Exercise 06 Multi Record Query

Fork the repo called “WebAppNW02-MultiRecord” from “RobbinLawCPSC1517/” to your github account.

Clone the newly forked repo from your account into Visual Studio.

Restore the packages, and then restart the Visual Studio IDE.

Change the name of the solution form “WebAppNW02-MultiRecord” to “WebAppFSIS”.

Run the code to make sure it works before making changes as per Exercise 05. When it works commit and push back to your github repo.

For this exercise replace the Entities that are already in place as described below. Remember to look at the actual database data types and reflect them properly in the Entity definitions. Also remember that when you define something with say “int?”, this means that the database field in the actual database is NULLABLE. Strings don’t need this and of course the primary key as it will never be nullable.

In the Entities01.cs class file change the Categories definition of the Northwind database to the Teams definition of the FSIS database.

In the Entities02.cs class file change the Products definition of the Northwind database to the Players definition of the FSIS database.

In the Entities03.cs class file change the Suppliers definition of the Northwind database to the Guardians definition of the FSIS database.

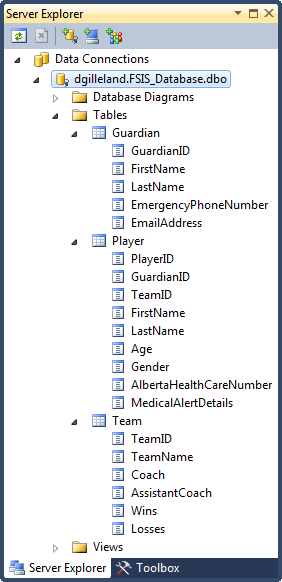
For this exercise change the name of the database from “NWDB” to “FSIS\_db” in the Context.cs file.

For this exercise change Controller02.cs to reflect the changes from the Northwind database to the FSIS database. This means calling the right Stored Procedure.

Exercise 05 Client Server Setup PART 2

ENTITIES Layer

Create the following Entity classes for the tables from the database. Be sure to

* select the proper data type for all properties, based upon the SQL data type of the corresponding table column;
* no constructors required

You must create Entity classes for the following. Ensure you use appropriate Entity Framework notation. Add the System.ComponentModel.DataAnnotations to the application library project.

* **Guardian**
* **Player**
* **Team**

### Checking Guide

|  |  |
| --- | --- |
| **✓** | **Item** |
|  | Each class has the correct property names and data types. |
|  | Necessary notation is in place. |
|  | Nullable types (if any) are correctly applied to the appropriate  properties |
|  | Selected property created in the requested implementation. |
|  |  |

EntityFramework Package Download, DAL Layer, and web.config

1. Using Manage NuGet Packages, add *EntityFramework* to your FSISSystem and web application projects. Add a reference to System.Data.Entity to both the projects.
2. Create your Data Access Layer class FSISContext which inherits DbContext. Create the default constructor which will assign the base value of “FSIS\_db” for DbContext class. Add a property using the datatype DbSet<T> for each of your entities. Access type for this class will be “internal”.
3. Configure your solution’s knowledge of the database by making the following changes to your system.
   * Edit the web application’s *web.config* file to have the following for the <connectionStrings> tag  
     <connectionStrings configSource="WebConnectionStrings.config"/>
   * Add a new configuration file to the root of the web application and name it “*WebConnectionStrings.config*”; on the inside, put the following code  
     <connectionStrings>  
      <add name="FSIS\_db"  
      connectionString="Data Source=.;Initial Catalog=FSIS\_Database;Integrated Security=True;"  
      providerName="System.Data.SqlClient"/>  
     </connectionStrings>
   * Edit the web application’s *web.config* file to have the following tag <contexts> added to the <entityFramework> tag.  
     <contexts>

<context disableDatabaseInitialization="true"  
 type="FSISSystem.DAL.FSISContext,FSISSystem">

</context>  
</contexts>

### Checking Guide

|  |  |
| --- | --- |
| **✓** | **Item** |
|  | The provided **files** and package **libraries** are correctly added to the project |
|  | References have been properly set up. |
|  | Context class has been properly set up. |
|  | Correctly modified *web.config* file and correctly added/edited the *WebConnectionStrings.config* file. |
|  | Correctly modified *web.config* file and correctly added the entityframework *<contexts>* tag. |

Exercise 05 Client Server Setup PART 3

Add a Class to your BLL folder and call it TeamController.cs

To this file add two using directives to the DAL and ENTITIES folders.

To this file add a public class called TeamController, and inside the class add a method called Teams\_FindByID(int teamid).

Add the following code as the guts of the method:

using (var context = new FSISContext())

{

return context.Teams.Find(teamid);

}

In the ExcercisePages/SimpleQuery Form add the tags necessary to present labels, a textbox, and a button, for a simple PKey Query, similar to the Northwind Demo, but for the Team Table of the FSIS Database instead of the Region Table of the Northwind Database.

In the SimpleQuery code behind add the code necessary to make everything work when the button is pressed. This code will be similar to the code behind in the Northwind Demo. Remember to put in the proper using directives, so your code can access the BLL, and ENTITIES code.

Test your code and when it works, save it to GitHub.

At this point you can demonstrate all of Exercise 05 to your instructor for evaluation.

