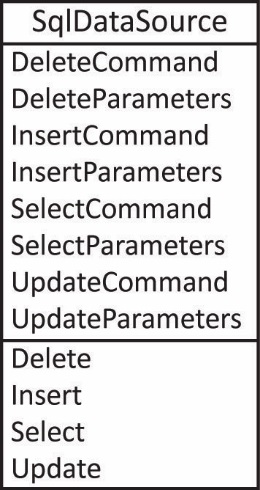
**Data-Binding**

These notes briefly introduce the idea of data-binding and provide an overview of Labs 10-12. In Lab 9, Stage 2, you did an example of binding a *GridView* to a *SQLDataSource*. In the labs that follow, we consider this in much more detail.

**How does Data-Binding Work?**

1. ***Data Source*** – A control that is used to connect to a source of data. The .NET Api provides data source classes[[1]](#footnote-1) to connect to: a database, xml, object model, and others. This type of control is sometimes called a *component.* A component appears on the design surface as you are working on your page in VS, but does not appear when the page is displayed. Another way that you could express what a component (at least in VS sense) is that it is an instance variable for the page class, and it happens to have a visual appearance at design time to make it easier (*e.g.* wizards) to set its properties.

In this section we consider using a SqlDataSource[[2]](#footnote-2) to connect to a database. As shown in the figure on the right, this class has properties that contain the four Sql statements we have considered (DeleteCommand, *etc.*) and methods to execute them (Delete, *etc.*). It also has a ConnectionString property (not shown) to connect to the database.

When you use the wizards to configure a data source, you can choose an option to have the SQL statements automatically built. However, we do have to tinker with them in some situations.

1. ***Data-Bound Web Server Control*** – These are controls that can be bound to a data source control to make it easy to display and modify data in your Web application. All data-bound controls inherit from the abstract class, *DataBoundControl* which exposes (among other members):

* *DataSourceID –* Property used to define a reference to a data source.
* *DataBind* – Method used to connect a control to a data source.

Note: Confusingly, there is also a *DataSource* property. You can only have one of them: *DataSourceID* or *DataSource*, otherwise the page will crash.

1. **Examples of Data-Bound Controls –** We will only consider these two major, data-bound controls: *DropDownList* and *GridView*. We will also consider binding data to a label and a textbox. Here are some others:

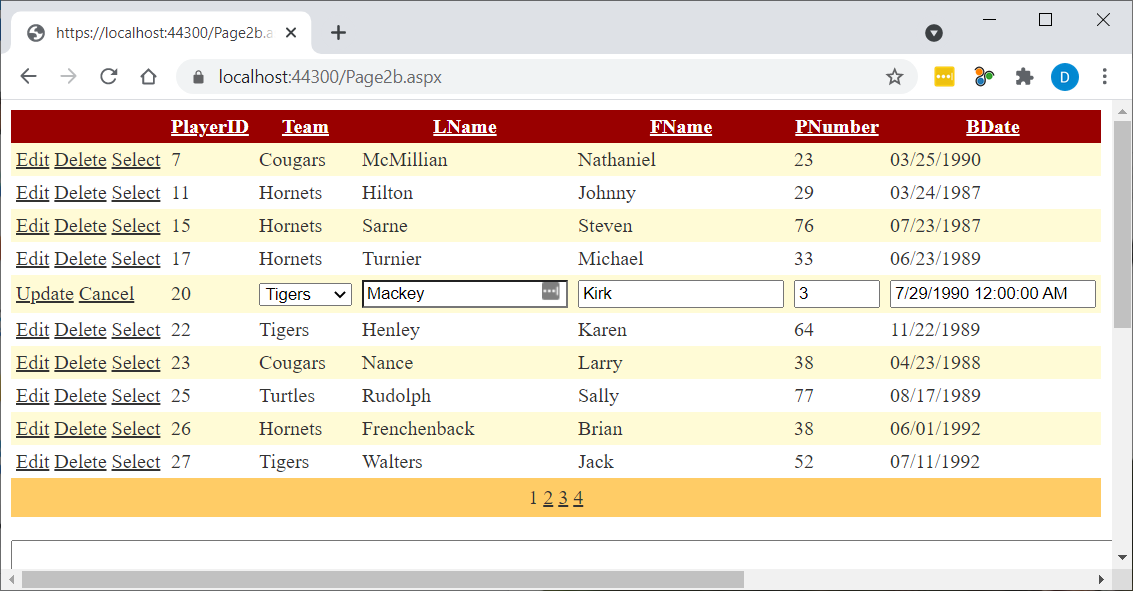
*BulletedList, CheckBoxList, DropDownList, ListBox, RadioButtonList, ListView, DetailsView, FormView, GridView, AdRotator, Chart*

