**Lab 10 – 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…”

**Lab Objectives**

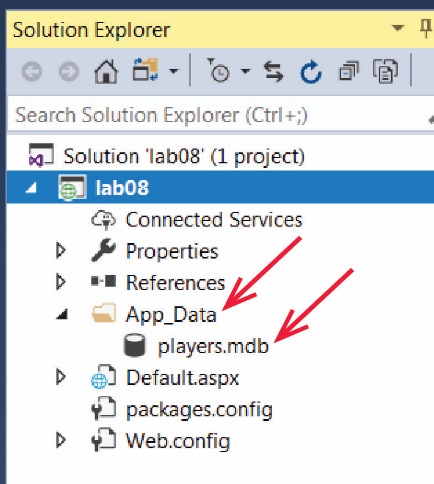
1. Be able to bind data to a GridView and provide edit, delete, select, sort, and pagination features.
2. Write a SelectedIndexChanged event handler for a GridView.
3. Write code to access the value in a BoundField in a selected row of a GridView
4. Configure a DataSource to select data from a database table.
5. Provide editing of a foreign key via a DropDownList in a GridView.
6. Use the Query Builder.
7. Know how to use a TemplateField and why we use them.
8. Know how to use a HiddenField and why we use them.
9. Understand how and why to use data bindings.
10. Write code to access the value in a TemplateField and HiddenField in a selected row of a GridView

**Lab Organization**

There are 7 stages to complete Lab 10.

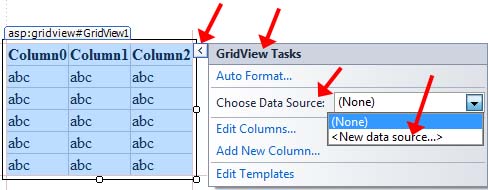
|  |  |
| --- | --- |
| Stage | Description |
| 1 | Bind Players Table to GridView |
| 2 | Explore Structure of GridView and SqlDataSource |
| 3 | Enhance the GridView |
| 4 | Display TeamID in GridView |
| 5 | Display Team Name in GridView |
| 6 | Display Team Drop Down in GridView When in Edit Mode |
| 7 | Programming the Select Event |

**Stage 1 – Bind Players Table to GridView**

1. Locate a copy of *players.mdb* from Lab 8 or 9.
2. Create your *lab10* project (solution folder must be named *lab10\_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.*

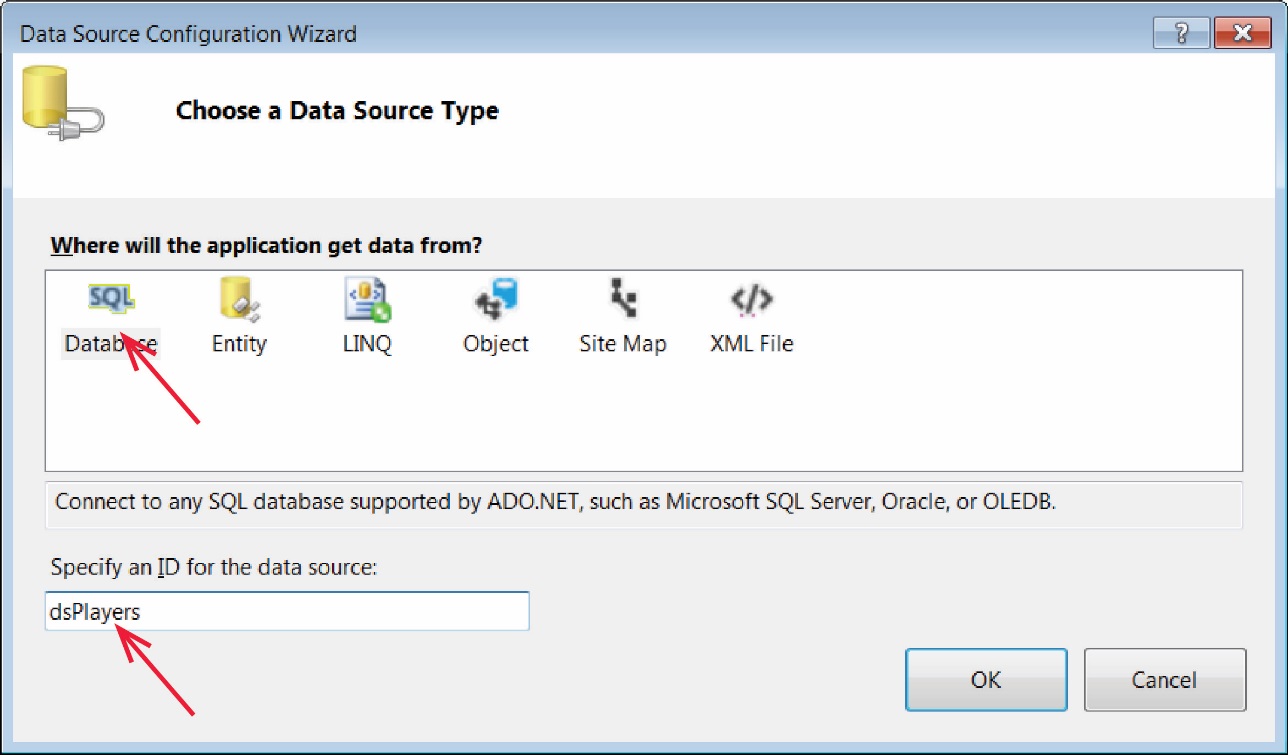
1. Drag the database, *players.mdb* into the *App\_Data* folder.
2. Add a web form named, *Default.aspx*.
3. Do the following:
4. Display the Toolbox, open the Data section.
5. Drag a GridView control to the page.
6. **Set the ID to “gvPlayers”.**



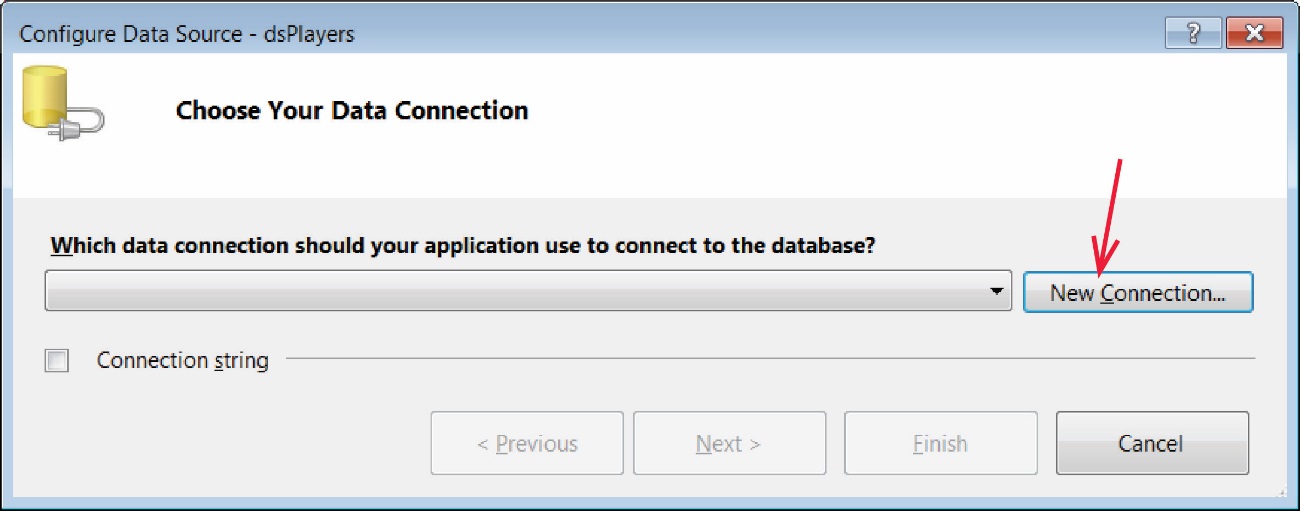
1. Do the following:
2. Select the GridView
3. Expand the fly-out menu which displays the *GridView Tasks* menu.
4. Choose *<New data source…>*

