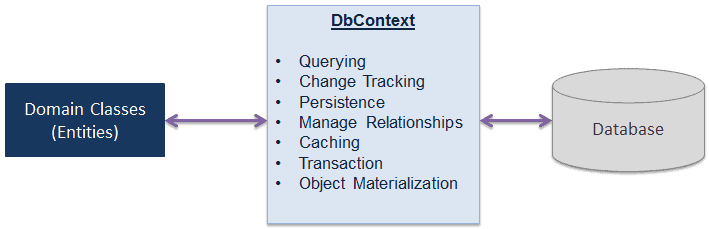
**Objective: To implement CRUD operation using DbContext on ASP.Net Core.**

**Theory:**

**DbContext:**

DbContext is an important class in Entity Framework API. It is a bridge between your domain or entity classes and the database.



DbContext is the primary class that is responsible for interacting with the database. It is responsible for the following activities:

* **Querying:** Converts LINQ-to-Entities queries to SQL query and sends them to the database.
* **Change Tracking:** Keeps track of changes that occurred on the entities after querying from the database.
* **Persisting Data:** Performs the Insert, Update and Delete operations to the database, based on entity states.
* **Caching:** Provides first level caching by default. It stores the entities which have been retrieved during the life time of a context class.
* **Manage Relationship:** Manages relationships using CSDL, MSL and SSDL in Db-First or Model-First approach, and using fluent API configurations in Code-First approach.
* **Object Materialization:** Converts raw data from the database into entity objects.

**Source Code:**

* EmployeeController.cs for CRUD operation using DbContext:

using EmployeeManagement.Data;

using EmployeeManagement.Models;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace EmployeeManagement.Controllers

{

public class EmployeeController : Controller

{

private EMSContext db;

public EmployeeController(EMSContext \_db)

{

db = \_db;

}

public IActionResult Index()

{

var employees = db.Employees.ToList();

return View(employees);

}

public IActionResult Detail([FromQuery] int id)

{

var employee = db.Employees.Find(id);

return View(employee);

}

[HttpGet]

public IActionResult Add()

{

return View();

}

[HttpPost]

public ActionResult<string> Add(Person person)

{

db.Employees.Add(person);

db.SaveChanges();

return RedirectToAction(nameof(Index));

}

public ActionResult Edit(int id)

{

var employee = db.Employees.Find(id);

return View(employee);

}

[HttpPost]

public ActionResult Edit(Person person)

{

db.Employees.Attach(person);

db.Employees.Update(person);

db.SaveChanges();

return RedirectToAction(nameof(Index));

}

public ActionResult Delete(int id)

{

var employee = db.Employees.Find(id);

return View(employee);

}

[HttpPost]

public ActionResult Delete(Person person)

{

db.Employees.Attach(person);

db.Employees.Remove(person);

db.SaveChanges();

