

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 their data to be saved in the database and have the flexibility to use the in-memory data if in case the database server will be down.

Now the EXPERIENCEPOST web site maintenance team wants their database to stop for **log shipping**, for that purpose the team wants the database to stop for time being. That for reason team wants the repository to switch from SQL REPOSITORY to IN MEMORY 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** 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)
{
}
public DbSet<Employee> Employees { get; set; }
public DbSet<Skill> Skills { get; set; }

```

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

2. Add InMemoryRepository: Class

Implement interface: IEmployeeRepository

Hint: Reference code

```

public class InMemoryRepository : IEmployeeRepository
{
    private static List<Employee> _employeeList = new
List<Employee>()
    {
        new
Employee(){EmpID=1,FirstName="Aaron",LastName="Hawkins",Password=
"arron@123",CellNumber="(660) 663-
4518",Email="aron.hawkins@aol.com" },
        new
Employee(){EmpID=2,FirstName="Hedy",LastName="Greene",Password="h
edy@123",CellNumber="(608) 265-2215",Email="hedy.greene@aol.com"
},
    }
}

```

```

        new
Employee(){EmpID=3,FirstName="Melvin",LastName="Porter",Password=
"melvin@123",CellNumber="(959) 119-
8364",Email="melvin.porter@aol.com"}
    };

    private static List<Skill> _skillList = new List<Skill>()
    {
        new
Skill(){SkillId=1,EmployeeID=1,SkillName="Microsoft Office
Suite",Role="Business Analyst",ExperienceInYears=2},
        new
Skill(){SkillId=2,EmployeeID=1,SkillName="Testing",Role="Develope
r",ExperienceInYears=3},
        new
Skill(){SkillId=3,EmployeeID=1,SkillName="Stakeholder
Management",Role="Project Lead",ExperienceInYears=4}
    };

    public ClsEmployee Add(ClsEmployee employee)
    {
        if (_employeeList.Count == 0)
        {
            employee.EmpID = 1;
        }
        else
        {
            employee.EmpID = _employeeList.Max(e => e.EmpID) + 1;
        }
        _employeeList.Add(employee);
        return employee;
    }
}

```

3. 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();
}
```

4. 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);
    }
}
```

5. Add view `SignIn` and registered the Employee.

The screenshot shows the 'EXPERIENCE POST' application interface. At the top, there is a dark header bar with the text 'EXPERIENCE POST' on the left, and two input fields labeled 'Enter email' and 'Enter password' on the right, followed by a blue 'Login' button. Below the header, the main content area has a white background. On the left, there is a 'Sign In' section with the following fields: 'First Name', 'Last Name', 'Password', 'Cell Number', and 'Email'. Each field has a corresponding input box. Below the 'Email' field is a blue 'Create' button. On the right side of the main content area, there is a large graphic with the word 'POST' in a large, bold, serif font, and the word 'EXPERIENCE' in a smaller, bold, sans-serif font, all in blue. The 'EXPERIENCE' text is set against a blue rectangular background.

This screenshot shows the same 'EXPERIENCE POST' application interface as the previous one, but with validation errors displayed below the input fields. The 'First Name' field has the error message 'Please Enter First Name e.g. John'. The 'Last Name' field has the error message 'Please Enter Last Name e.g. Doe'. The 'Password' field has the error message 'Password should not be blank'. The 'Cell Number' field has the error message 'Cell Number should not be blank'. The 'Email' field has the error message 'Enter valid email address'. The 'Create' button is now highlighted with a red rectangular border. The header and the 'POST EXPERIENCE' graphic remain the same.

6. After enter details employee has to click on the create button and get registered on the website.

EXPERIENCE POST

Enter email Enter password Login

Sign In

First Name
Aaron

Last Name
Hawkins

Password
.....

Cell Number
(660) 663-4518

Email
aron.hawkins@aol.com

Create

POST
EXPERIENCE

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

EXPERIENCE POST

aron.hawkins@aol.com Login

Sign In

First Name

Last Name

Password

Cell Number

Email

Create

POST
EXPERIENCE

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

1

Skill Name

Testing

Role

Developer

Experience In Years

4

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