There is a large third-party market that provides many more and enhanced controls. [GrapeCity](https://www.grapecity.com/componentone/aspnet-mvc-ui-controls), [Telerik](https://www.telerik.com/), [List of 30 companies](https://www.infoq.com/research/dotnet-web-components/) (2014, some no longer around)

In the following three labs, we will mostly work with the GridView and the DropDownList.

**Overview of Labs 10, 11, 12**

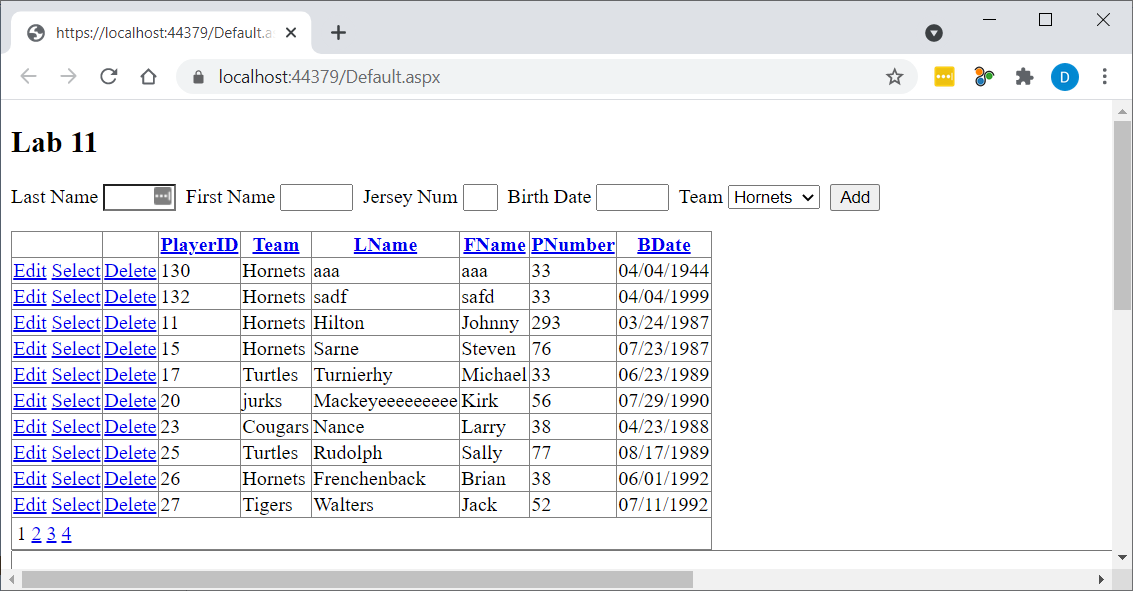
**Lab 10** – We use databinding to display a GridView of players that provides sorting and pagination as well as the ability to edit, delete, and select players. This includes the ability to change a player’s team. Thus, in edit mode, the team name displays a dropdown listing all the teams.