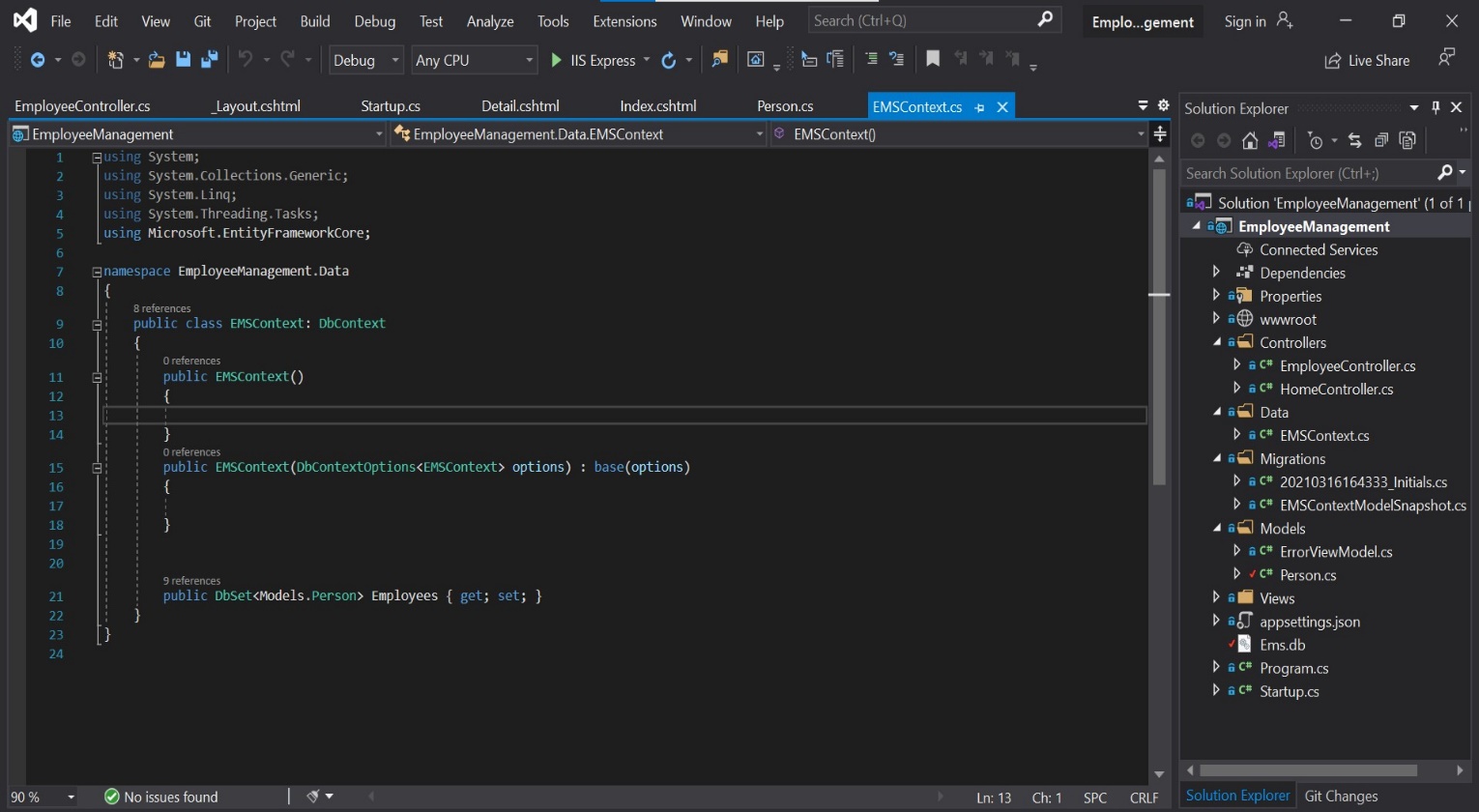
return RedirectToAction(nameof(Index));

