**Employee Module:**

**Employee Data Access Layer:**

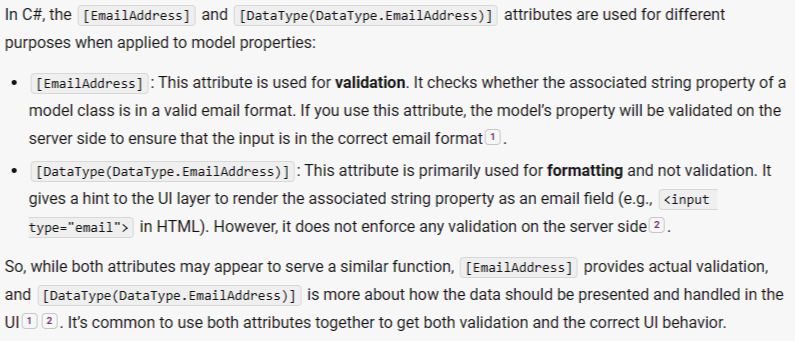
As we are working 3-layer architecture we start with the DAL layer

Adding new Model Employee in the Models folder

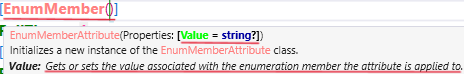
Make the Entity[class] Public as we need to access it from other layers

Any validation attribute we use to be mapped in SQL Server Service we won’t use it, instead we ‘ll write it as Fluent API in the Configuration Class

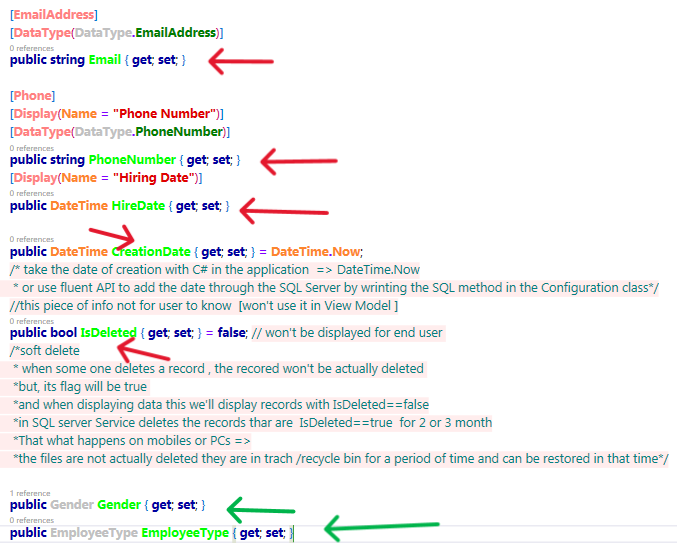
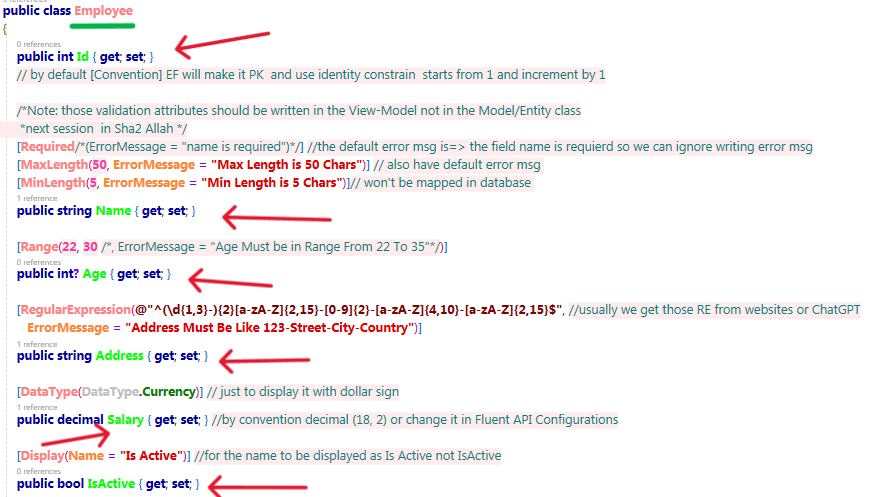
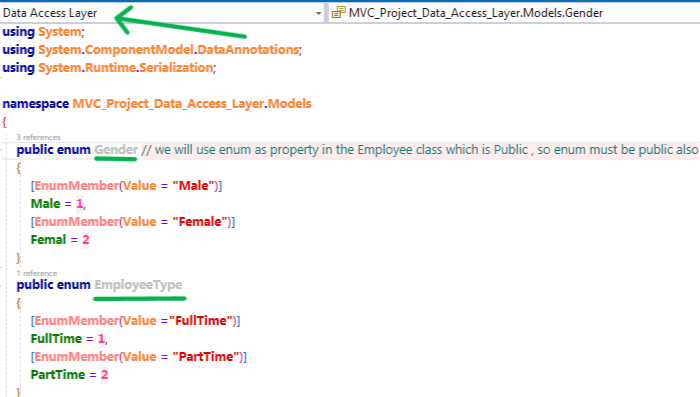
And the validation attributes we use for application as client side and server side will be written in the View-Model