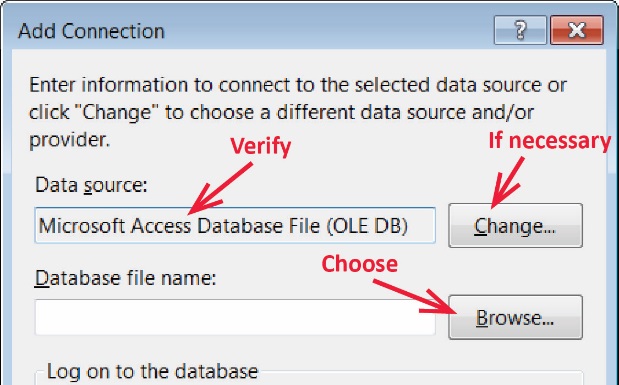
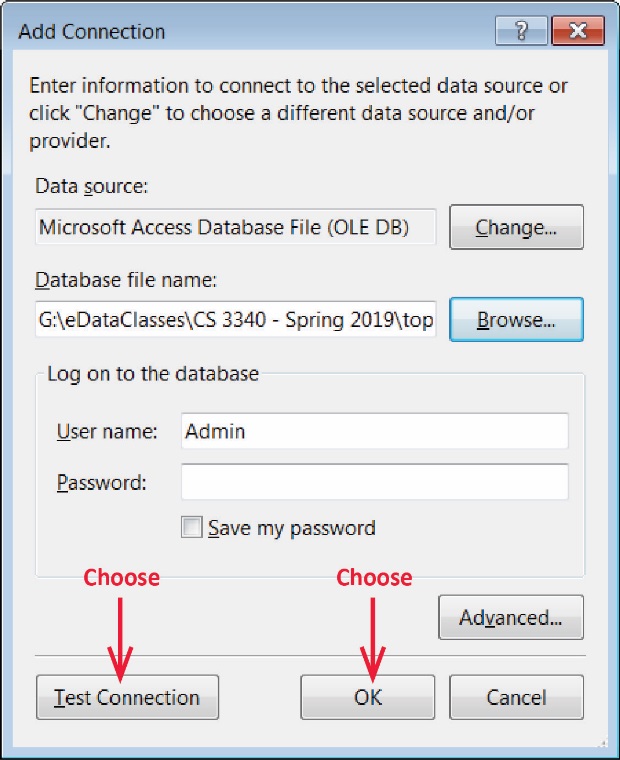
* Remember how to “display the GridViewTasks menu” as many subsequent steps will refer to this.

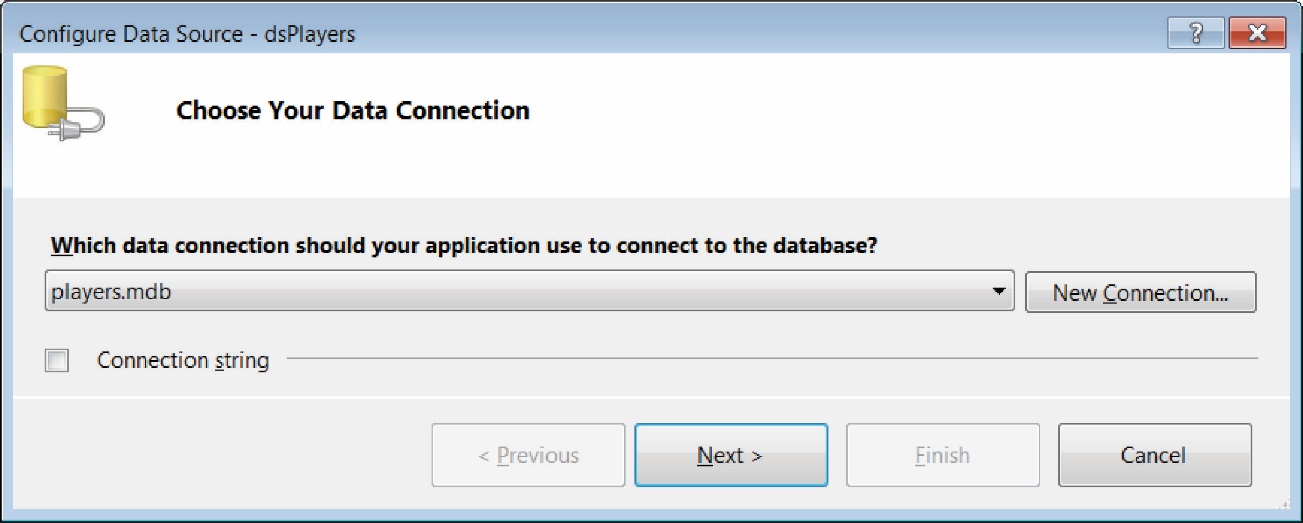
1. Do the following:
2. Select the “SQL Database” icon (left most)
3. Provide the ID, *dsPlayers*
4. Choose OK.



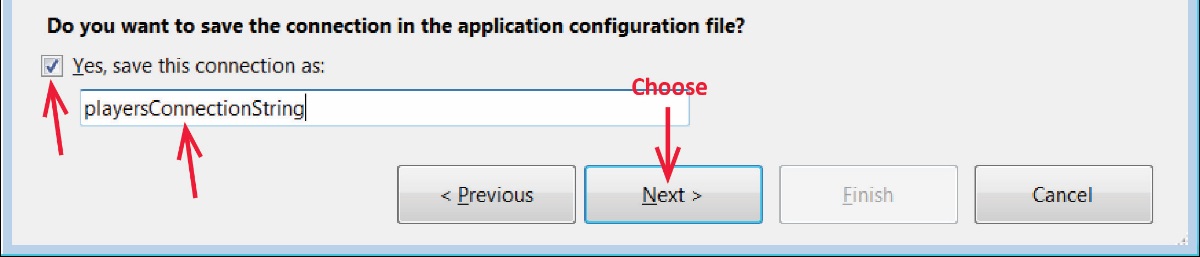
1. Choose: New Connection.



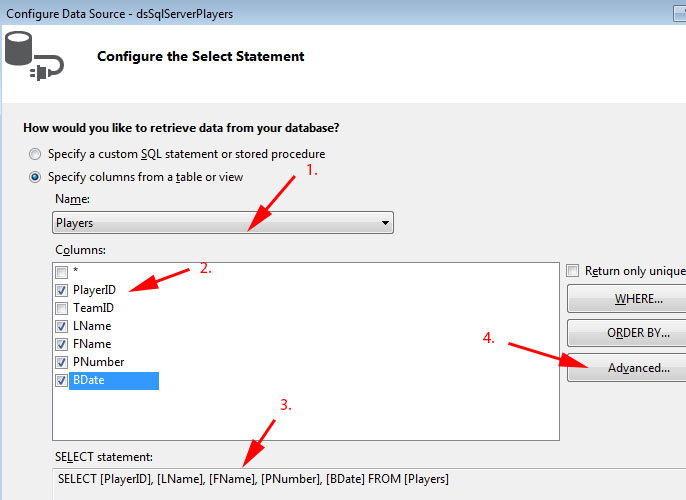
1. Do the following:
2. Verify that the *Data Source* is, “Microsoft Access Database File” (it will NOT show, “(OLE DB)” as shown in the figure on the right). If not, choose: Change and select it from the dialog.
3. ~~Choose: Continue (NOT “Browse” as shown in the figure on the right)~~
4. The Add Connection dialog will change a bit. Choose: Browse to Navigate to your “Database file name”, select it, and choose: Open. Note that the path is hard-coded. We will fix this later.
5. Choose: Test Connection and verify that it succeeded, and then choose: OK
6. Select *players.mdb* from the drop down (if necessary) and choose: Next.