}

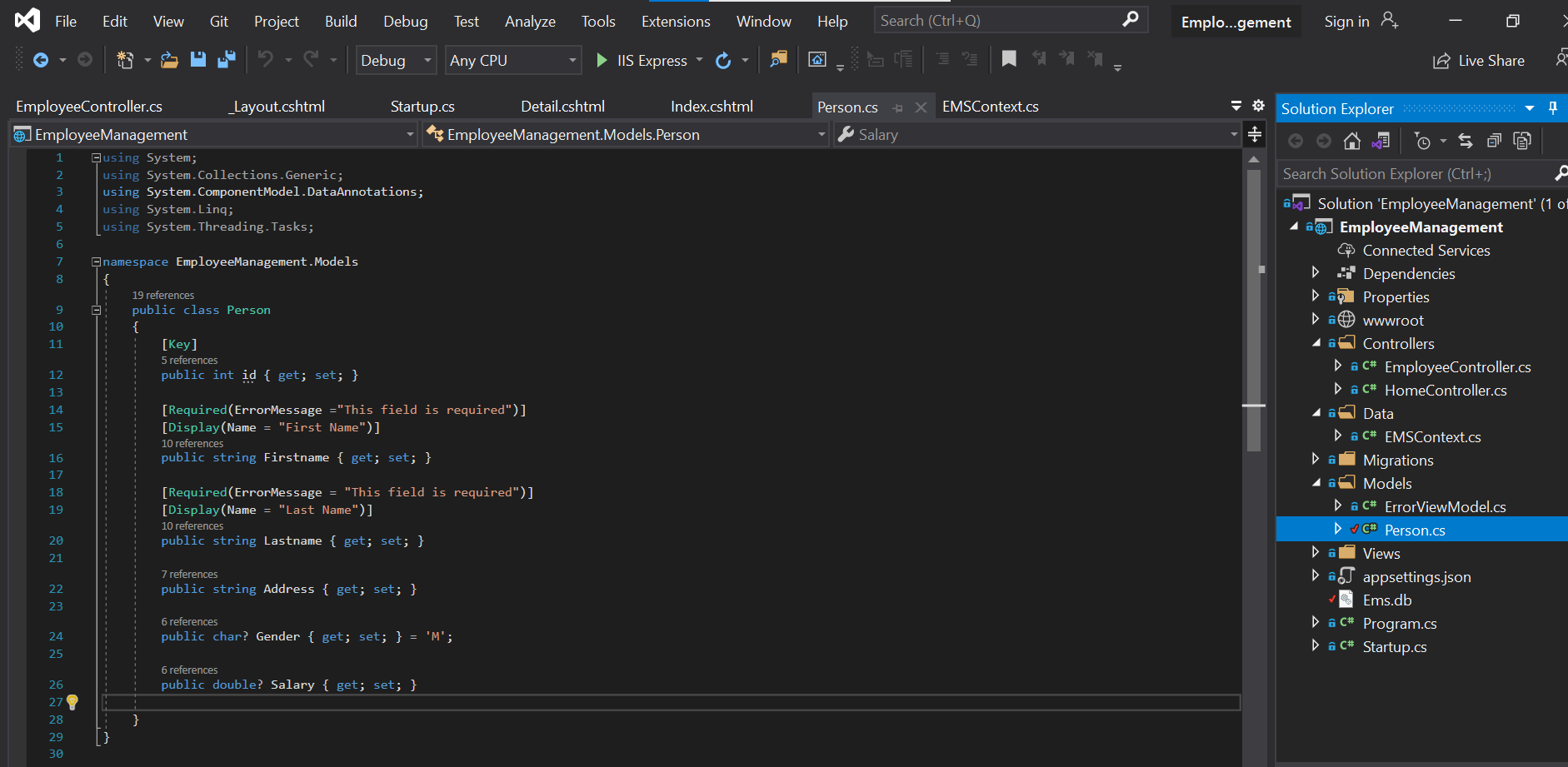
}

}

* To use DbContext in our application, we need to create the class that derives from DbContext, also known as context class. This context class typically includes [DbSet<TEntity>](https://docs.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.dbset-1?view=efcore-2.0" \t "_blank) properties for each entity in the model. Consider the following example of context class in EF Core.



* **Model** contains the Person.cs file which shows the data, logic and rules of our EmployeeManagement application:



* **Views contains the all the views file for the operation of CRUD as following:**
* **Add.cshtml file is for the create operation in our application and this contains following:**

@model Person

@{

ViewData["Title"] = "Add Employee";

string[] genders = { "Male", "Female", "Others" };

}

<h3>Add New Employee</h3>

<hr />

<form **asp-controller**="Employee" **asp-action**="Add" method="post">

<div class="form-group">

<label **asp-for**="Firstname">First Name</label>

<span **asp-validation-for**="Firstname" class="text-danger"></span>

<input **type**="text" class="form-control" **asp-for**="Firstname" placeholder="Enter your first name.">

</div>

<div class="form-group">

<label **asp-for**="Lastname">Last Name</label>

<span **asp-validation-for**="Lastname" class="text-danger"></span>

<input **type**="text" class="form-control" **asp-for**="Lastname" placeholder="Enter your last name.">

</div>

<div class="form-group">

<label **asp-for**="Address">Address</label>

<textarea class="form-control" **asp-for**="Address" placeholder="Enter Address."> </textarea>

</div>

<div class="form-group">

<div>Gender</div>

@foreach (var gender in genders)