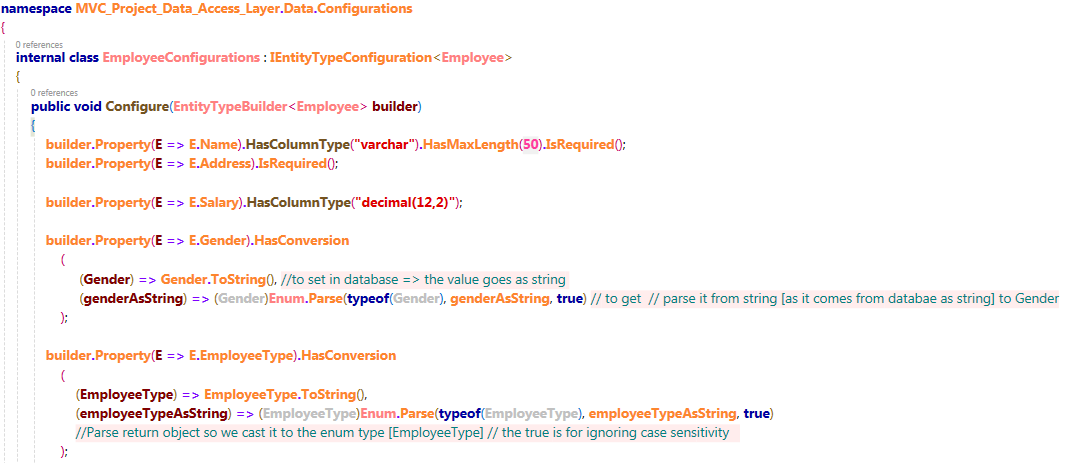
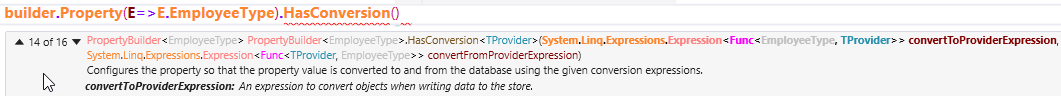
If we make input for enum property and we didn’t specify the type of the input … what’ll be its default type? 🡺 It will be text from type int (input will take 1, 2…) [remember that the enum is numeric value inherits from int] as if we re-represented those numeric values and gave them names[label], we need to store the label[string] values in the database not the numeric values 🡺 use the and handle it in the fluent APIs configurations



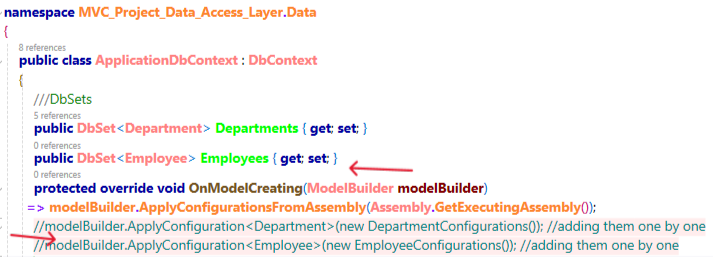
Employee Model 🡺



In the Configurations folder create EmployeeConfigurations implements IEntityTypeConfiguration interface



