**ADO-Using DataSets and TableAdapters  
*50 Points***

In this exercise we will experience the true power of ADO.Net’s Builder features using Visual Studio. Upon completion of this project, you will likely write no more than 10-15 lines of code. The rest will be manipulated through builders. I assume you have either watched my lecture on-line or in class prior to attempting this relatively easy assignment.

You are now advancing through the Software Development program and need to be able to research. Not all instructions I give will be step-by-step. Some of the material you step through should be intuitive if you take time to read the screen messages.

**Preparation**

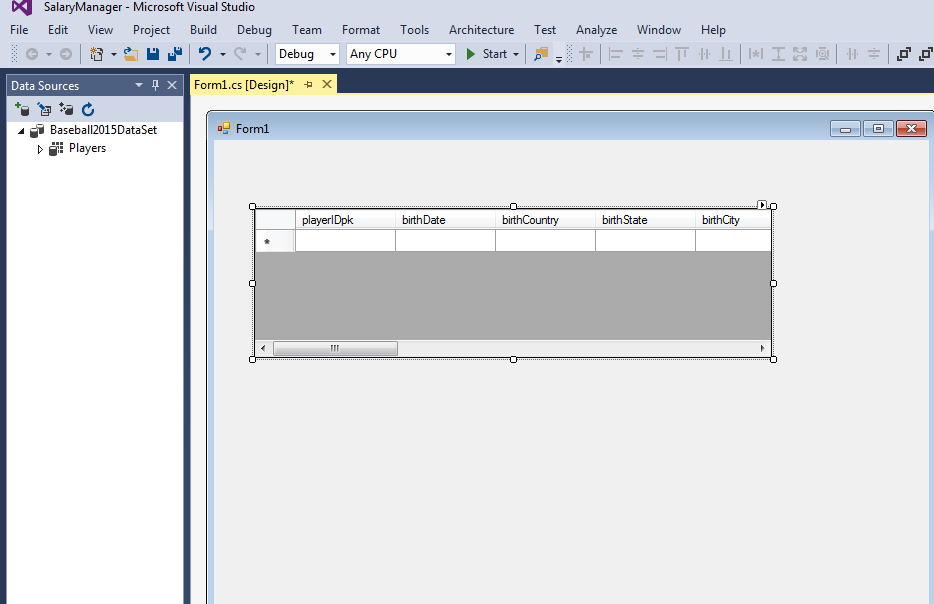
In the SQL Class (CIS126) we had a project in which you were asked to import data into your personal database. You should have the tables Players and Salaries. If you do not have the data in your Database, open SQl Server Management Studio, right-click on your database then select *Tasks/Import Data…* You can then walk through the Import Wizard. Keep in mind the source will be the Baseball201X database.

**Project**

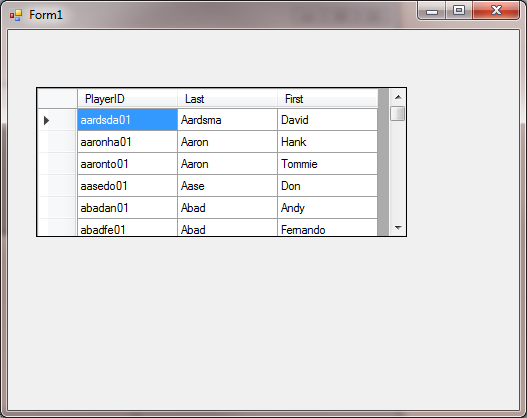
Your project will be simple, create a CRUD screen (*Create, Remove, Update, Delete*). The purpose of this app will not be managing Players but rather those player’s yearly Salaries. Your *DataSet* will include the two tables, *Players* and *Salaries*.

1. Start a new Windows Forms project. Name it ***SalaryManager***.
2. Add a *DataGridView* for Players by dropping it in your form. *DataGridView* Tasks should open by default. If not, click the small black right-arrow that appears on the top right corner of the selected grid. Next select the down arrow to the right of *Choose a Data Source* and select *Add Project Data Source*. Walk through the Wizard which will create the initial *DataSet*. You will be creating a new connection to our SQL Server (134.39.173.35) with your authentication information which should be included in the connection string when asked. You will add only the Player’s table for now. This will generate a Players TableAdapter.

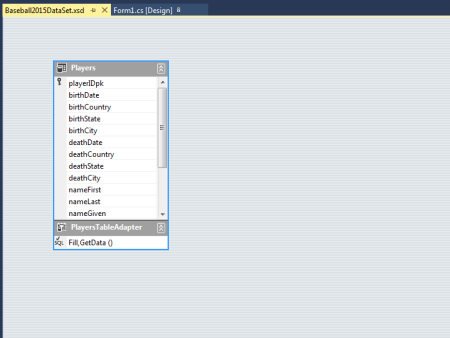
When data is read to your computer from the SQL Server, databases become DataSets and tables become TableAdapters in memory. As we will discover, you can relate TableAdapters just like you relate Tables in a database.



