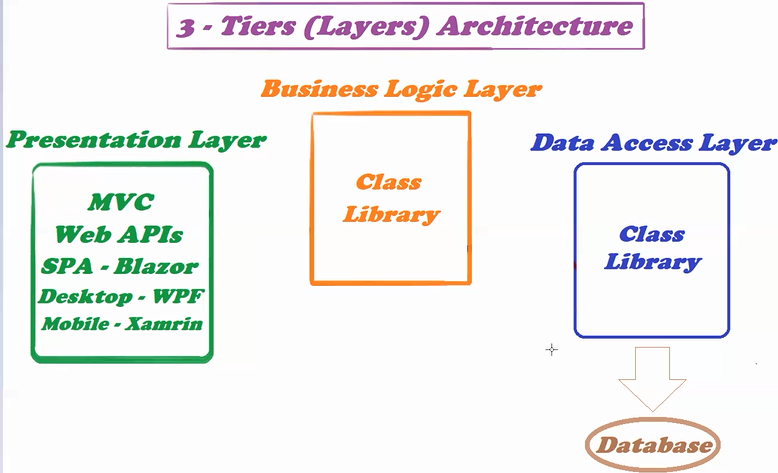
**Architecture Pattern:**

Before starting any project decide the architecture pattern you will use to build that project

We have 2 categories for projects classification:

* monolithic architecture: the project is divided modules or services in the same project /or solution /or system 🡺 what we’ll cover on MVC course
* micro services: each group of services in separate project [each project may be implemented with different technology]

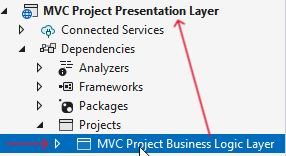
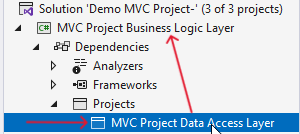
**3 Layers Architecture Pattern:**



Add reference from data access layer to the business access layer and add reference from the business access layer to the presentation access layer

Each installed package in one layer is seen in the other layers

Add reference then build the solution so that the changes in packages takes place

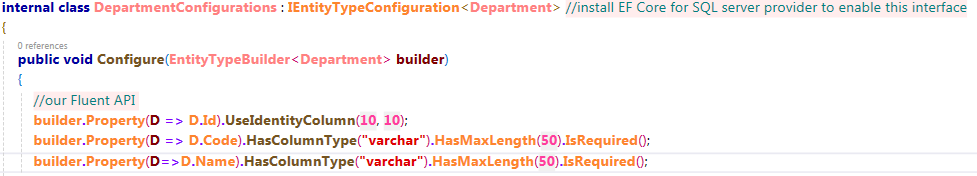
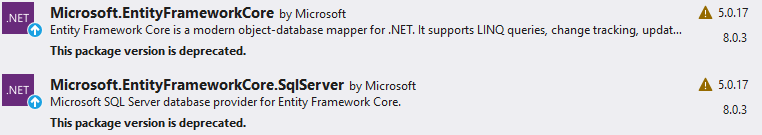
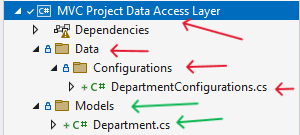
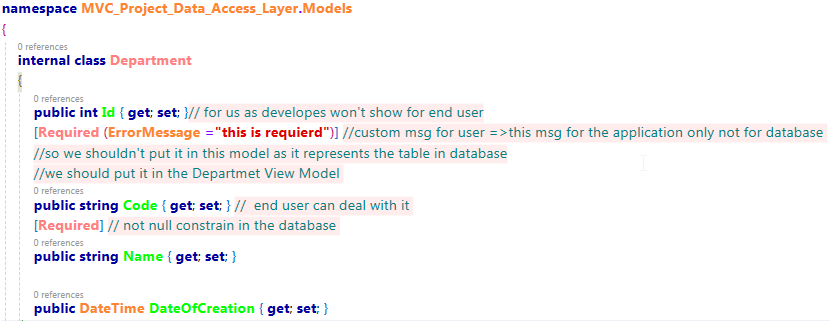