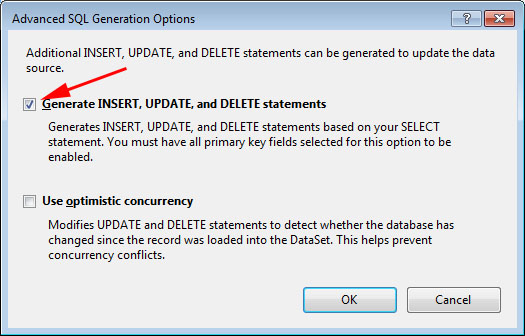
1. Select the checkbox to save the connection string, name the connection: *playersConnectionString* and choose: Next.

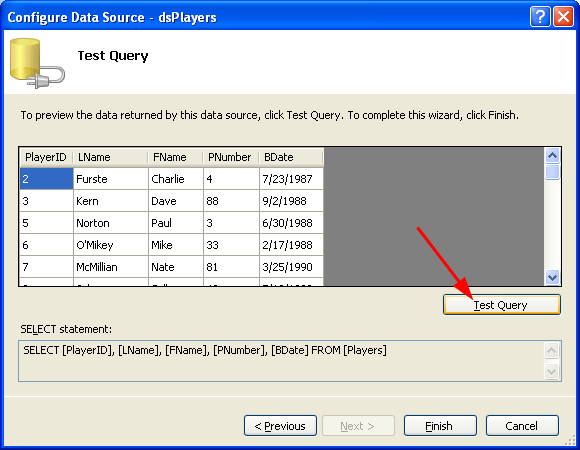
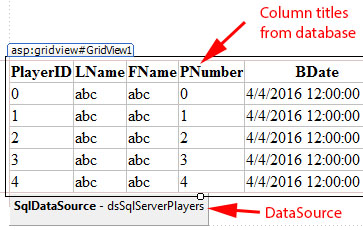


1. As shown in the figure below, do the following:
2. Select the *Players* table.
3. Select all fields except *TeamID.*
4. Note the SQL statement that was generated.
5. Choose *Advanced*



1. Select, “Generate INSERT, ... statements” and choose “OK”.



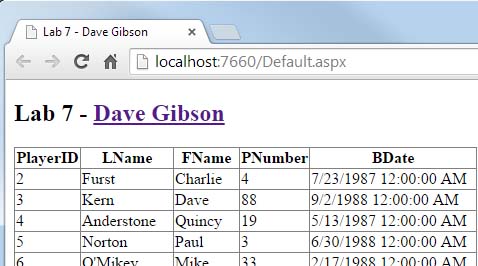
1. On the “Configure Data Source” dialog that appears, choose, “Next.”
2. Do the following:
3. Choose “Test Query.” The data from the *Players* table should be displayed.
4. Choose Finish.
5. All dialogs should be closed now and you should be back at your page in Design mode. The page should be similar to what is shown on the right (the name of the data source will be *dsPlayers*, however)
6. (Read, no action required) Usually, the directions say, “View your page”. In response you right-click and choose: View in Browser. This actually compiles your page first. An alternate way to do this, that I find useful is to compile your page first:

Choose: Build, Build (or Ctrl+Shift+B)

And then:

Choose: Debug, Start Without Debugging (or Ctrl+F5)

In subsequent directions, I refer to this as “Build and run your page”, which is equivalent to “View your page”.

1. Build and run your page (Ctrl+Shift+B, Ctrl+F5). The display should show the players in the *Players* table similar to what is shown on the right.

**Stage 2 – Explore Structure of GridView and SqlDataSource**

1. Return to VS and view your page in “Source” mode. The numbers in parentheses in the text below refers to the numbers in the figure below. Examine the markup for the GridView (1) as shown below. Note:

* The *DataKeyNames* (2) attribute of the GridView defines the primary key for the data being displayed. It must be present; if it is not there add it.
* The *DataSourceID* (3) attribute of the GridView defines the data source that will be used to bind the data.
* The GridView is composed of *Columns*. Here, each column is a *BoundField* (4) Later, we will replace some of the *BoundFields* with *TemplateFields* which allows us to customize the display.
* The *DataField* (5) attribute shows the name of the field in the database that is bound to a column.
* The *HeaderText* (6) attribute shows the title for the column that will be displayed in the GridView. The default is the name of the database field, but you can change this.
* The *SortExpression* (7) attribute shows the name of database field to sort on when the column header is pressed in the GridView. Normally, this will be the same as the *DataField* .
* The *InsertVisible* and *ReadOnly* (8) attributes for the “PlayerID” column specify that this field cannot be changed (because it is the primary key of the database table).



1. Find the *SqlDataSource* markup; it scrolls far to the right. You can force line breaks to make it easier to read if you like. Note:

* The “<%$” markup indicates that the statement inside the markup is code that is executed when the page is loaded.
* The *ConnectionString* and *ProviderName* are obtained from *Web.config* at run time.
* The next four attributes, *DeleteCommand*, *InsertCommand*, *SelectCommand,* and *UpdateCommand* provide parameterized SQL statements. The “?” symbol is used to represent parameters which are defined next. This is different than when we considered parameterized queries in the Database 1 notes. I suspect this is a newer or alternate form.
* The parameters for each command are defined. See the <DeleteParameters>, <InsertParameters>, and <UpdateParameters>.



1. Open *Web.config* and note the *connectionStrings* node. And notice that the path to the database is hard-coded.

<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>

Carefully change it so that it looks exactly as shown below (highlighted):

<connectionStrings>

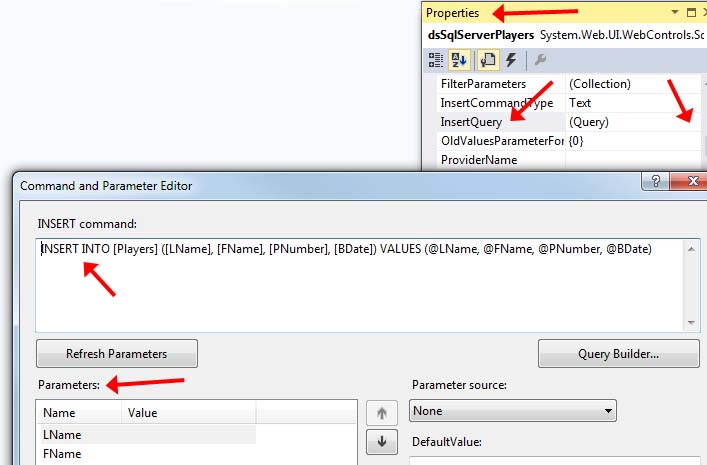
<add name="playersConnectionString"