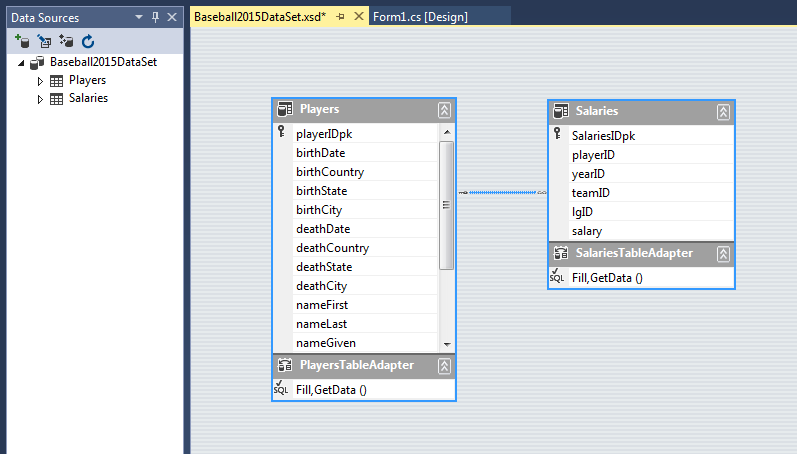
1. Make the Grid read-only (its properties).
2. Format the Grid by clicking the small right-arrow on the top right corner of the grid then *Edit Columns*, so that the grid displays only the player’s ID, Last and First Names (remove unneeded columns). Make sure column headers and widths are appropriate.
3. Run and test that the player’s grid populates. If all works, I suggest you back up the entire solution. Undoing errors in generated code is often problematic.



1. Open the DataSet Designer in your Solution Explorer. The file will end in an *.XSD* extension. You file will likely have your database name as its file name. In my case I pulled the data directly from the BaseBall2015 database.

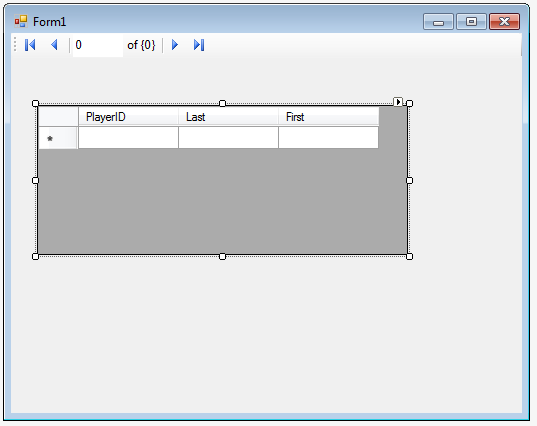


1. The Players *TableAdapter* should be visible. Right-click and add a new *TableAdapter* (Salaries). Walk through and complete the wizard. You should not have to create a new connection. It should default to the connection created for the Players TableAdapter. You will be generating a new SQL Statement. Use the query builder and select the Salaries table and all columns.
2. Once you complete the wizard and are back to the designer, drag the PlayerID column from the Players TableAdapter to the Salaries PlayerID column to establish the relationship you want your app to maintain.

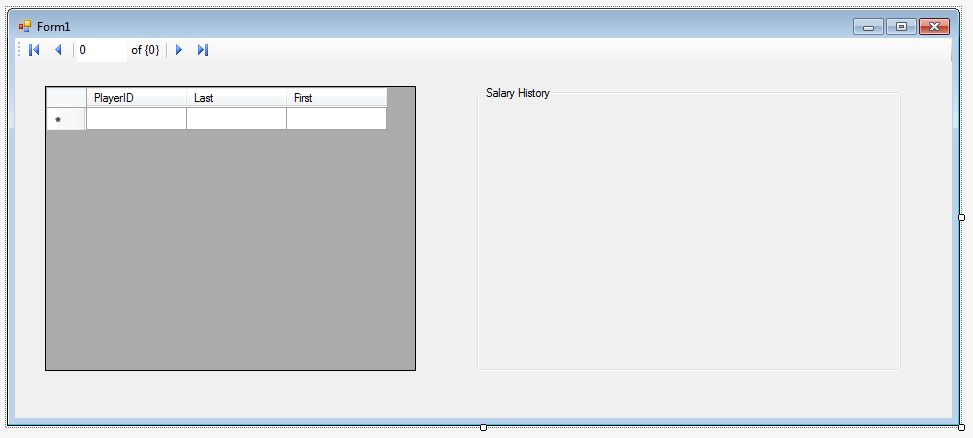


If you do not see the Data Sources dialog as in my example above, open it by selecting: *View/Other Windows/Data Sources*

1. Return to your form. If a BindingNavigator was not automatically created for you (VCRish controls), drag one on the form and assigning it the BindingSource name that was created for the dataset (most likely *<yourDBName>BindingSource* (e.g. Name JSmith\_2017BindingSource). Using the arrow that appears on the BindingNavigator control when selected, remove the Add, Delete and Update options.