FSIS Client-Server Query – MultiRecord Query using Code Behind

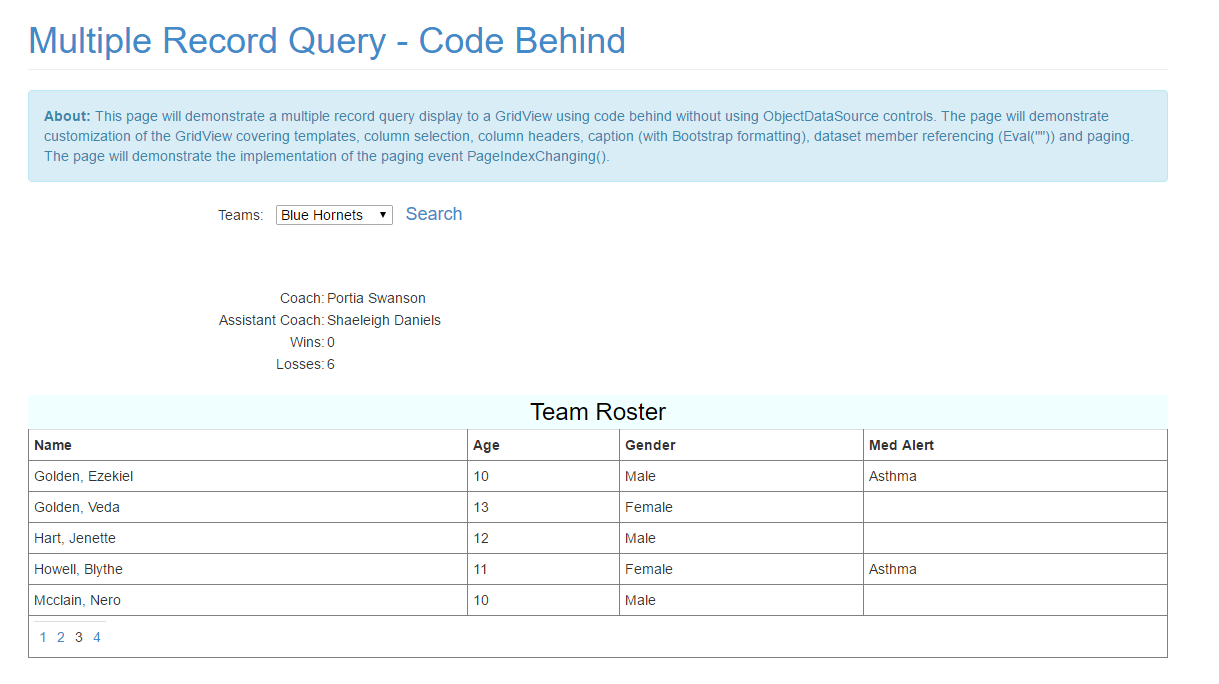
Displaying Multiple Rows

Modify the Visual Studio solution from your previous exercise, to add a form to display multiple rows of data using a GridView control. Base it upon the screenshot below. Perform the lookup of data for the GridView by the DropDownList in the code-behind of the form. All code to load the DropDownList and GridView is to be done in the code behind, **NOT** using an ObjectDataSource. Include a prompt line for the DropDownList.

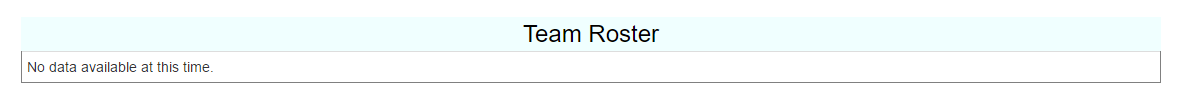
This exercise will require you to make 2 separate queries to the database under the one button event. One query will retrieve the Team data record, and then display the data in individual fields. The second query will retrieve all the team player records (List<>), and then assign the collection to a GridView. You will need to customize the GridView for column headers, columns displayed, caption and paging (5 items/page). Use templates for customizing your GridView columns.

Create the following:

* BLL
  + Create the PlayerController, TeamController and GuardianController BLL Classes
  + Create the following method in the appropriate TeamController BLL Classes
    - Team\_List() which will return a List<T> containing all the Team records
    - Team\_Find(int teamid) which will return a single Team record matching the input parameter
  + Create the following methods for the PlayerController
    - Player\_GetByTeam(int teamid) which will return a List<T> containing all the Player records for that team



Team with no players should have a EmptyTemplate message shown.



### Checking Guide

|  |  |
| --- | --- |
| **✓** | **Item** |
|  | Creation of a Team, Guardian and Player BLL controller(s) |
|  | BLL method to perform lookup all Team data correctly coded (TeamController) |
|  | BLL method to perform lookup Team by Team ID correctly coded (TeamController) |
|  | BLL method to perform lookup of Players by Team ID correctly coded (PlayerController) |
|  | GridView’s EmptyDataTemplate has appropriate text to indicate if no information if found |
|  | Form’s DropDownList is correctly populated using code behind |
|  | Selected Team data has been show in non-editable web controls (ie Label) |
|  | Null values of Wins or Losses are displayed as 0 (zero) |
|  | Form’s GridView is correctly populated in event code in the Code-File |
|  | GridView uses Templates (ie Label controls) |
|  | Gender is formatted from single character to Male and Female |
|  | Paging of Gridview properly implemented using call to the database |
|  | Appropriate error handling (Try/Catch is present where required) |
|  | Appropriate not found or no selection message is displayed |
|  | SiteMapNode has been added to your project to reach this page. |