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

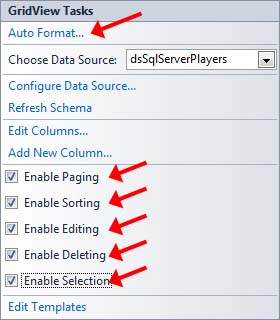
providerName="System.Data.OleDb" />

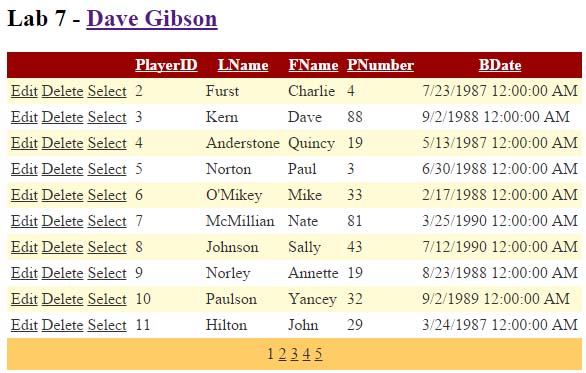
</connectionStrings>

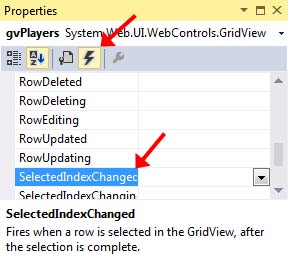
1. Run your page and make sure it still works.
2. Display your page in Design mode. Here, we will just look under the covers, but won’t do anything to affect our page. In a more advanced setting, you may need to change/modify things. However, for this class, I don’t think we will need to.
3. Select the SqlDataSource component (probably below the GridView) and view the properties window (as shown below).
4. Find the *InsertQuery* property and click the ellipses which displays the “Command and Parameter Editor”
5. Note the INSERT statement and the Parameters and then Cancel.
6. Repeat for the *DeleteQuery* and *UpdateQuery.*



**Stage 3 – Enhance the GridView**

1. Display the *GridView Tasks* menu and do the following:
2. Select *Auto Format* and Select a theme. (I chose “Colorful”).
3. Choose, *OK.*
4. Enable the following options: Paging, Sorting, Editing, Deleting, Selection.
5. Build and run your page (Ctrl+Shift+B, Ctrl+F5). Do the following:
6. Press a column heading several times to see how sorting works.
7. Press “Edit” on a row, change the value of one of the fields and then press “Update”. Note that *PlayerID* is not editable.
8. Press “Delete” on one of the rows. It will delete immediately – no confirmation. We will learn how to do delete confirmation in another lab.
9. Press the pagination links at the bottom of the GridView.
10. You can press “Select” for a row, but it doesn’t do anything yet other than highlight the row.
11. Close the page and return to VS.



1. Next, we will create an event handler for when the *Select* button is pressed. Do the following:
2. Select the GridView and then go to the Properties Window.
3. Choose the Events icon.
4. When the events are displayed, double-click the “SelectedIndexChanged” event. This will create an event-handler code stub in the code-behind file. When the Select button is pressed on a row in the GridView, the *SelectedIndex* property changes to indicate the row that was selected and then the *SelectedIndexChanged* event occurs.

**protected void gvPlayers\_SelectedIndexChanged(object sender, EventArgs e) {**

**}**

1. Return to Design mode and do the following:
2. Add a TextBox to your page below the GridView.
3. Set the *ID* propertyto “txtMsg”
4. Set the *TextMode* property to *“MultiLine”*
5. Stretch the text box to make it larger.
6. Open the code-behind file and add the code shown below to the *SelectedIndexChanged* event handler. This code takes the values in the selected row and puts them in a text box. In a real application, we would probably do something more useful. Note the following by studying the code below:
7. The *GridView* was a *SelectRow* property which (of course!) represents the row that was selected by the user.
8. The *SelectedRow* property has a *Cells* collection to represent each of the columns in the *GridView*.
9. The Cells (columns) are numbered: 0, 1, 2, *etc.* from the left. The first cell, Cell[0] contains the controls (Edit, Delete, Select). The second cell, Cell[1] contains the player ID*.* The third cell, Cell[2] contains the last name, *etc.* The code below will take the values in the selected row and display them in the TextBox. Study the code carefully.

**protected void gvPlayers\_SelectedIndexChanged(object sender, EventArgs e)**

**{**

**// Getting value from a bound field.**

**int playerID = Convert.ToInt32(gvPlayers.SelectedRow.Cells[1].Text);**

**string lName = gvPlayers.SelectedRow.Cells[2].Text;**

**string fName = gvPlayers.SelectedRow.Cells[3].Text;**

**int jerseyNum = Convert.ToInt32(gvPlayers.SelectedRow.Cells[4].Text);**

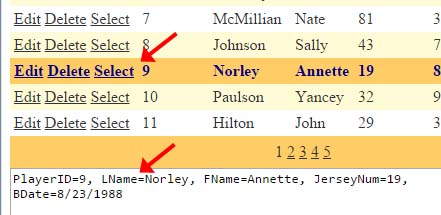
**DateTime bDate = Convert.ToDateTime(gvPlayers.SelectedRow.Cells[5].Text);**

**string output = String.Format("PlayerID={0}, LName={1}, FName={2}, JerseyNum={3}, BDate={4}",**

**playerID, lName, fName, jerseyNum, bDate.ToShortDateString());**

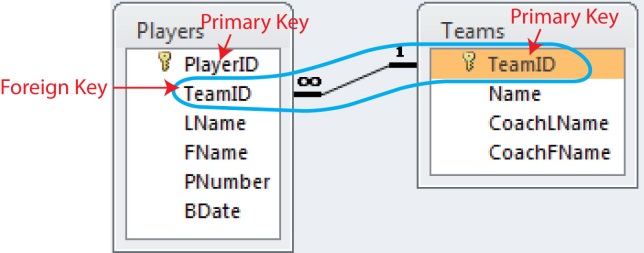
**txtMsg.Text = output;**

**}**

1. Build and run your page (Ctrl+Shift+B, Ctrl+F5). Press “Select” on a row and see the results in the TextBox.

**Stage 4 – Display TeamID in GridView**

1. (Read, no action required)
2. Remember that each Player is associated with a Team through the foreign key – primary key relationship between the two tables as shown below.



1. With the current page, we are not displaying the team the player is on (nor the TeamID) and when we edit, we cannot change the team a player is on. In this stage, we fix these things.
2. In Figure 1 below, we display the Players table as before, but also display the Team name in place of the TeamID.

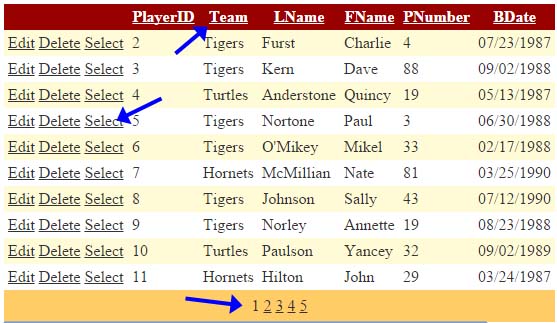


Figure 1 – GridView in display mode.

1. In Figure 2 below, when the GridView is in edit mode, we display a drop down showing all the team Names. To make this happen is a little detailed, so follow along closely.

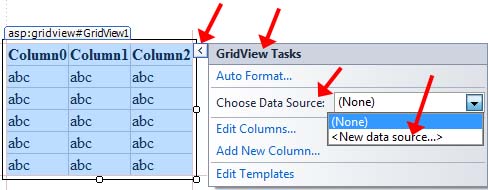


Figure 2 – GridView in edit mode.