We are going to divide our project as modules 🡺 start with Department module

1. **Data Access Layer:**

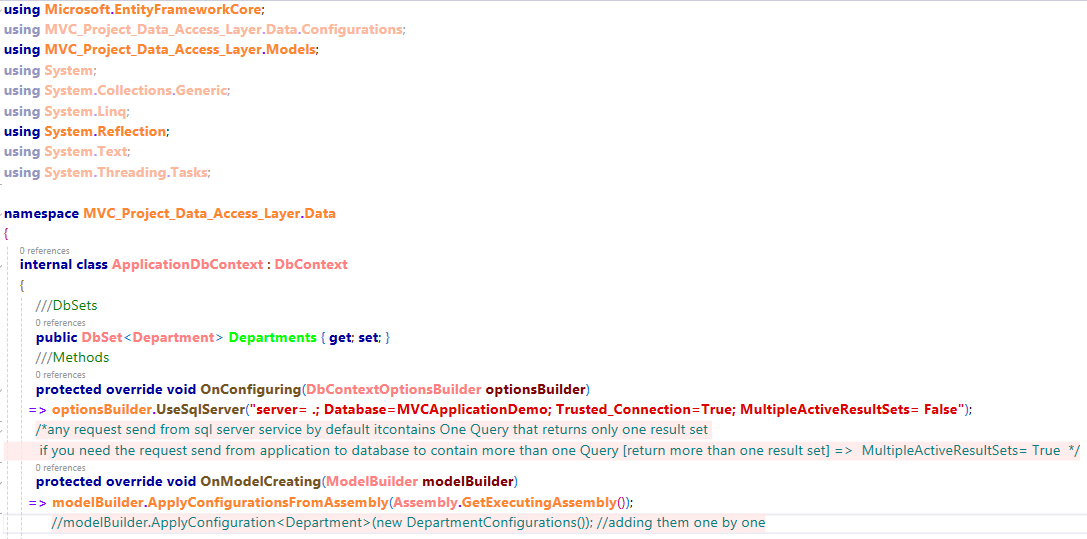
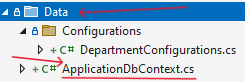
Has 2 folders

* Data :contains DbContext class and configuration classes and migrations
* Models/domains/entities: contains the domain models
* Create model Department

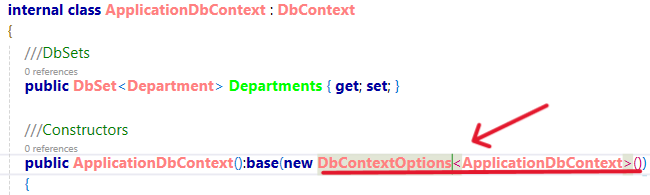


Applying the configurations in the OnModelCreating in the DbContext class

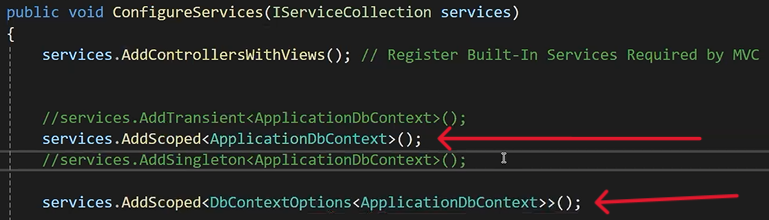
* Creating the DbContext class in the Data folder directly



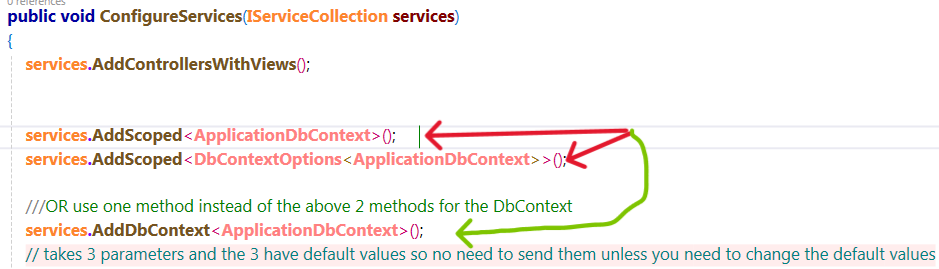
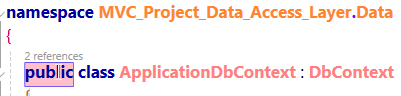
**DbContext – Dependency Injection:**