{

<div class="form-check form-check-inline">

<input class="form-check-input" **type**="radio" **name**="genderRadio" id="@gender" **asp-for**="Gender" **value**="@gender[0]">

<label class="form-check-label" for="@gender">@gender</label>

</div>

}

</div>

<div class="form-group">

<label **asp-for**="Salary">Salary</label>

<input **type**="number" class="form-control" **asp-for**="Salary" placeholder="Enter your last name.">

</div>

<button type="submit" class="btn btn-primary">Add Employee</button>

</form>

* **Edit.cshtml file is for the update operation in our application and this contain following logic:**

@model Person

@{

ViewData["Title"] = "Update";

}

<h1>Update employee</h1>

<form **asp-controller**="Employee" **asp-action**="Edit" method="POST">

<input **type**="hidden" **asp-for**="id">

<div class="form-group">

<label **asp-for**="Firstname"></label>

<input **type**="text" class="form-control" **asp-for**="Firstname" placeholder="First Name">

<span **asp-validation-for**="Firstname" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Lastname"></label>

<input **type**="text" class="form-control" **asp-for**="Lastname" placeholder="Lastname">

<span **asp-validation-for**="Lastname" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Address"></label>

<textarea class="form-control" **asp-for**="Address" placeholder="Address"></textarea>

</div>

<div class="form-group">

<label **asp-for**="Gender"></label>

<input **type**="text" class="form-control" **asp-for**="Gender" placeholder="Gender">

</div>

<div class="form-group">

<label **asp-for**="Salary"></label>

<input **type**="number" class="form-control" **asp-for**="Salary" placeholder="Salary">

</div>

<button type="submit" class="btn btn-primary">Update Employee</button>

</form>

* **Delete.cshtml file is for the delete operation in our application and this contain following logic:**

@model Person

<h2 class="text-danger">Are you sure to delete folloing?</h2>

<ul class="list-group list-group-flush">

<li class="list-group-item">First Name : @Model.Firstname</li>

<li class="list-group-item">Last Name : @Model.Lastname</li>

<li class="list-group-item">Address : @Model.Address</li>

<li class="list-group-item">Gender : @Model.Gender</li>

</ul>

<form **asp-controller**="Employee" **asp-action**="Delete" method="POST">

<input **type**="hidden" **asp-for**="id" />

<input class="btn btn-danger" type="submit" value="Delete it you fool, don't ask again?">

<a class="btn btn-primary" href="/employee/index">Cancel</a>

</form>

* **Detail.cshtml file is for the read operation in our application and this contain following logic:**

@model Person

@{

ViewData["Title"] = @Model.Firstname + " " + @Model.Lastname;

}

<h1>Employee Detail</h1> <h2><a href="/employee/index">Go back</a></h2>

<ul class="list-group list-group-flush">

<li class="list-group-item">First Name : @Model.Firstname</li>

<li class="list-group-item">Last Name : @Model.Lastname</li>

<li class="list-group-item">Address : @Model.Address</li>

<li class="list-group-item">Gender : @Model.Gender</li>

<li class="list-group-item">Salary : @Model.Salary</li>

</ul>

**And the index.cshtml which**  **renders presentation of the model in a particular format and has following logic:**

@model List<Person>

@{

ViewData["Title"] = "Employee Home";

}

<h1>All Employees [@Model.Count]</h1>

<table class="table">

<thead class="thead-dark">

<tr>

<th scope="col">#</th>

<th scope="col">First Name</th>

<th scope="col">Last Name</th>

<th scope="col">Address</th>

<th scope="col">Gender</th>

<th scope="col">Salary</th>

<th scope="col"></th>

<th scope="col"></th>

</tr>

</thead>

<tbody>

@for (int i = 0; i < Model.Count; i++)

{

<tr>

<td>@(i+1)</td>

<td>

@{

string fn = Model[i].Firstname;

}

<a href="/employee/detail?id=@Model[i].id">@fn</a>

</td>

<td>@Model[i].Lastname</td>

<td>@Model[i].Address</td>

<td>@Model[i].Gender</td>

<td>@Model[i].Salary</td>

<td>

<a class="btn btn-warning" href="/Employee/Edit?id=@Model[i].id">Edit</a>

<a class="btn btn-danger" href="/Employee/Delete?id=@Model[i].id">Delete</a>

</td>

</tr>

}

</tbody>

</table>

**In startup.cs, we maintains the database (here, we Sqlite database) connections string as:**

public void ConfigureServices(IServiceCollection services)