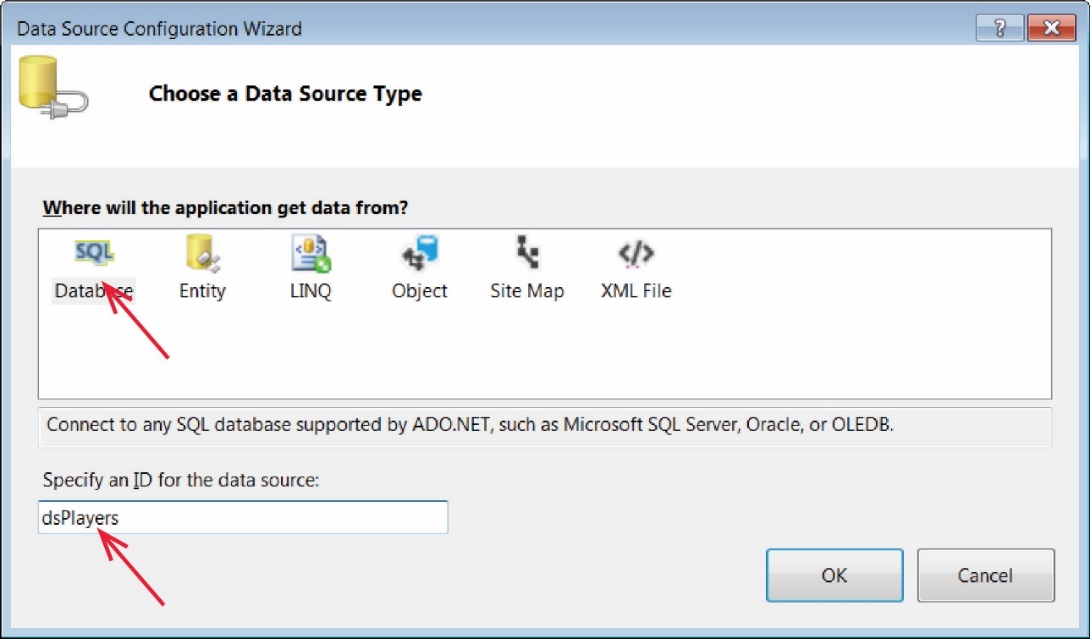
1. In this stage we will start a new page and repeat many of the steps we did in the earlier stages. We could build on the previous page, but I feel it will be better to start a new page.

**Warning: If some things don’t work correctly in subsequent steps, simply start this Stage over.**

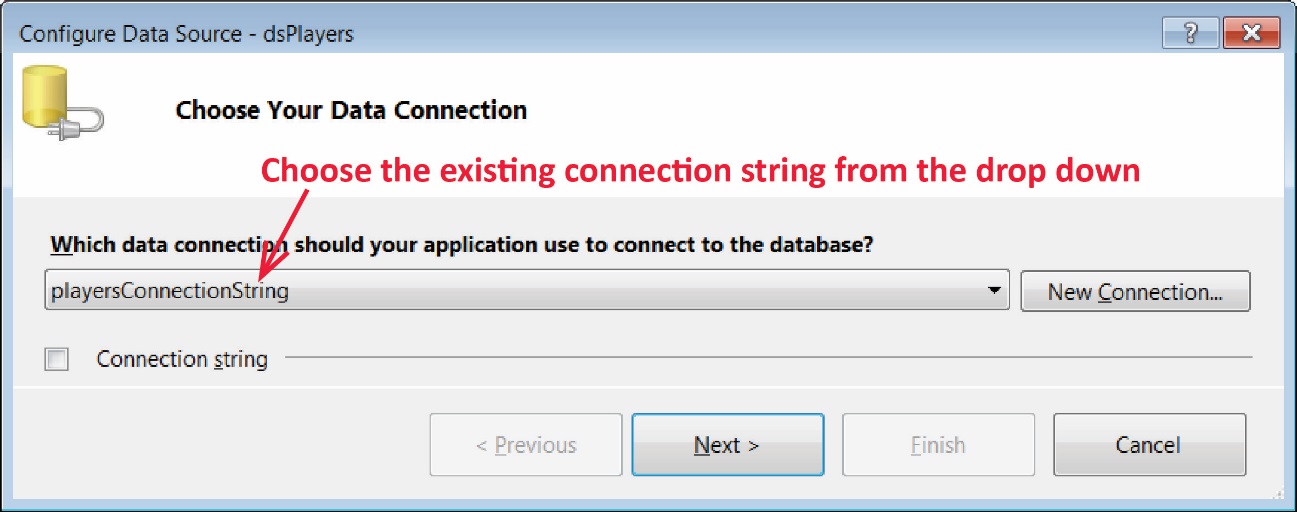
1. Create a web form named, *Page2.aspx*
2. Do the following:
3. Display the Toolbox, open the Data section.
4. Drag a GridView control to the page.
5. Set the ID to “gvPlayers”.



1. Do the following:
2. Select the GridView
3. Expand the fly-out menu which displays the *GridView Tasks* menu.
4. Choose *<New data source…>*
5. Do the following:

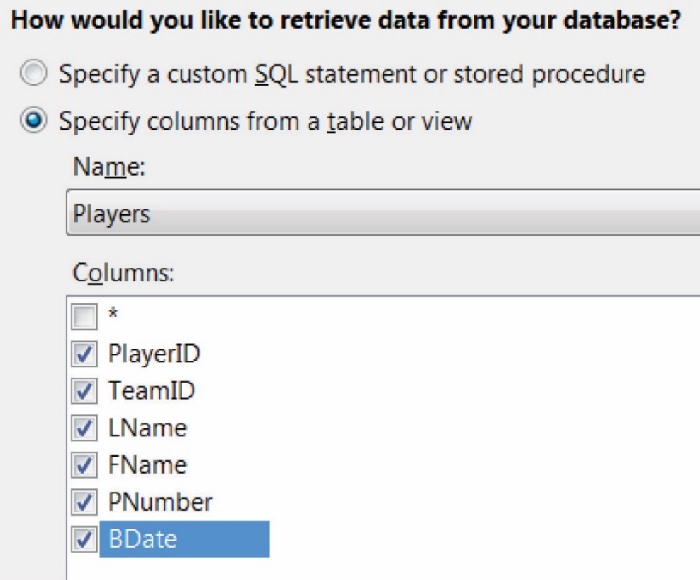


1. Select the “SQL Database” icon (left most)
2. Provide the ID, *dsPlayers*
3. Choose OK.
4. Choose the existing connection string, *playersConnectionString*, from the drop down. We could create a new one, but there is no need. Then, choose: Next.



1. Next, select each of the columns, from top to bottom so that they are all selected (**the order you select them is important as that is reflected in the Sql statement and affects subsequent steps in this lab**).

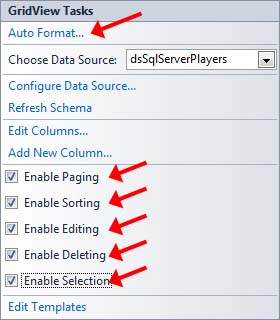
Note: this time we are selecting *TeamID*

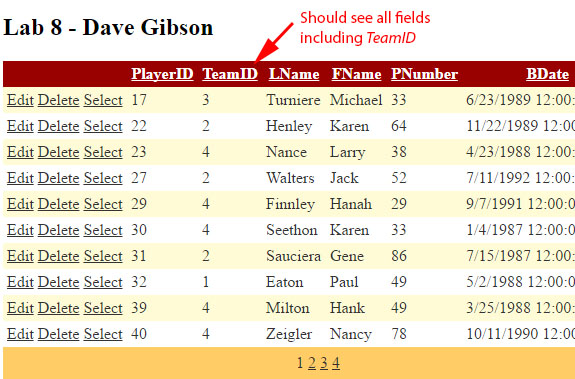


1. Choose: Advanced, Generate Insert, Update, and Delete Statements, and the OK.

Note:

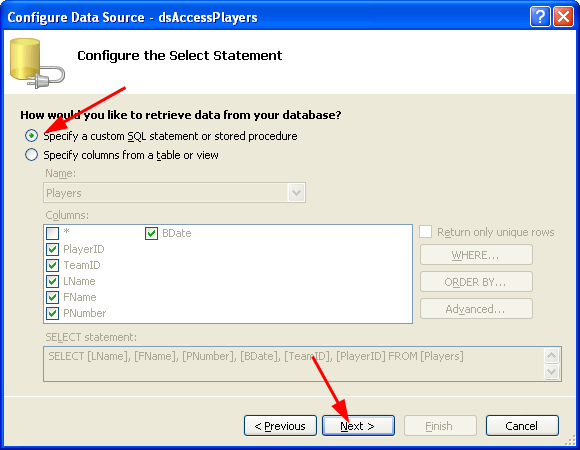
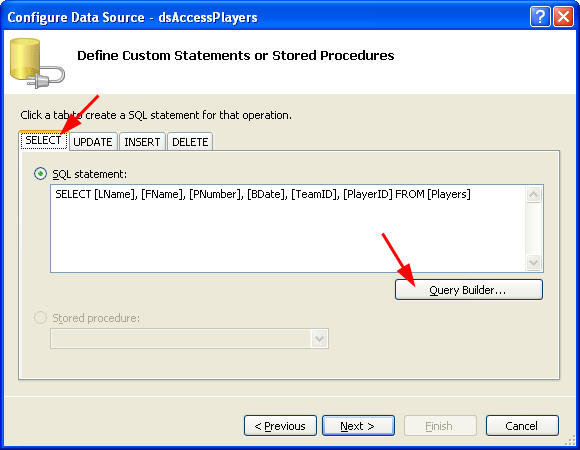
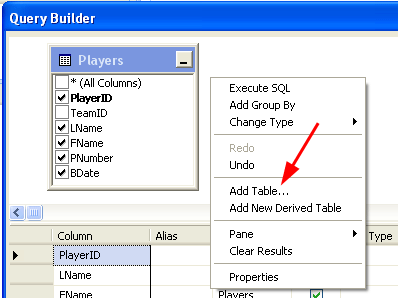
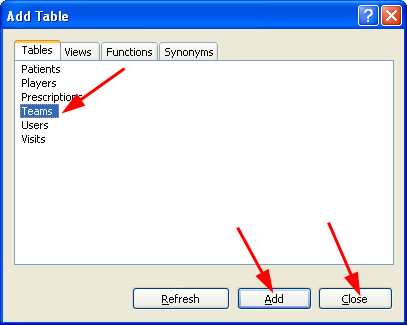
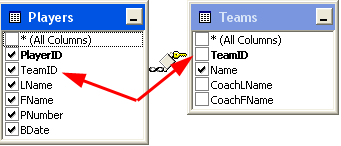
* In this Stage, we will display the TeamID. In the next Stage we will replace TeamID with Team Name.
* To select the Team Name, we need an Inner Join. However, when we do an Inner Join, the ability to “Generate Insert, Update, and Delete Statements” will no longer be available.
* Thus, this step was done to “trick” the wizard into creating the Insert, Update, and Delete statements, which we can modify manually in Stage 5.

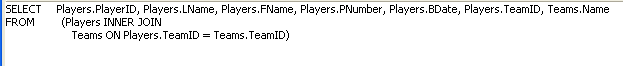
1. Choose: Next, Test Query, Finish.
2. Display the *GridView Tasks* menu and do the following:
3. View your page. Note the following:
4. You can Edit, but if you supply an incorrect TeamID, then the page will bomb. Experiment.
5. You can Delete. Experiment.
6. Select does nothing presently.

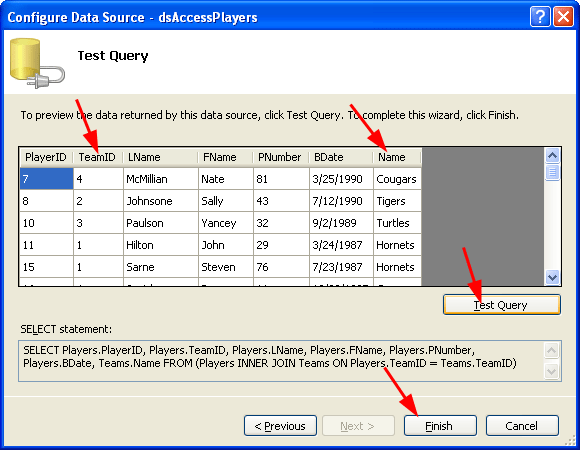


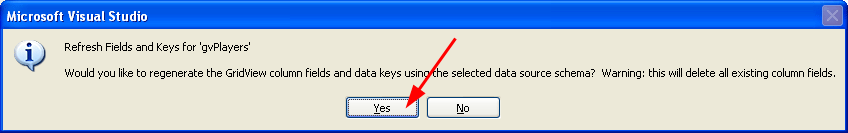
**Stage 5 – Display Team Name in GridView**

In this stage we will display the Team Name in the GridView by changing the Select statement to use an inner join on the Teams table.

1. Select your GridView and display the “GridView Tasks” menu.
2. Choose “Configure Data Source”. We are going to go back through this series of dialogs and change a few things.
3. Choose, “Next”.
4. Select “Specify a custom SQL Statement…”
5. Choose, “Next”.
6. Choose the “Select” tab
7. Select the “Query Builder...” button.
8. The Players table should be displayed. Stretch the dialog so you can see things clearly. Note: your dialog will show that TeamID IS selected, which is correct. The figure on the right shows (incorrectly) that it is not selected.
9. Right-click in a blank area and choose “Add Table...”.
10. Select the *Teams* table, then “Add”, then “Close”.
11. Our objective is to display the players along with their corresponding team “Name”. Select the fields shown in the two tables on the right.
12. Note that the Query Builder shows the Select SQL statement (now with an inner join). When complete, choose “OK” on the Query Builder. This will return you to the “Configure Data Source...” dialog. Choose, “Next”.

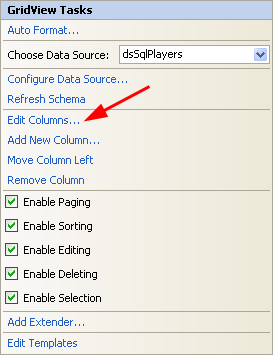
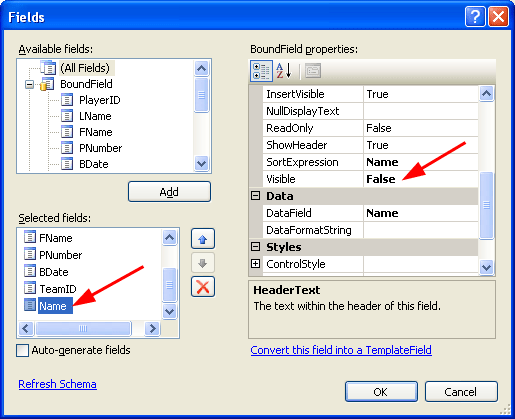


1. Choose, “OK” and then “Next”.
2. Choose “Test Query” and observe that the TeamID and team Name are now displayed.
3. Choose, “Finish”
4. You will get the message box shown below. Answer, “Yes”, because we have added fields to the query and want them to be displayed.

