

EF Core using Code first approach using a Console application

Objectives:

1. Why Entity framework core?
 - Development approaches
2. Perform any 2 CRUD operation EF core for Database first approach using a Console application
 - NuGet package inclusion: Microsoft.EntityFrameworkCore.SqlServer, Microsoft.EntityFrameworkCore, Microsoft.EntityFrameworkCore.Tools, Use it in NuGet package manager console to the scaffold, Add & Save data.

The **EXPERIENCEPOST** web site maintenance team wants **IN MEMORY REPOSITORY** employee details to be pushed into **SQL REPOSITORY** before switching back **IN MEMORY REPOSITORY** to **SQL REPOSITORY**.

The technical details are as follows

Develop the skill post website www.EXPERIENCEPOST.com MVC web project with Entity Framework **Code First approach Repository Pattern** which before start functioning push all in-memory collection data to be upload SQL-repository in the database for the details are as follows.

1. Create model Classes

Create classes for Customer and Address under the Models folder, these classes are used as entities and entities set. These classes will have a mapping with a database because we are using the code-first approach and these classes will create a table in a database using the **DbContext** class of Entity Framework.

Employee: Class

Employee ID (PRIMARY KEY): Integer

First Name: String

Last Name: String

Password: String

Land Line: String

Cell Number: String

Email: String

Skill: Class

Skill Id: (PRIMARY KEY): Integer

Employee ID (FOREIGN KEY): Integer

Skill Name: String

Role: String

Experience In Years: Integer

PostalCode: String

Employee: virtual Customer

IEmployeeRepository: Interface

```

ClsEmployee GetEmployee (ClsEmployee employee);
IEnumerable<ClsEmployee> GetAllEmployee ();
ClsEmployee Add (ClsEmployee employee);
ClsEmployee GetEmployeeByID (int id);
ClsEmployee Update (ClsEmployee employeeChanges);
ClsEmployee Delete (int id);
ClsSkill GetSkill (int Id);
IEnumerable<ClsSkill> GetAllSkill (int Id);
void AddSkill (ClsSkill skill);
Void DeleteSkill (int id);

```

AppDbContext: DbContext Class

```

public AppDbContext (DbContextOptions<AppDbContext> options) : base(options)
{
}

protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Seed();
}

public DbSet<Employee> Employees { get; set; }
public DbSet<Skill> Skills { get; set; }

```

Hint: Add modelBuilderExtensions class of static type and

ModelBuilderExtensions : static class

```

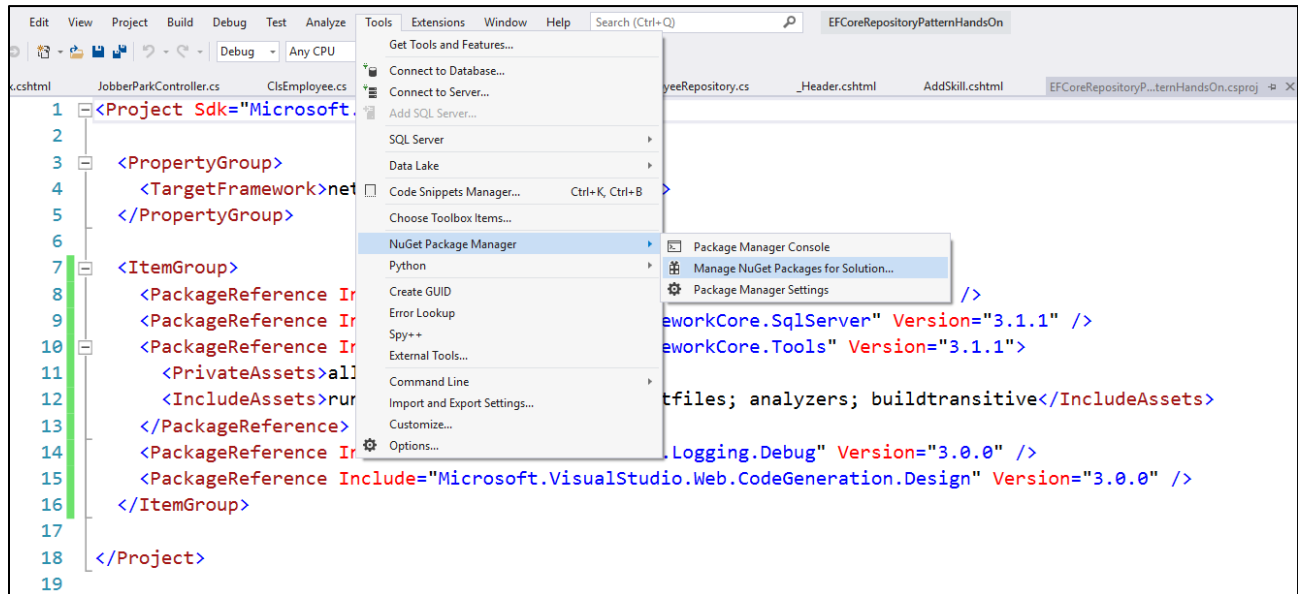
public static void Seed (this ModelBuilder modelBuilder)
{
    modelBuilder.Entity<ClsEmployee> ().HasData (
        new ClsEmployee {EmpID = 1, FirstName = "Aaron", LastName =
            "Hawkins", Password = "arron@123", CellNumber = "(660) 663-4518", Email =
            "aron.hawkins@aol.com" },
        new ClsEmployee {EmpID = 2, FirstName = "Hedy", LastName =
            "Greene", Password = "hedy@123", CellNumber = "(608) 265-2215", Email =
            "hedy.greene@aol.com" },
        New ClsEmployee { EmpID = 3, FirstName = "Melvin",
            LastName = "Porter", Password = "melvin@123", CellNumber = "(959) 119-
            8364", Email = "melvin.porter@aol.com" }
    );

    modelBuilder.Entity<ClsSkill> ().HasData (
        new ClsSkill {SkillId = 1, EmployeeID = 1, SkillName = "Microsoft Office
            Suite", Role = "Business Analyst", ExperienceInYears = 2},
        new ClsSkill {SkillId = 2, EmployeeID = 1, SkillName = "Testing", Role =
            "Developer", ExperienceInYears = 3},
        new ClsSkill {SkillId = 3, EmployeeID = 1, SkillName = "Stakeholder
            Management", Role = "Project Lead", ExperienceInYears = 4}
    );
}

```

Hint: Use **DataAnnotations** namespace to define Primary Key, Required Field, Email Address validation, Foreign Key, Display Name.

2. Open NuGet Manager and add the following packages



Add the following packages.

Microsoft.EntityFrameworkCore

Microsoft.EntityFrameworkCore.Tools

Microsoft.EntityFrameworkCore.SqlServer

3. Add **SQLEmployeeRepository**: Class

Implement interface: **IEmployeeRepository**

Hint: Reference code

```
private readonly AppDbContext context;
public SQLEmployeeRepository (AppDbContext context)
{
    this.context = context;
}

public Employee Add (Employee employee)
{
    context.Employees.Add(employee);
    context.SaveChanges();
    return employee;
}
```

4. Make the changes in the Startup.cs code method

ConfigureServices Reference code is as follows

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContextPool<AppDbContext>(
        options =>
        options.UseSqlServer(_config.GetConnectionString("EmployeeDBConnection")));
    services.AddMvc().AddXmlDataContractSerializerFormatters();
    services.AddScoped<IEmployeeRepository, SQLEmployeeRepository>();
    //services.AddTransient<IEmployeeRepository, InMemoryRepository>();

    services.AddControllersWithViews();
}
```

Note: **EmployeeDBConnection** is the name of the connection string, specified in the

appsettings.json