Techniques (page references are for Lab 10):

* Stage 1: Bind GridView(GV) to a DataSource (DS). See p.2-7
  + Select statement: select all fields exceptforeign key. For now, not displaying the foreign key.
  + Generate Insert, Update, and Delete statements
* Stage 2: Inspect the markup.
* Stage 3: Enable Selection, Pagination, Sorting, Editing, Deleting
  + Enable paging, sorting, editing, deleting, selection. All of these features, except selection, are handled *internally* by the GV (but can be intercepted via GV events).
  + Selection triggers the GV’s *SelectedIndexChanged* event. Program this to retrieve the values in the *SelectedRow*. There, you can access the values in the selected row.
* Stage 4: Display Foreign Key, currently can’t change teams nor know team name
  + Started a new project, repeated Stage 1, except select all fields, including foreign key
  + Generate Insert, Update, and Delete statements. Will have to modify in Stage 5
* Stage 5: Display Team Name in GV
  + Re-configured datasource with a custom select sql statement by using the QueryBuilder to build an inner join that retrieves the team name.
* Stage 6: Display Team Drop-Down in GV When in Edit Mode
  + Edit Columns, hide team name
  + Make PlayerID read-only
  + Format the birthdate
  + Make TeamID a TemplateField
  + Edit ItemTemplate and attach databindings to the label to display team name
  + Edit EditItemTemplate, add a drop down, and a DS. Bind SelectedValue to TeamID
  + Change HeaderText to “Team” (was “TeamID”)
  + Enable paging, sorting, editing, deleting, selection
  + Add to GV markup: DataKeyNames=”PlayerID”
* Stage 7: Programming the Select Event
  + Add TeamID as a hidden field in page (might need it later?)
  + Shows how to access value a TemplateField, and a HiddenField in SelectedIndexChanged.

**Lab 11** – We develop an insert feature that updates the gridview. We implement a client-side confirmation of delete dialog. Finally, we put a range validator (not shown) on the PNumber when in edit mode.

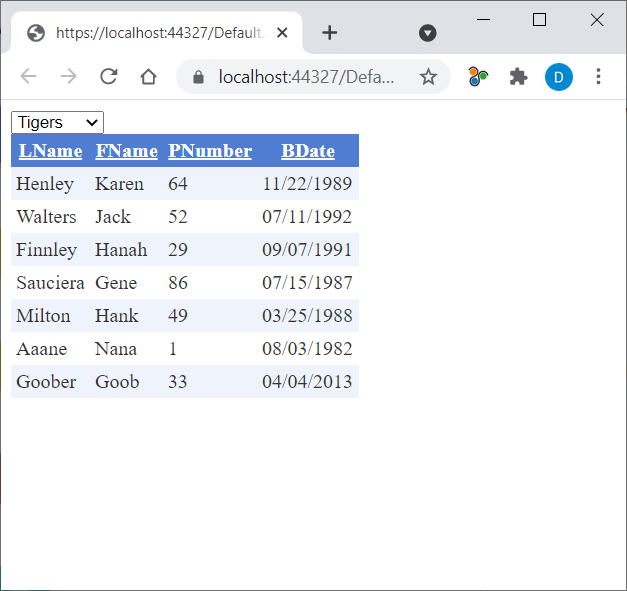


Techniques (page references are for Lab 11):

* Stage 1: Add Insert Feature
  + Add text fields for data entry, and a dropdown
  + Bind dropdown to DS that displays TeamName and TeamID is Value.
  + Add a Add button event handler that: (a) defines the Insert parameters, (b) use the DS to execute the Insert, (c) clears the text fields
  + Modify the DS’s Insert to remove PlayerID (primary key’s are never inserted).
  + Remove the PlayerID parameter.
* Stage 2: Confirm a delete on the client.
  + Add a template field.
  + Edit its ItemTemplate and put a LinkButton there.
  + Set the: (a) CommandName to “Delete”, (b) OnClientClick to name of a JS function, Text to “Delete”
  + Add JS to page.
* Stage 3: Add a Validator:
  + Convert PNumber to TemplateField, on EditItemTemplate, add a validator

