Dotnet Week 09 Day 01 Intro to ORMs

Object Relational Mappers-Allow us to create and communicate with web server in C#. We can query our database using ORMs.

In dotnet we use Entity Framework (EF)- the open source ORM that's built by Microsoft to go with C#. Most popular. Its what takes your LINQ queries and converts them to sql and then takes your sql results and converts it back to C#.

** dotnet tool install --global dotnet-aspnet-codegenerator   //Only once. It will then be global to your machine.

```
dotnet new console -n IntroToOrms
```

**Always add gitignore to your project.

**ADD these to every project that will use Entity Framework //Throw them into your terminal each time you start project

```
dotnet add package Microsoft.EntityFrameworkCore.Design    //Here are the core libraries.

dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL  //Postgres EF driver.
```

**You can check that these have been added to your .csproj .

Create you database(db):

```
createdb SafariVacation
```

Create a link from C# to my new db. This is your db context:

```
dotnet ef dbcontext scaffold "server=localhost;database=SafariVacation"
Npgsql.EntityFrameworkCore.PostgreSQL -c SafariVacationContext
//Run this command in your terminal.
```

Now you can check what is in the SafariVacationContext.cs you just created. // Ignore and delete the warning for now. It has now created a new class that inherits properties from db context.

Start creating your db tables: Create a Models folder and place a SeenAnimals.cs class in it. In the file: give it properties(create a table):

```
 namespace IntroToOrms
{
  public class SeenAnimals
{
  public int Id { get; set; }
  public string Species { get; set; }
  public int CountOfTimesSeen { get; set; }
 public string LocationOfLastSeen { get; set; }
}
}
```

Dotnet Week 09 Day 01 Intro to ORMs (continued)

Next add that table to your db: To your SafariVacationContext.cs:

```
        □□public DbSet<SeenAnimals> SeenAnimals { get; set; }
        }
}
```

//Add this property to the very bottom of your SafariVacationContext.cs

Now you need to make this exist in your db:
// Add this to your terminal and run

```
        dotnet ef migrations add SeenAnimals
```

// This will update your db to reflect the update
// This will create your migration files
□// To undo this action, use 'ef migrations remove'

Now you need run your migartion:
// Add to terminal

```
        dotnet ef database update
```

Test to see if you can see your tables in the terminal :

```
        pgcli SeenAnimals

        \dt “SeenAnimals”
        and/or

        \d “SeenAnimals” □□\d "SeenAnimals"
```

**Add data to your db is next: YOU MUST be using System.Linq;   AT THE TOP OF program.cs BEFORE ADDING ANY DATA□

Dotnet Week 09 Day 01 Intro to Entity Framework and LINQ

Four main database operations. **Create, read, update, delete data.**

Create data: //Add this to your program.cs // This is the hardcoding method of creating data

```
        var db = new SafariVacationContext();

        db.SeenAnimals.Add(new SeenAnimals {
                Species = "Lions",
                CountOfTimesSeen = 10,
                LocationOfLastSeen = "Dessert"
    });

         db.SaveChanges();
```

**Test it: dotnet run,   pgcli SeenAnimals,   SELECT * FROM SeenAnimals // Addition shows in table

Dotnet Week 09 Day 01 Intro to Entity Framework and LINQ (continued)

// OR you could ask the user to input (create data)

Get info from console and add to DB:

```
        System.Console.WriteLine("Enter the seenanimals details");
        var species = Console.ReadLine();
        var countoftimesseen = int.Parse(Console.ReadLine());
        var locationoflastseen = Console.ReadLine();

        db.SeenAnimals.Add(new SeenAnimals {
                Species = species,
                CountOfTimesSeen = countOfTimesSeen.
                LocationOfLastSeen = locationOfLastSeen
        });

        db.SaveChanges();
```

**Now you can test it: dotnet run and it will cue you for responses. // In terminal:

```
        Enter the seenanimals  details
                Tigers
                20
                Jungle
```

**Test it: pgcli SeenAnimals, SELECT * FROM SeenAnimals // Addition shows in table

Read data: //Add this to your program.cs **Display all animals the user has seen**

```
        var allSeenAnimals = db.SeenAnimals;

        foreach (var seenanimal in allSeenAnimals)
        {
        System.Console.WriteLine(seenanimal.Species);
    }
```