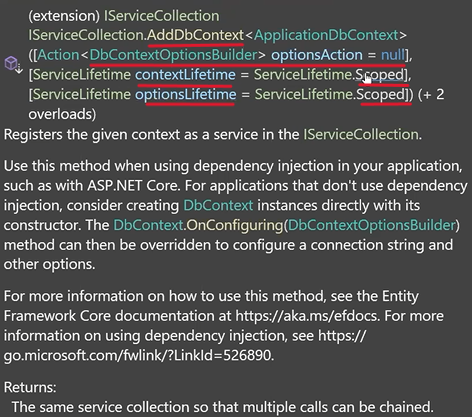


Allow DI for ApplicationDbContext 🡺 in the presentation layer 🡺 Startup class 🡺

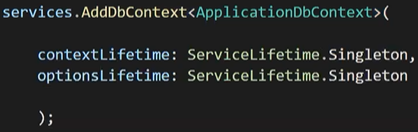


Remember to make class ApplicationDbContext public so that we can use it in the dependency injection in the Startup class

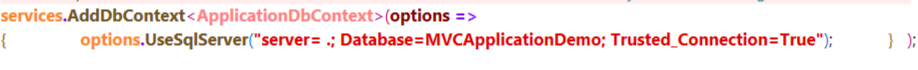
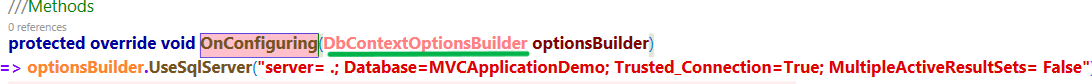




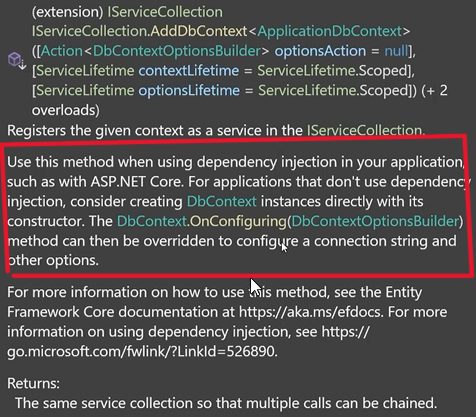
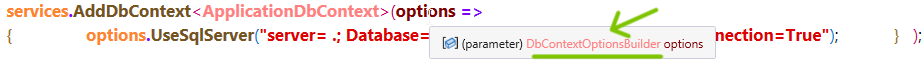
* To change the default values of the parameter send them by name 🡺



* The 1st parameter action void method DbContextOptionsBuilder which is in the ApplicationDbContext class in the OnConfiguring method 🡺 which means we can send the connection string in this method instead of overriding the OnConfiguring method

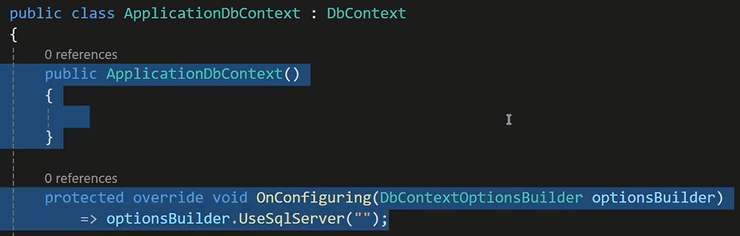


* What happens is when someone needs object from class ApplicationDbContext 🡺 CLR will find out we are allowing dependency injection for ApplicationDbContext so CLR will create object from it with Scoped life time [the default, we didn’t change it in the dependency injection AddDbContext method] 🡺 creating object from ApplicationDbContext depends on dependency injection from object DbContextOptions[remember the Constructor] which we also allowed its dependency injection service through the same method [AddDbContext] 🡺 CLR will create object from class DbContextOptions with lifetime Scoped 🡺 and when creating object from DbContextOptions it uses options-builder 🡺 so now we configured the OptionsBuilder no need to configure it in the OnConfiguring method again



So we have 2 ways one is for the dependency injection way and the other normal way //explain them later

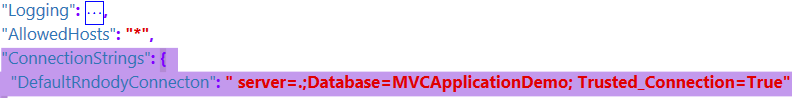
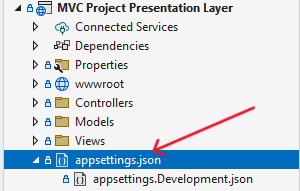
🡺normal way , we used it in the console application as in console we don’t have dependency injection