**Lab 12**

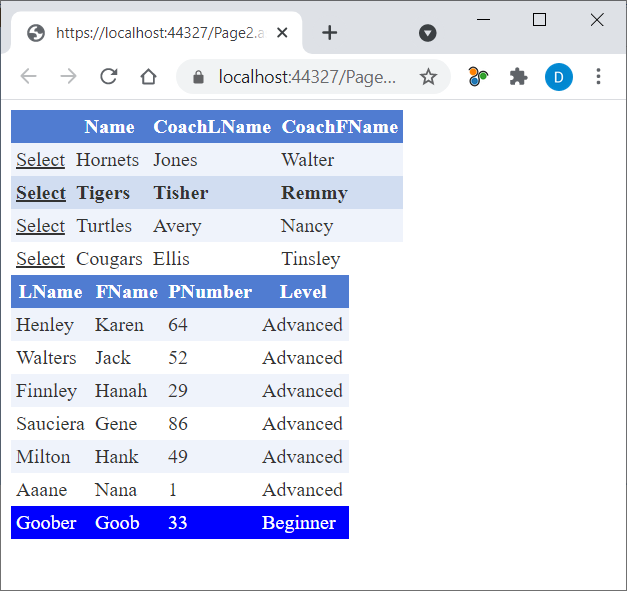
1. Stages 1-2: We illustrate master/detail filtering using a DropDownList and a GridView. When a team is selected from the dropdown, the corresponding players are displayed in the GridView.



Techniques:

* Specify Where clause on GV’s Select SQL statement that binds to drop down’s SelectedValue.

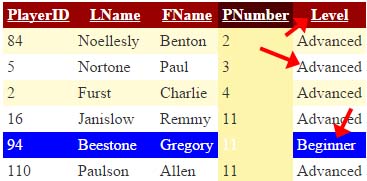
1. Stages 3-4: Next, we illustrate master/detail filtering using two GridViews. When a team is selected from the upper gridview, the corresponding players are displayed in the lower gridview.



Techniques (page references are for Lab 12):

* Specify Where clause on GV’s Select SQL statement that binds to other GV’s TeamID.

1. Stage 5: Add a Calculated Field



Techniques (page references are for Lab 12):

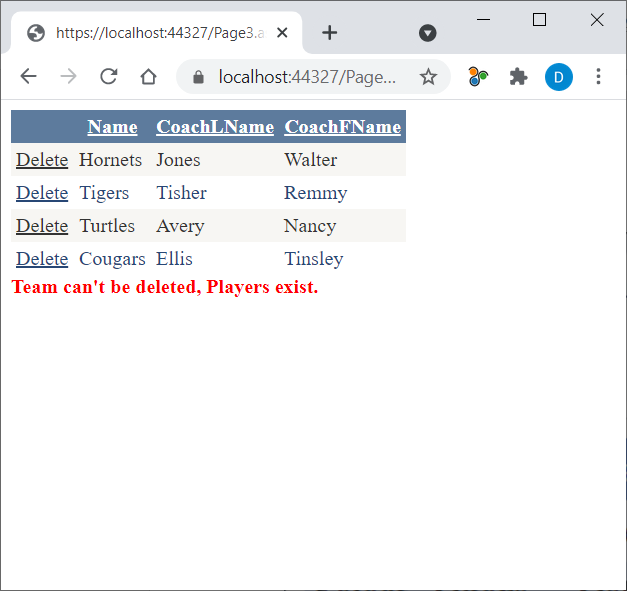
* Convert BDate into TemplateField, set label binding to a code-behind method.

1. Stage 6: Customize the Display: Highlight Beginners as shown above.

Techniques (page references are for Lab 12):

* Intercept each row before it is rendered, change as desired. Program the GV’s RowDataBound event handler.

1. Stage 7: Detecting an Invalid Delete. In this case, we can’t delete a team with players on it.



Techniques (page references are for Lab 12):

* Program the GV’s RowDeleted event handler. See p.8

1. <https://msdn.microsoft.com/en-us/library/ms227679%28v=vs.100%29.aspx> [↑](#footnote-ref-1)
2. <https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource(v=vs.110).aspx> [↑](#footnote-ref-2)