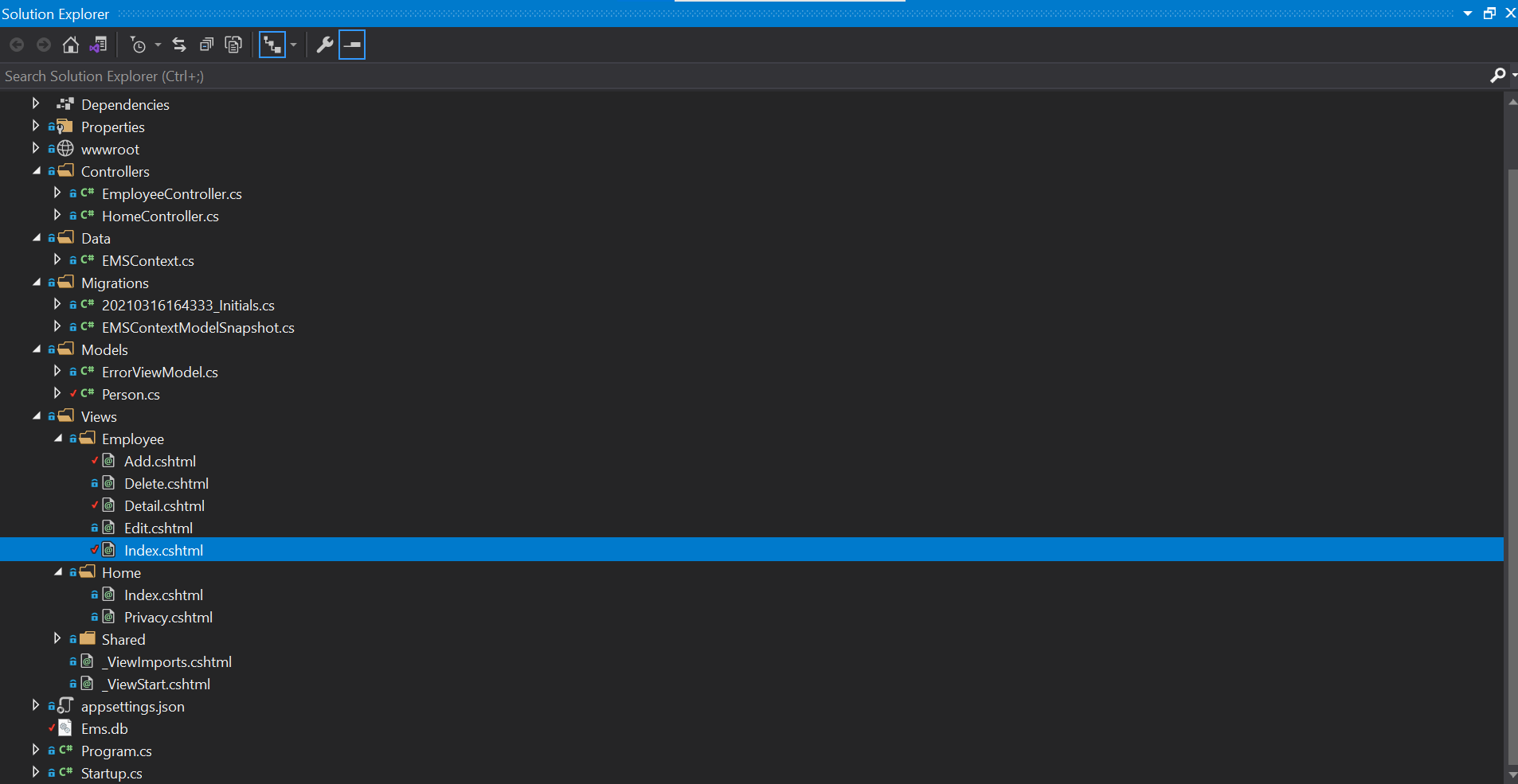
{

services.AddDbContext<EMSContext>(options => options.UseSqlite("Data Source Ems.db"));

services.AddControllersWithViews();