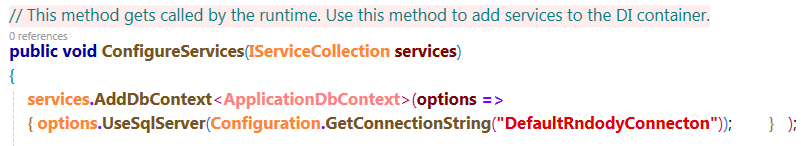
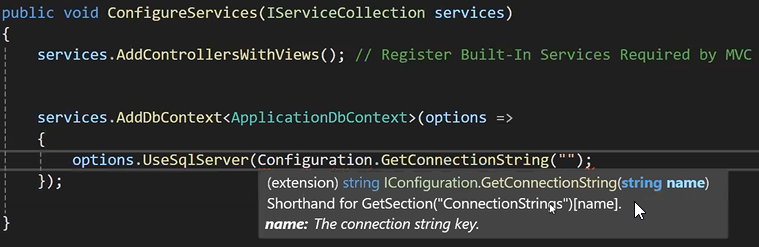
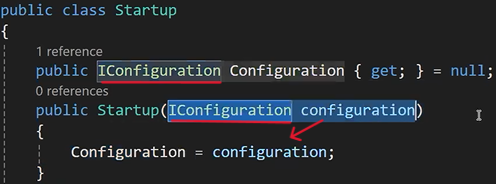
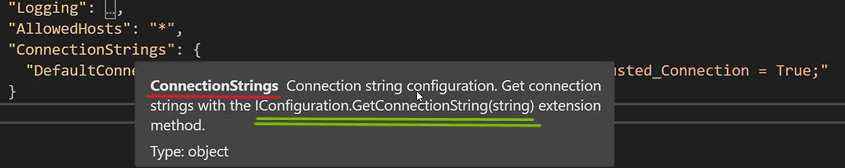
**App Settings – Connection String:**

Writing the connection string in source code is not valid as it changes from environment to another and it’s available for anyone to see it and it should be encrypted as well

We write the connection string in that file

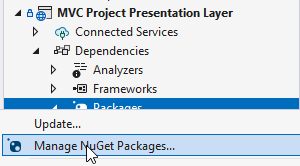


🡺



**Generate Migration:**

We need to run the migration on the project which has appsetting as it has the connection string so we will add the package in that project [MVC Project /presentation layer]



Installing package on the startup project [MVC/presentation layer] and adding migration on the data access layer



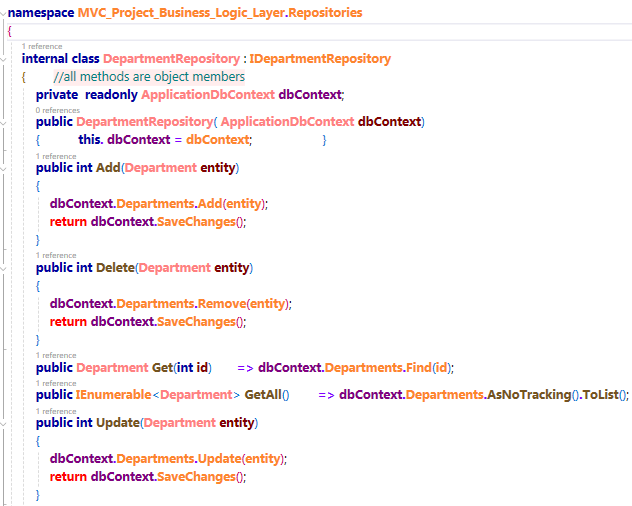
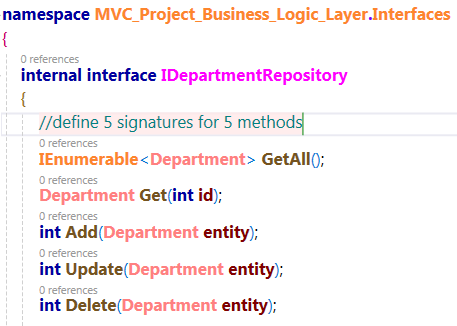
1. **Business Logic Layer**

Use generic repository design pattern

Create 2 folders in the business logic layer project [Interfaces & Repositories]

Repositories is service contain crud operations or behaviors for table it is related to

Repository deal with table using DbContext class



1. **Presentation Layer – MVC Template**

* revision

**Department Controller – Dependency Injection:**

**Department Controller – Index**

**Department Controller - Create**