Go to our DbContext Class and add a dataset for the Employee and add the configuration in the OnModelCreating method for EmployeeConfigurations class if you are not using the method that add all classes that implements the IEntityTypeConfiguration

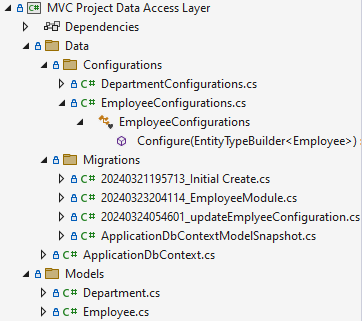


Remember: make sure the MVC project [presentation layer] is the startup project that has the appsetting

When adding migration and updating database use the default project 🡺 data access layer not the startup project



Now add migration then make sure that all information in the up method that will be mapped in database are okay then update database



**Employee Business Logic Layer:**

In repository design pattern 🡺 for each entity/model/domain [domain means range of like domain of departments/employees 🡺 range/group of departments/employees], we create repository

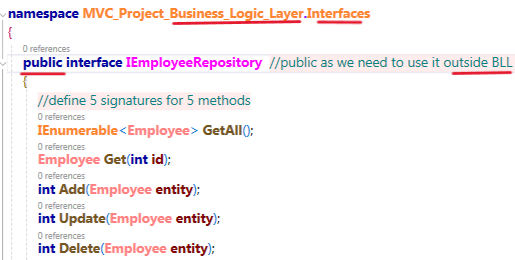
The repository has some behaviors like get /get all /create/update/delete record

We need to grantee the repository behaviors so we created folder for interfaces in the business logic layer and add for each repository an interface to sign the contract when implementing it

We have another folder in the business logic layer for the repositories that contain the behaviors

For each model 🡺 repository 🡺 for each repository 🡺 IRepository 🡺 in the BLL

IEmployeeRepository 🡺



EmployeeRepository 🡺its implementation for the interface methods won’t change from the implementation of the DepartmentRepository same as the implementation of the 5 methods of the ProductRepository all are the same implementation for the 5 methods

Since the Code in all XRepository classes is the same, we are going to use generic type so that we don’t repeat our code

**Generic Repository:**

In interfaces folder in the BLL layer add new interface called IGenericRepository <T> 🡺 write our method signatures inside it 🡺 then make the other interfaces inherit that Interface

When we inherit it in IDepartmentRepository the T will be Department and in IEmployeeRepository the T will be Employee

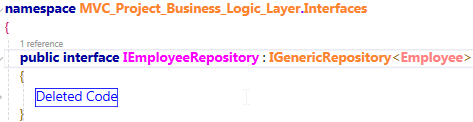
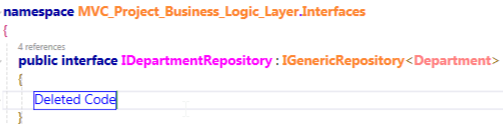
We need to add constrain on the T type to be Model/Entity/Domain

To make the type T only Model not any class 🡺 in the DAL 🡺 Models Folder 🡺 add class called ModelBase/BaseEntity🡺 inside it add the common properties between all our Models 🡺 in our case we only have one common property which is PK and its name is Id

Make the models inherit the ModelBase class and remove the Id property as they will inherit it from parent [ModelBase]🡺 to say that Employee is A ModelBase & Department is A ModelBase

Go to the IGenericRepository interface and set condition that the T type must be ModelBase [and any class inherits from it]

Now the IEmployeeRepository & IDepartmentRepository interfaces are empty now, so they are useless



But in case the interface has a method signature specific for its type then it will be no longer useless

Ex: