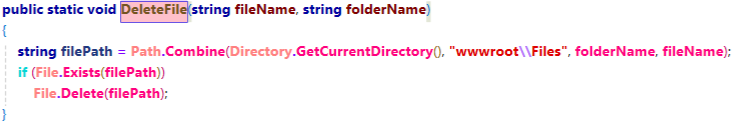
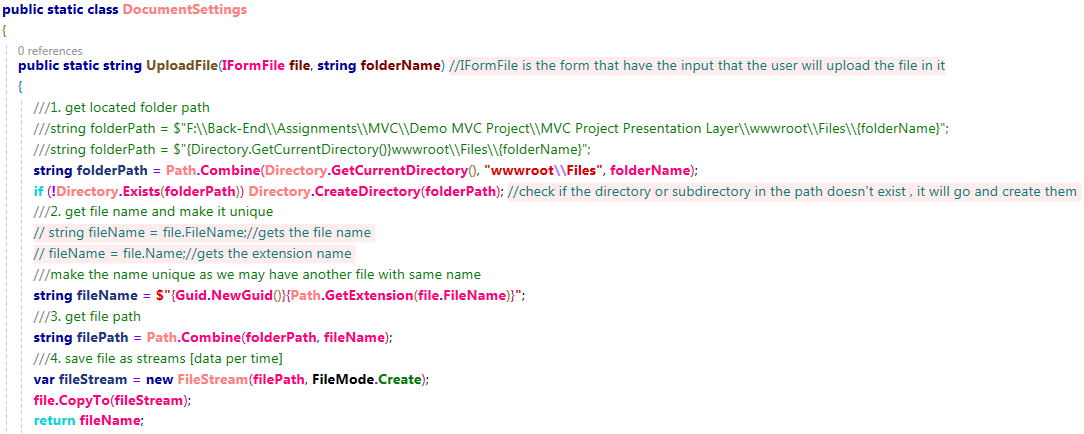
**Document Settings:**

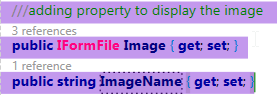
* We need to make user able to upload & download file [any file type]
* In case we are working Onion architecture pattern, the class that handle the document settings will be in the infrastructure layer
* In 3-layer architecture pattern the document settings class will be in the presentation layer 🡺 in the Helpers folder 🡺 static class called DocumentSettings have 2 behaviors [upload and delete ]
* Note: when we upload file on server we do 2 things 🡺 Create file location & we fill this file location with an object that will contain the data of the file 🡺 the object is filled by streaming method [data/time] 🡺 those functionalities and ready-made we won’t build them from scratch
* We have 2 approaches for dealing with files:
* Upload the file in database 🡺 add column for that file [ex: image for each employee 🡪 add column of type binary/image in employees table that will hold the image as 0s and 1s] 🡺 this approach is not ideal as it will take time and processing for converting 0s and 1s ⬄ image each time we need to set or retrieve the image
* We upload the file on the server that the app is deployed on or another server and just store its path only [store the path as string]
* Remember we store the static files in the wwwroot folder 🡺create folder [Files] that is classifies to another folders for each file type 🡺 create folder for images for now[upload image for each employee] 🡺 right click on the Files folder and copy the full-path [it’s better to use the Path class and use the Combine method to add the path of the file]



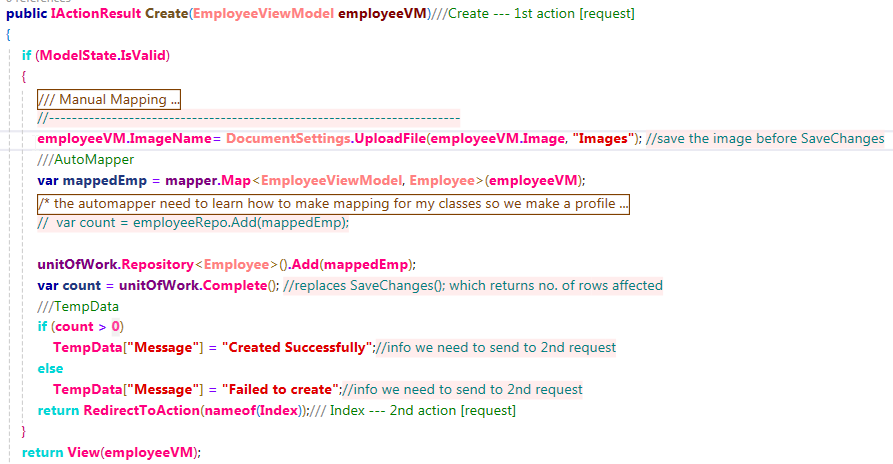
**Upload Image:**

Add column in the Employees table to hold the image name🡺 go to DAL layer in the Employee Model add new string property 🡺 add migration and update database