```
"ConnectionStrings": {
    "EmployeeDBConnection": "Data Source= (localdb)\\MSSQLLocalDB;Initial
Catalog=Database; Trusted_Connection=True"
}
```

5. Migrate and generate the table in the database with the following command

PM> Add-Migration

Note: provide the name to this transaction

PM> update-database

6. Add Controller **ExperiencePostController** with following reference code

```
public class ExperiencePostController: Controller
{
    private readonly IEmployeeRepository _employeeRepository;

    public ExperiencePostController (IEmployeeRepository
employeeRepository)
    {
        _employeeRepository = employeeRepository;
    }
    public ActionResult Index()
```

```
{  
    return View();  
}  
  
[HttpGet]  
public ActionResult AddSkill(int id)  
{  
    Skill skill = new Skill();  
    skill.EmployeeID = id;  
  
    return View(skill);  
}
```

7. After registration Employee has to provide credentials by the time of login.

The screenshot displays the 'EXPERIENCE POST' web application. At the top, a dark header bar contains the text 'EXPERIENCE POST' on the left and a login section on the right. The login section includes a text input field with the email 'aron.hawkins@aol.com', a password input field with masked characters '.....', and a blue 'Login' button. Below the header, the main content area features a 'Sign In' section on the left with five input fields: 'First Name', 'Last Name', 'Password', 'Cell Number', and 'Email'. A blue 'Create' button is positioned below the 'Email' field. To the right of the sign-in form is a large graphic with the word 'POST' in large, bold, blue letters, and the word 'EXPERIENCE' in smaller, blue letters below it, all set against a light blue background.

8. After login, the employee will switch to Home view / Details, where employee details will be flashes along with their experiences.

EXPERIENCE POST

SignOut

Details

Aaron
Hawkins
(660) 663-4518
aron.hawkins@aol.com
[Add Skill](#) | [Edit Details](#)

Skill Name	Role	Experience In Years	
Microsoft Office Suite	Business Analyst	2	Delete
Testing	Developer	3	Delete
Stakeholder Management	Project Lead	4	Delete

9. When the Employee will click on the edit details the view will switch to Edit Employee details view where Employee will allow to amendment in their details.

EXPERIENCE POST

SignOut

Edit your details

First Name

Last Name

Password

Cell Number

Email

[Save](#)

[Back](#)

10. Similarly, when the employee will click on the delete skill, it will delete the employee skill from the repository.

EXPERIENCE POST

SignOut

Details

Aaron
Hawkins
(660) 663-4519
aron.hawkins@aol.com
[Add Skill](#) | [Edit Details](#)

Skill Name	Role	Experience In Years	
Microsoft Office Suite	Business Analyst	2	Delete
Testing	Developer	3	Delete
Stakeholder Management	Project Lead	4	Delete

11. When Employee will click on the Add Skills link, the view will switch to add skill view. Where Employee has to input skill details.

EXPERIENCE POST

SignOut

Details

Aaron
Hawkins
(660) 663-4519
aron.hawkins@aol.com
[Add Skill](#) | [Edit Details](#)

Skill Name	Role	Experience In Years	
Microsoft Office Suite	Business Analyst	2	Delete
Stakeholder Management	Project Lead	4	Delete

12. After providing skill details, the employee has to click on the create button and skill details were added to the employee skill details and the view will switch back to the details view of the employee.

EXPERIENCE POST

SignOut

Add Skill

EmployeeID

Skill Name

Role

Experience In Years

Create

[Back](#)

EXPERIENCE POST

SignOut

Details

Aaron
Hawkins
(660) 663-4519
aron.hawkins@aol.com
[Add Skill](#) | [Edit Details](#)

Skill Name	Role	Experience In Years	
Microsoft Office Suite	Business Analyst	2	Delete
Stakeholder Management	Project Lead	4	Delete
Testing	Developer	4	Delete

13. When the Employee will click on the Sign-out, the view will switch back to the login view.

EXPERIENCE POST

Login

Sign In

First Name

Last Name

Password

Cell Number

Email

Create

POST

EXPERIENCE