TextBox in the EditItemTemplate and choose: “Edit DataBindings”. Inspect the Text property binding and then Cancel.



1. Drag a “RangeValidator” from the ToolBox into the *EditItemTemplate*, beside the Textbox. Set the following properties:

|  |  |
| --- | --- |
| **Property** | **Value** |
| ControlToValidate | *txtGvPNum* |
| ErrorMessage | Jersey Number must be between 1 and 99. |
| Font/Bold | True |
| ForeColor | Red |
| MaximumValue | 99 |
| MinimumValue | 1 |
| SetFocusOnError | True |
| Text | \* |
| Type | Integer |
| Width | 20 |

1. Choose: End Template Editing.
2. Drag a “ValidationSummary” from the ToolBox to just above the GridView (and below the Add feature). Set the following properties:

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

1. Open *Web.config*. Add the *appSettings* node above (or below) the *system.web* node.

**<appSettings>**

**<add key="ValidationSettings:UnobtrusiveValidationMode" value="None" />**

**</appSettings>**

1. Build/Run (Ctrl+Shift+B, Ctrl+F5). Edit a player and try to change a PNumber to an invalid value.
2. (Read, no action required) View the Source for the page. Find the GridView and then find the PNumber TemplateField inside the GridView. Study the structure. Note the two *Bind* statements as shown highlighted in yellow below.

<asp:TemplateField HeaderText="PNumber" SortExpression="PNumber">

<EditItemTemplate>

<asp:TextBox ID="txtGvPNum" runat="server"

Text='<%# Bind("PNumber") %>'></asp:TextBox>

<asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="txtGvPNum"

ErrorMessage="Jersey Number must be between 1 &amp; 99"

Font-Bold="True" ForeColor="Red" MaximumValue="99" MinimumValue="2"

Type="Integer" SetFocusOnError="True" Width="20px">\*</asp:RangeValidator>

</EditItemTemplate>

<ItemTemplate>

<asp:Label ID="lblGvPNum" runat="server" Text='<%# Bind("PNumber") %>'></asp:Label>

</ItemTemplate>

</asp:TemplateField>

**Stage 4 – Add a Select Event**

1. Add a TextBox below the GridView. Provide the ID: *txtMsg*, and set “TextMode” to Multiline in the Properties Window.
2. Display the “GridView Tasks” menu and *Enable Selection*
3. Do the following:
4. Select the GridView
5. Select the lightning bolt icon in the Properties Window
6. Double-click the *SelectedIndexChanged* event
7. Supply this code in the event handler:

// Bound Fields

int playerID = Convert.ToInt32(gvPlayers.SelectedRow.Cells[2].Text);

string lName = gvPlayers.SelectedRow.Cells[4].Text;

string fName = gvPlayers.SelectedRow.Cells[5].Text;

//int jerseyNum = Convert.ToInt32(gvPlayers.SelectedRow.Cells[5].Text);

DateTime bDate = Convert.ToDateTime(gvPlayers.SelectedRow.Cells[7].Text);

// Template Fields

Label lblTeamName = (Label)gvPlayers.SelectedRow.FindControl("lblGvTeamName");

string teamName = lblTeamName.Text;

Label lblJerseyNum = (Label)gvPlayers.SelectedRow.FindControl("lblGvPNum");

int jerseyNum = Convert.ToInt32(lblJerseyNum.Text);

// Hidden Field

//HiddenField teamIDHidden = (HiddenField)gvPlayers.SelectedRow.FindControl("TeamIDHidden");

//string teamID = teamIDHidden.Value;

string output = String.Format("PlayerID={0}, LName={1}, FName={2}, " +

"JerseyNum={3}, BDate={4}, " +

"Team={5}",

playerID, lName, fName,

jerseyNum, bDate.ToShortDateString(),

teamName);

txtMsg.Text = output;

**Note: In the previous lab, we put a HiddenField to store the TeamID. This could be added if needed.**

1. Build/Run (Ctrl+Shift+B, Ctrl+F5). Test thoroughly.

**Stage 5 – Package Assignment for Submission**

1. Close VS and zip your *lab11\_lastName* solution folder and submit on Blazeview in the *Lab 11* dropbox.

**You’re done!**