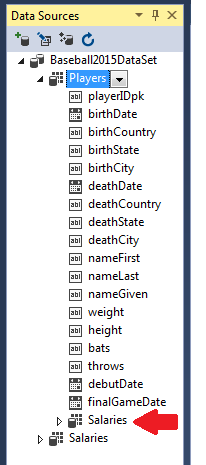


1. Test your work. You should be able to thumb through player names using the BindingNavigator.
2. Widen your form and add a GroupBox to your form large enough the handle a small Salary grid and details to the right of your Players grid. Change the caption of the grid to read “Salary History”.

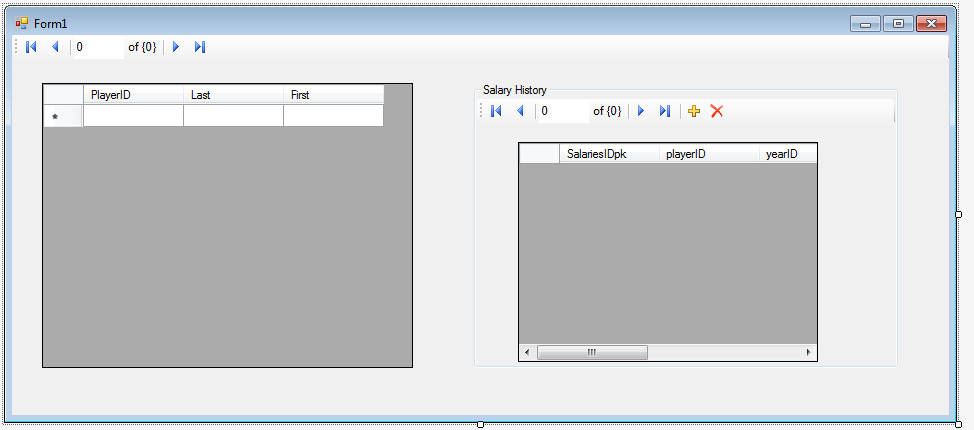


1. From your Data Source Dialog, open players and at the bottom of the list will be the Salaries child table (see warning).

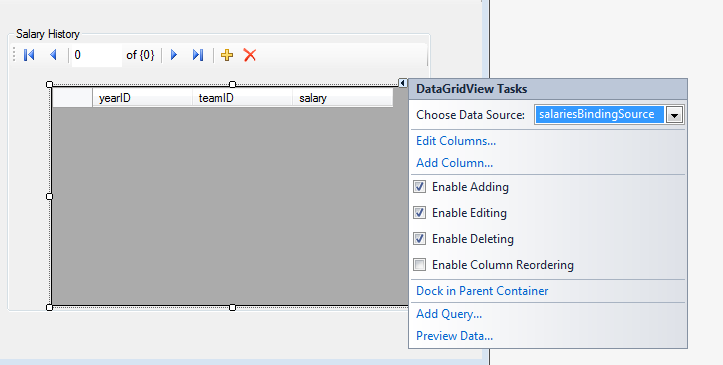
**!WARNING! There will be two instances of the Salaries tablAdapter, one just below Players (do not use this one), and one listed within Players below the field list (use this one).**



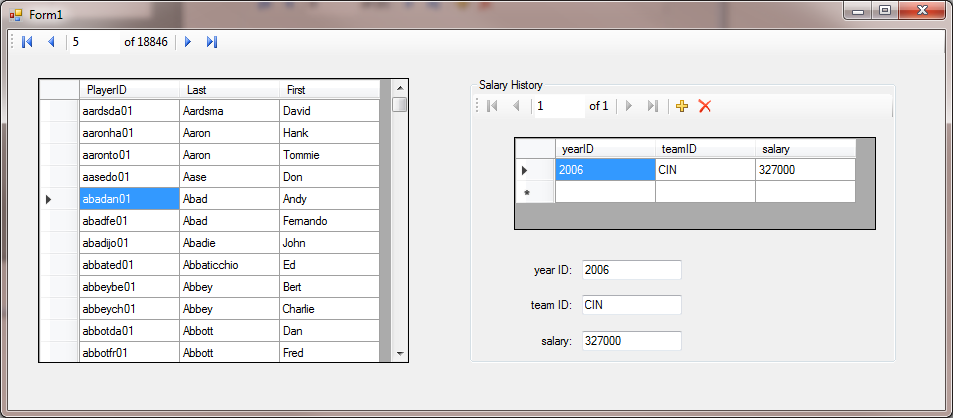
1. Drag Salaries into the GroupBox. If a BindingNavigator is not created for you automatically, add your own as we did above. Make sure to assign the new BindingSource just created (most likely salariesBindingSource) to the BindingSource property of the new BindingNavigator.