Go to Employee views [CreateEditEmployeePartial] 🡺 add div for the image to allow user to browse his files and upload it [remember we get the data we display in the view from the ViewModel class, so add property of type IFormFile for the image and another one for the Image name to map ViewModel ⬄ Model]



In the EmployeeController 🡺 Create action 🡺



Remember to edit the encryption type for the HTML form 🡺 in the Create View



* Displaying the uploaded Image in the other views



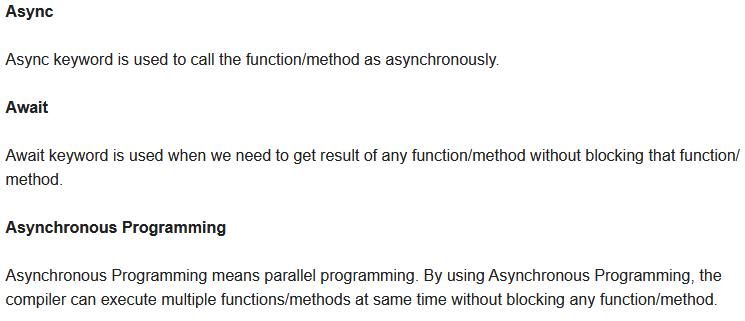
**Delete Image:**

On deleting Employee, we should delete his image from server

Remember to save the name of the image before deleting the employee, as if we delete the employee the name of the image will be deleted without deleting the image so the image will be on server after deleting the employee

We can store the image name in an input in the view or send it as extra information in temp data

**Asynchronous**:



**Back-End Security:**

Authenticating: who are you [user name and password]

Authorization: what are the privileges allowed to you [user roles]

* If we don’t have roles, then the authentication = authorization

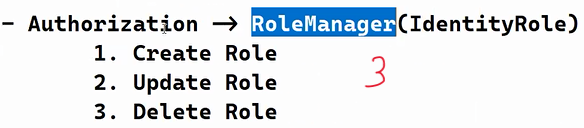
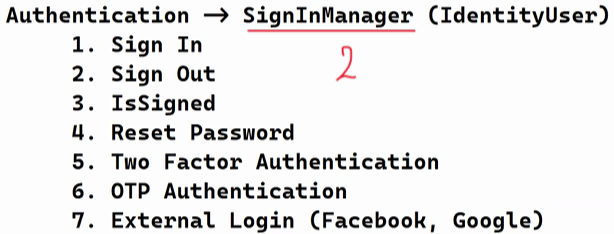
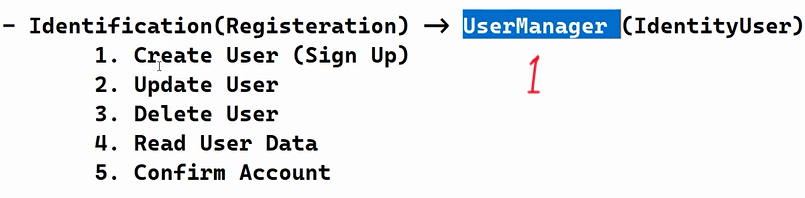
In the security module we have 2 important domain models [user & role], the relation between users and roles is many to many relationship

**Microsoft Identity Package:**

We won’t implement the security module from scratch, we are going to use Identity package

This package has 3 main services

* this service internally deals with user repository but we don’t need to create user repository class as it has one called UserStore 🡺 deals with DbContext 🡺 Entity Framework

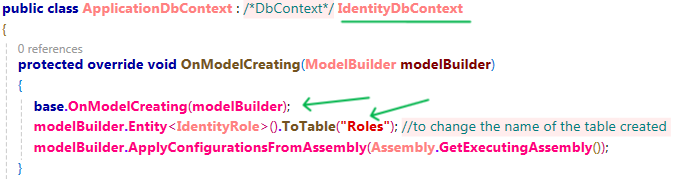


Install the package in the DAL [we need to use it in the PL and the DAL]



We don’t need to create classes in the DAL [the package already has the 2 domain models we need ‘IdentityUser & IdentityRole’]

In the ApplicationDbContext create DbSet for each new class [we can inherit from class IdentityDbContext instead of re-creating the 7 security modules]



**Account Controller [Sign-up]:**

Add controller for auth 🡺 [AccountController]

In the shared folder add new razor page layout to add the design for the template we are going to use 🡺 past the HTML code in that layout

In the wwwroot we have CSS folder that contains Site.CSS file add another CSS file to add our style in it

Create view for the sign up action and create for it view-model [add new folder in the ViewModel folder for the user class views-models]