// dotnet run to test if they show in terminal

Now try to **Display only animals seen in the jungle

```
var onlySeenInTheJungle = db.SeenAnimals.Where(seenanimal => seenanimal.LocationOfLastSeen
== "Jungle");

System.Console.WriteLine("Animals seen in the Jungle");

foreach (var seenanimal in onlySeenInTheJungle)
    {
      System.Console.WriteLine(seenanimal.Species);
    }
    // dotnet to test if they show up in the terminal
```

Dotnet Week 09 Day 01 Intro to Entity Framework and LINQ (continued)

Update data: // Add this to your program.cs to update any of the properties for a specific id

// Find it

```
  var Lions = db.SeenAnimals.FirstOrDefault(seenanimal => seenanimal.Id == 1);
      if (Lions != null)
```

//Update it

```
      {
        Lions.LocationOfLastSeen = "Jungle";
        Lions.CountOfTimesSeen = 30;
```

//Save it

```
        db.SaveChanges();
     } else {
             Console.WriteLine("SeenAnimal with ID 1 not found");
}
```

**To test if they show up in db through the program without using pgcli in the terminal

```
var allanimals = db.SeenAnimals;
      foreach (var animals in allanimals)
      {
        Console.WriteLine($"{animals.Id}, {animals.Species}");
      }
```

Delete data:

// Find it

```
var seenAnimalsToDelete = db.SeenAnimals.FirstOrDefault(seenanimals => seenanimals.Id == 1);
```

//Delete it

```
If (seenAnimalsToDelete != null)
{
db.SeenAnimals.Remove(seenAnimalsToDelete);
```

// Save it

```
db.SaveChanges();
}
```

Dotnet Week 09 Day 01 Intro to Entity Framework and LINQ (continued)

Adding a column to a table:

Go to SeenAnimals.cs and add at the bottom of the properties:

```
namespace IntroToOrms
{
  public class SeenAnimals
{
 public int Id { get; set; }
 public string Species { get; set; }
 public int CountOfTimesSeen { get; set; }
 public string LocationOfLastSeen { get; set; }

//Add it here

public bool DoesItSwim { get: set: }
}
}
```

// This will throw an error when you dotnet run it
// We need to update migration: //Run in terminal:

```
dotnet ef migrations add DoesItSwim
```

// This will show new migrations
// You can add as many as needed

Then update your db:

```
dotnet ef database update
```

//You will get a response from terminal (Applying migration ‘20181226175200_DoesItSwim’.) and you will see the new columns in db.

Dotnet Week 09 Day 01 How to Get started with EF

Create a new project: . // Run all these commands in your terminal

```
dotnet new console -n NewProjectExample

        // in general for all projects in C# :
        // dotnet build
        // dotnet run
        // git init
        // hub create

cd NewProjectExample

dotnet add package Microsoft.EntityFrameworkCore.Design

dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL

createdb SafariVacation

dotnet ef dbcontext scaffold "server=localhost;database=NewProjectExample"
Npgsql.EntityFrameworkCore.PostgreSQL -c NewProjectExampleContext

code .
```

// Now it has given us the NewProjectExampleContext.cs file

Create a Models folder and add a file called ExampleClass.cs. In the file: give it properties(create a table):
    // In your ExampleClass.cs file add the following:

```
namespace NewProjectExample
        {
          public class ExampleClass
        {
                public int Id { get; set; }
                public string NameOfClass { get; set; }
                public int NumberOfClasses { get; set; }
                public string LocationOfClasses { get; set; }
  }
}
```

Next add that table to your db: To your **NewProjectExampleContext.cs**:
    // At the bottom add this. This line below is what will allow us to actually query the database

```
        // public DbSet<ModelClassName> TableName{ get; set; }
        public DbSet<ExampleClass> ExamplesClass { get; set; }
        }
}
```

Dotnet Week 09 Day 01 How to Get started with EF (continued)

Next we go to the program.cs file: //Should look like this. Add the following to the file:

```
using System;
using System.Linq;

namespace SampleDatabase
{
        static void Main(strings[] args)
        {
                Console.WriteLine("Hello World");
                var db = new NewProjectExampleContext();

                db.ExampleClass.Add(new ExampleClass{
                        NameOfClass ="Class A",
                        NumberOfClasses = 100,
                        LocationOfClasses = "Location A"
                });

                var classNames = db.ExampleClass.Where(exampleclass =>
                exampleclass.NumberOfClasses > 100).Select(s => s.NameOfClass);

                classNames.ToList().ForEach(Console.WriteLine);
                }
        }
}
```

Then from the terminal run these commands:

```
        dotnet ef migrations add AddedFirstTable
        // This will add migration files to the Migrations folder

        dotnet ef database update
        // This will apply all the migrations you have built to your database
```