1. Edit the grid to include only the Team, Year and Salary. Make sure column widths are appropriate. Set the grid to read-only by removing the edit features from the grid. Deselect “Enable Adding”, “Enable Editing” and “Enable Deleting”. We will let users edit only in detail view.

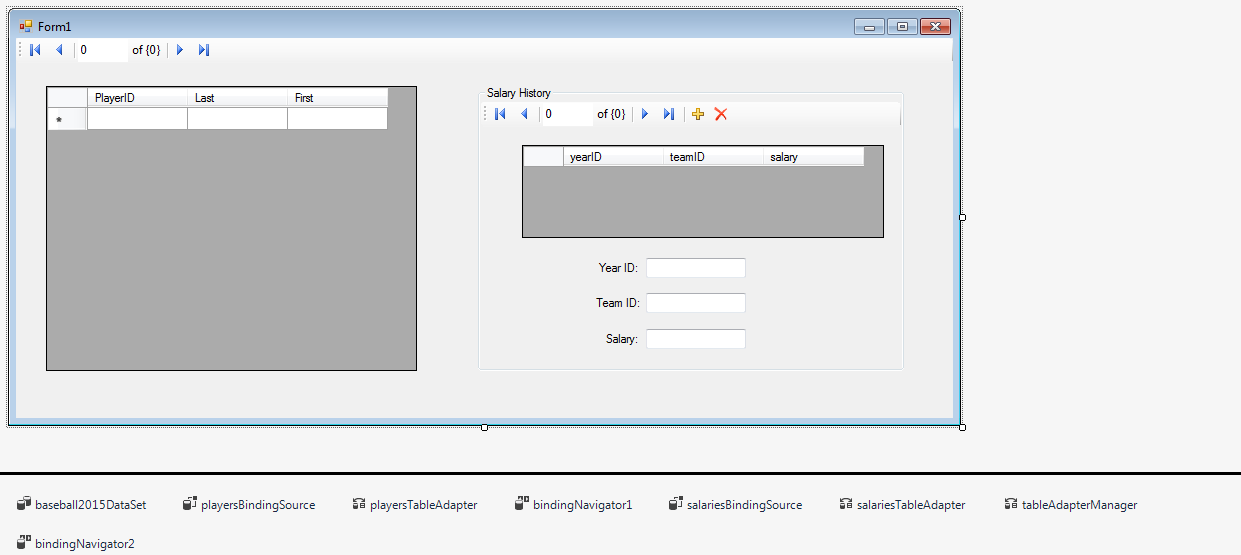


1. Shorten your Salaries grid to provide edit space below. Return to your Data Sources Dialog. A field list appears below the child Salaries table adapter under the Players TableAdapter (see warning). Drag the Team, Year and Salary fields (individually) below the Salary grid still inside the Group box.
2. Test by running your app. Keep in mind many players will not have salary history is they played prior to the 1980s. Make sure that as you select a player, only that player’s salary history appears in the salary grid and details. If not, you likely selected the wrong Salary in the Data Source. It would be easiest to return to the backup I had you create in an earlier step.



Details

1. Spend some time to beautify the application making it look professional. I should not have to scroll horizontally to see data. Add a baseball related graphic to the form. I would suggest you backup your work again.
2. The only code thus far we will need is code to update the main database with changes that could be made to the DataSet. Select the Players BindingSource object below the screen.



Select event handlers in the properties window (Lightning Bolt). Double-click on PositionChanged event handler and add this code:

this.Validate();

this.tblDonationsBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.Players);

This assumes Players is your dataset name. If you are not sure, the name you enter will be the name of your dataset in your Solution Explorer (e.g. Players.xsd).

1. Test your work by adding and deleting Salary information.
2. In my lecture I explained how to add search features. Add the ability to search any last name in the players table. You will be using the Players BindingSource object.
3. Also note that you NEVER leave a default caption of Form1 as a caption. Give your form a meaningful caption.
4. Make sure that the YearID and Salary are numeric using Try/Catches in a leave event handler.
5. Review the instructions to make sure you did not miss a step. Zip your project and submit it to the drop box. When completed, this will be a great demo to place in your jobs portfolio.

**Extra Credit: (notify me you did extra credit on submission)**

+2 Add your own buttons by replacing the BindingNavigator’s default buttons. I would expect you to also include the record counter as in the default BindingNavigator.

+2 Research and apply a feature that makes only the selected row in Players have background of yellow.