# Entity Framework

Entity Framework (EF) is used to access our SQLLite database within the Player and Games Controllers.

EF manages the DB and creates migration files. This will not be needed to complete the activity.

You’ll be working with DBSet object’s - [Reference](https://docs.microsoft.com/en-us/ef/core/api/microsoft.entityframeworkcore.dbset-1#Microsoft_EntityFrameworkCore_DbSet_1)

*<Set>.*[Add](https://docs.microsoft.com/en-us/ef/core/api/microsoft.entityframeworkcore.dbset-1#Microsoft_EntityFrameworkCore_DbSet_1_Add__0_)*( <object> )*

Adds the object to the target table.

*<Set>.*[Update](https://docs.microsoft.com/en-us/ef/core/api/microsoft.entityframeworkcore.dbset-1#Microsoft_EntityFrameworkCore_DbSet_1_Update__0_)*( <object> )*

Updates the object within the target table.

The object will need its [Key] value filled for the look up of the data to be updated. i.e. To update a player make sure the player.Id value is a value within the database.

# LINQ - .NET Language-Integrated Query

Linq is the way that we query sets of data. Allows us to sort a list, or add an element to that list, pull a single element from the list, etc.

Information: <https://msdn.microsoft.com/en-us/library/bb308959.aspx>

Reference: <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/query-keywords>

Below is a list of provided function that will be helpful in completing the Ping Pong activity.

*<Set>.*[*FirstOrDefault*](https://msdn.microsoft.com/en-us/library/bb340482(v=vs.90).aspx)*( set => set.<element> = <value> )*

Returns the First (or default, usually null) object found where that object’s element equals the inputted <value>.

*<Set>.*[*OrderBy*](https://msdn.microsoft.com/en-us/library/bb534966(v=vs.110).aspx)*( set => set.<element> )*

Returns the set sorted by the element specified in acceding order. (A -> Z, 0 -> 10)

*<Set>.*[*OrderByDescending*](https://msdn.microsoft.com/en-us/library/bb534855(v=vs.110).aspx)*( set => set.<element> )*

Returns the set sorted by the element specified in descending order. (Z –> A, 10 –> 0)

*<Set>.*[*Take*](https://msdn.microsoft.com/en-us/library/bb503062(v=vs.110).aspx)*( <value> )*

Returns the top <value> (i.e. 10) results.

Note: Within the ( set => set.element ) the “set” is just a temporary name for the data set you are accessing.

# Javascript/JQuery

*$.*[*ajax*](http://api.jquery.com/jquery.ajax/)*(xhr)*

Sends an XML Http Request. xhr structure can be found within the code.

*document.*[getElementById](https://www.w3schools.com/jsref/met_document_getelementbyid.asp) *( “<id>” )*

returns the element with inputted id. Gives us a way to save elements locally.

*<element>.*[appendChild](https://www.w3schools.com/jsref/met_node_appendchild.asp)*(<child>)*

Adds a child element to another element.

*$(‘#id’).*[*click*](https://www.w3schools.com/jquery/event_click.asp)*( <function> )*

Runs the inputted function when an element with id=’id’ is clicked.

*$(‘#id’).*[ready](https://www.w3schools.com/jquery/event_ready.asp)*( <function> )*

Runs the inputted function when an element with id=’id’ has been loaded.

$('#id').[empty](https://www.w3schools.com/jquery/html_empty.asp)()

Removes all child elements of element with id=’id’

$('#id').[append](https://www.w3schools.com/jquery/html_append.asp)()

Appends html to the end of element with id=’id’

*console.log(<string>)*

Outputs a string to the console of a browser, useful for viewing variables at runtime.