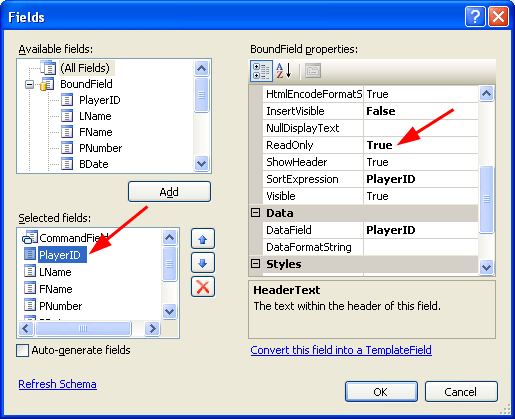


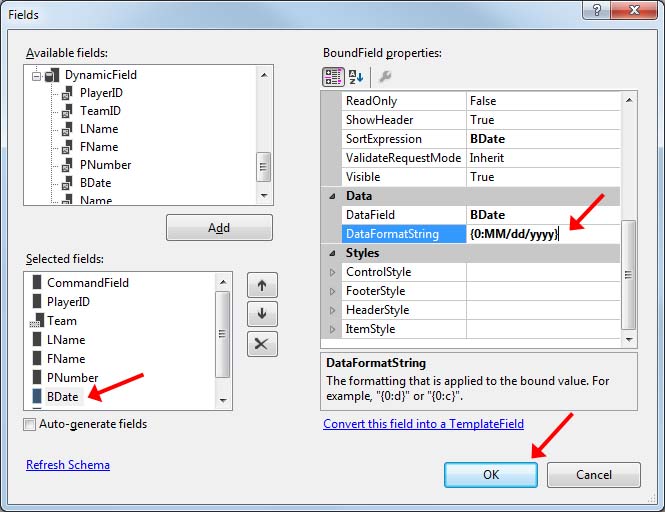
1. Run and test your page (Ctrl+Shift+B, Ctrl+F5). Note: both the *TeamID* and *Name* are displayed. **The Edit, Delete, and Select buttons are gone. We will add them back later.**

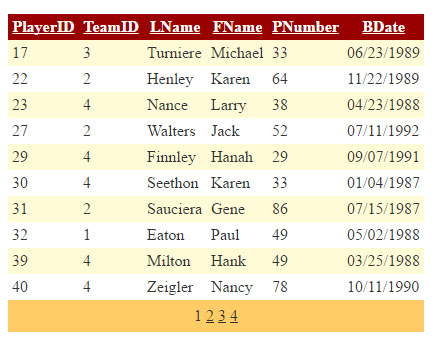
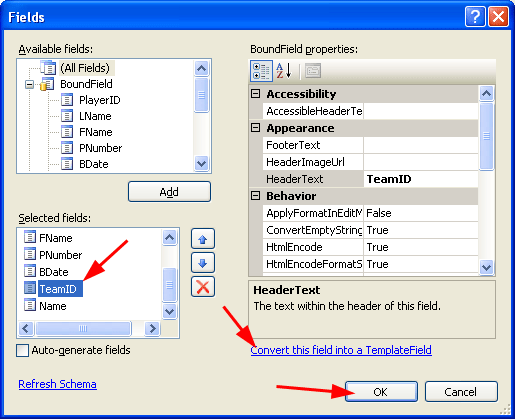
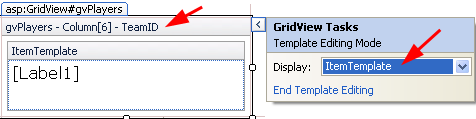
**Stage 6 – Display Team Drop Down in GridView When in Edit Mode**

1. Do the following:
2. Display the “GridView Tasks” menu and choose “Edit Columns”.
3. Select the “Name” field under “Selected fields” (NOT “Available Fields”)
4. Find the *Visible* property under “BoundField properties” (will have to scroll down). Set the value to “False”. Do not choose “OK”. Note:

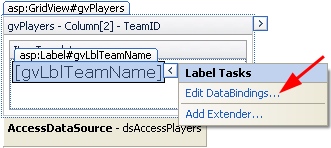
* This is the team Name from the Teams table. We will display it later in place of the TeamID (the foreign key).
* (Read, no action required) We might want to display some other fields from the table (*e.g.* CoachLName). If so, we would want to set “ReadOnly” to *true*. This will prevent it from being treated as a parameter when an Update is done.
* A field whose Visibility is set to *false* is not sent to the browser, nor is it available on the server.

1. While you still are on the previous dialog, select the *PlayerID* field from “Selected fields” and verify that it is read-only. You may need to make it ReadOnly by setting to True.
2. While you still are on the previous dialog, select the *BDate* field and change the *DataFormatString* property to (include the brackets): {0:MM/dd/yyyy}. Then, choose “OK”.

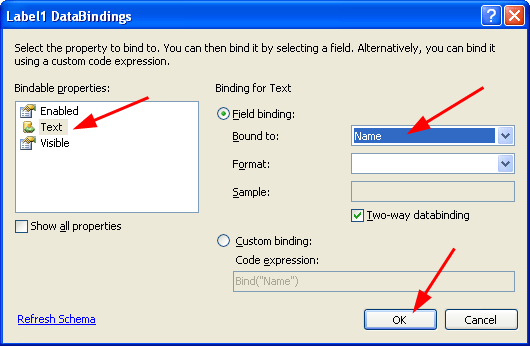


1. Build/Run (Ctrl+Shift+B, Ctrl+F5). Note that BDate is now formatted and Team Name is gone.
2. Select the GridView and :
3. Display the “GridView Tasks” menu
4. Choose “Edit columns...”.,
5. Select “TeamID” from “Selected fields” and convert it to a “TemplateField”
6. Choose “OK”.
7. Select the GridView and :
8. Display the “GridView Tasks” menu
9. Choose “Edit Templates” at the bottom of the GridView Tasks menu.
10. Make sure the “ItemTemplate” is selected for the TeamID.
11. Select the Label inside the ItemTemplate and in the Properties Window, give it the name: *gvLblTeamName*.

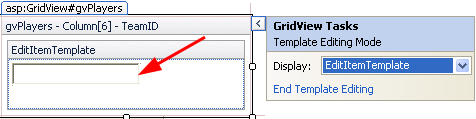
* Note: This is a Label *inside* the Gridview. We will need to access the Text of this Label (Team name), so we give it a meaningful name, one that reflects that it is a Label inside a GridView: *gvLblTeamName*.

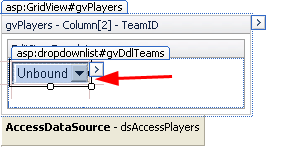
1. Select the Label, display the fly-out menu, and choose, “Edit DataBindings...”.

Note: next, we are going to define what database field we will *bind* to the label’s Text property.

1. Select the “Text” property and bind it to the “Name” field. Then, choose “OK”.

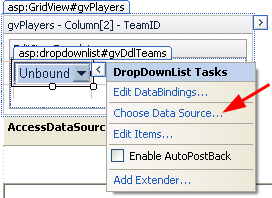
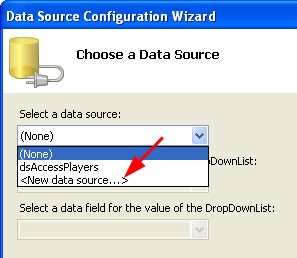
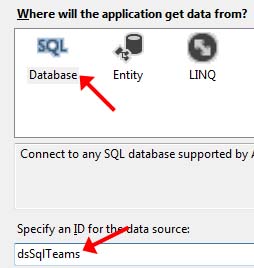
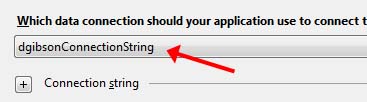
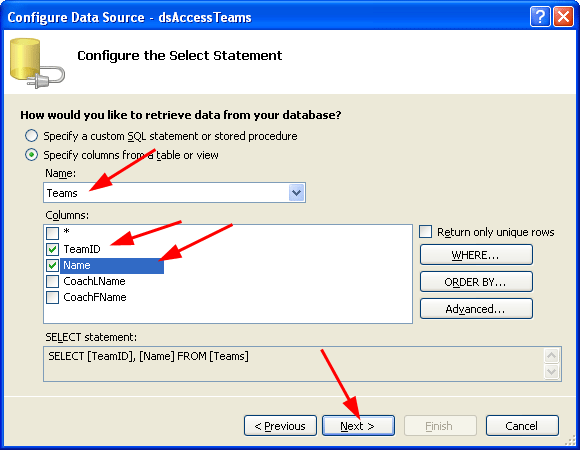
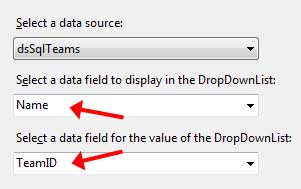
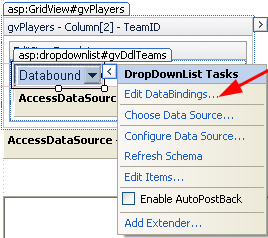
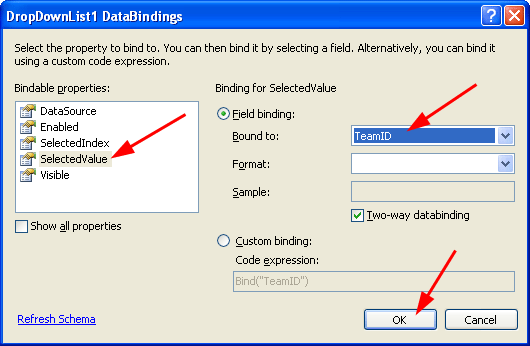
Note: we are now displaying the team Name in place of the corresponding TeamID.