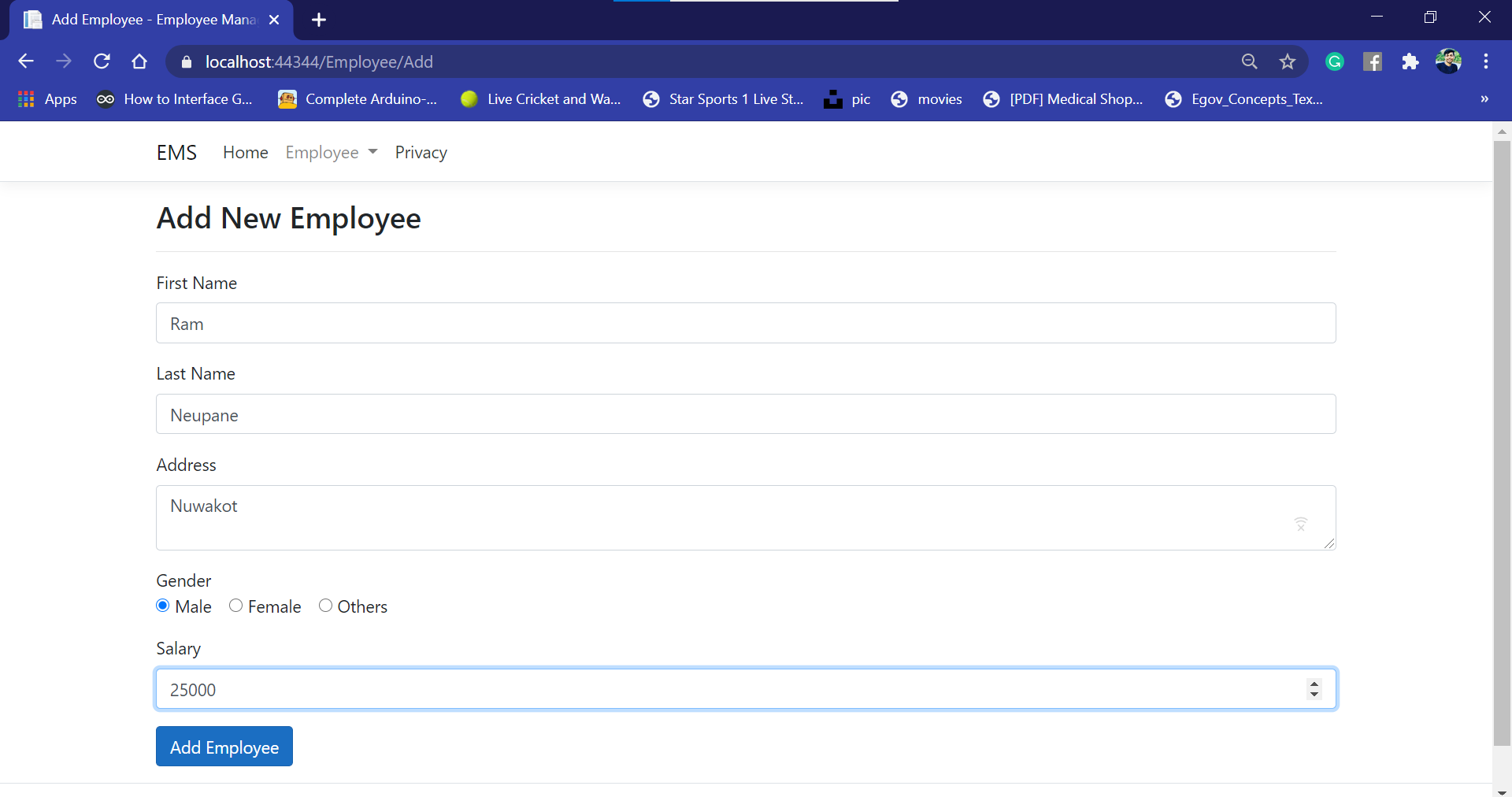
}

**Solution Explorer of our project:**