1. Display the “GridView Tasks” menu
2. Select “EditItemTemplate” from the drop down. The result will look as shown on the right. Notice the TextBox.

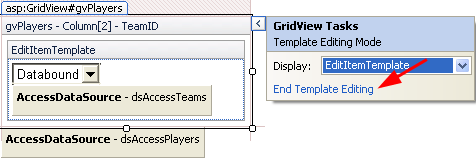
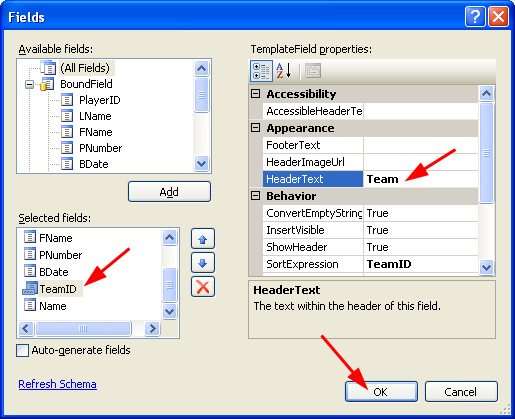
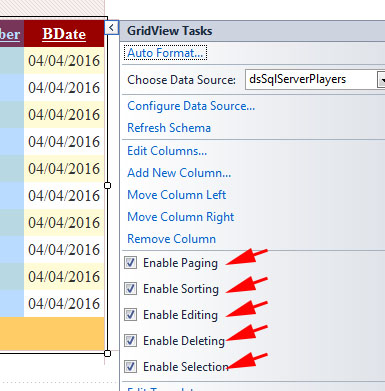


1. Delete the TextBox
2. Go to the ToolBox and add a DropDownList *inside* the Edit Item Template
3. Select the DropDownList and give it the ID: gvDdlTeams.

* Note: Now, when the user chooses to edit a player, a drop down will show in the place of the team name. Next, we need to define a data source for the drop down and specify the fields to use for the Text and Value properties of the drop down.

1. Select the DropDownList and display the fly-out menu.
2. Choose “Choose Data Source...”
3. Select “New data source...>”
4. Select “SQL Database”, give the DataSource the name: *dsTeams* (Not *dsSqlTeams* as shown in the figure on the right) and then choose “OK”.
5. Choose the connection you created earlier, *playersConnectionString* (Not the value shown in the figure on the right) and then “Next.”
6. Select the Teams table and the fields shown below. Then, choose “Next”, Test Query, and then “Finish”.
7. Select the “Name” field for the Text for the DropDownList and the “TeamID” field for the Value. Then choose “OK”.
8. Select the DropDownList, display the fly-out menu, and choose “Edit DataBindings...”
9. Select the values shown on the right and then “OK”.

Note: When the user chooses Edit on a row, the GridView will now display a DropDownList for the selection of the team. This step makes sure the DropDownList displays the team that the player is currently on by binding the SelectedValue to the TeamID, forcing the correct team Name to be displayed.

1. Display the “GridView Tasks” menu and choose “End Template Editing.”
2. Do the following:
3. Display the “GridView Tasks” menu
4. Choose “Edit Columns”.
5. Select the TeamID from the “Selected fields”
6. Change the “Header Text” property to “Team”.
7. Choose, “OK.”
8. Display the GridView Tasks menu and enable the options shown on the right:
9. Open the Source for *Page2.aspx*. Add the highlighted attribute to the GridView tag as shown below

<asp:GridView ID="GridView1" runat="server" AllowPaging="True"

AllowSorting="True" AutoGenerateColumns="False" CellPadding="4"

DataSourceID="dsPlayers" ForeColor="#333333" GridLines="None"

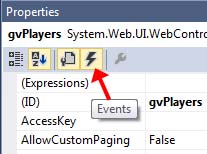
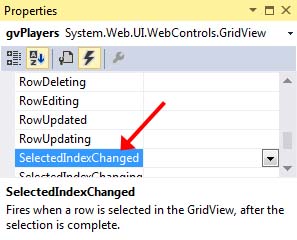
DataKeyNames="PlayerID">

**For some reason, it is removed in one of the steps we did above. This is very important!**

1. Build/Run (Ctrl+Shift+B, Ctrl+F5). Note the following:

* Edit a row. You should be able to change the team through the drop down.
* Delete should work fine

**Stage 7 – Programming the Select Event**

1. (a) Add a multi-line TextBox to your page below the GridView, (b) name it: txtMsg, (c) set TextMode property to Multiline, (d) stretch it larger.
2. Do the following:
3. Select the GridView, go to the Properties Window and display the Events.
4.  Double-click the “SelectedIndexChanged” event and an event handler will be created in the code-behind file.

Note: In a later step, we will write code here to display all the fields in the selected row. One problem is that the TeamID is no longer available. In some situations, when Select is pressed, we might need this value to look up additional information about the team. So, before we write the code, we will do a trick to make the TeamID available to us.

1. Do the following:
2. Open the page in *Source* view.
3. Find the ItemTemplate for the Team TemplateField as shown below
4. Insert (copy/paste below) the HiddenField as shown below, highlighted

<asp:TemplateField HeaderText="Team" SortExpression="TeamID">

<EditItemTemplate>

<asp:DropDownList ID="gvDdlTeams" runat="server" DataSourceID="dsTeams" ...

</asp:DropDownList>

<asp:SqlDataSource ID="dsTeams" runat="server" ConnectionString= ...

</asp:SqlDataSource>

</EditItemTemplate>

<ItemTemplate>

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

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

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

</ItemTemplate>

</asp:TemplateField>

1. Go to the code-behind file and add all the code below to the SelectedIndexChanged event handler.

Note

* As shown with the highlighted lines below, obtaining the value of a Template Field or a Hidden Field is different than how we obtain the value of a Bound Field.

**// Bound Fields**

**int playerID = Convert.ToInt32(gvPlayers.SelectedRow.Cells[1].Text);**

**string lName = gvPlayers.SelectedRow.Cells[3].Text;**

**string fName = gvPlayers.SelectedRow.Cells[4].Text;**

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

**DateTime bDate = Convert.ToDateTime(gvPlayers.SelectedRow.Cells[6].Text);**

**// Template Field**

**Label lblTeamName =**

**(Label)gvPlayers.SelectedRow.FindControl("gvLblTeamName");**

**string teamName = lblTeamName.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}, TeamID={6}",**

**playerID, lName, fName,**

**jerseyNum, bDate.ToShortDateString(),**

**teamName, teamID);**

**txtMsg.Text = output;**

1. Build/Run (Ctrl+Shift+B, Ctrl+F5). Thoroughly test. The Select button should work now. Notice that it displays the Team Name as well as the TeamID (from the hidden field).

**Stage 8 – Package Assignment for Submission**

1. Close VS and zip your *lab10\_lastName* solution folder and submit on Blazeview in the *Lab 10* dropbox.

**You’re done!**