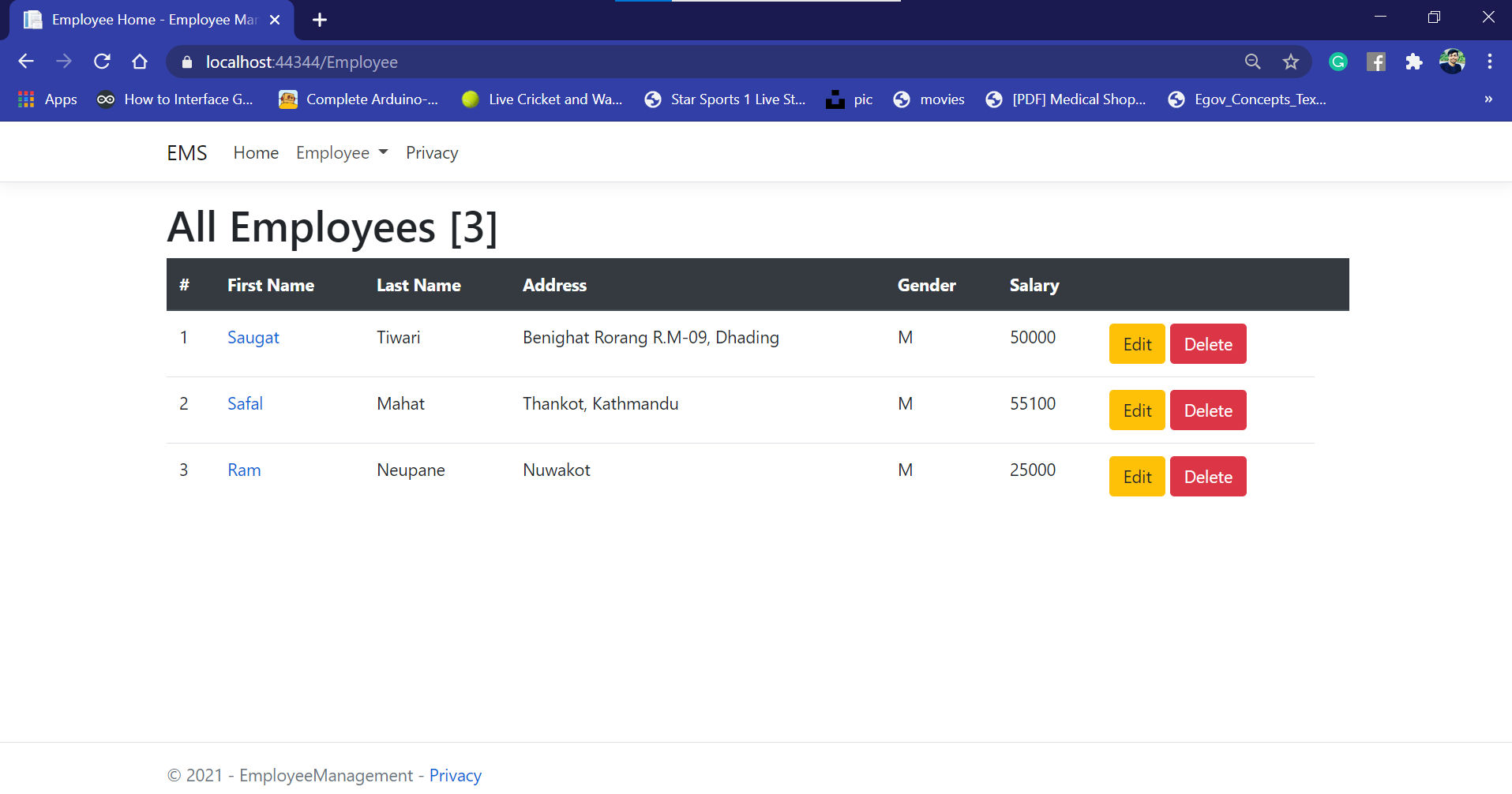
****

**Results on Chrome Browser:**

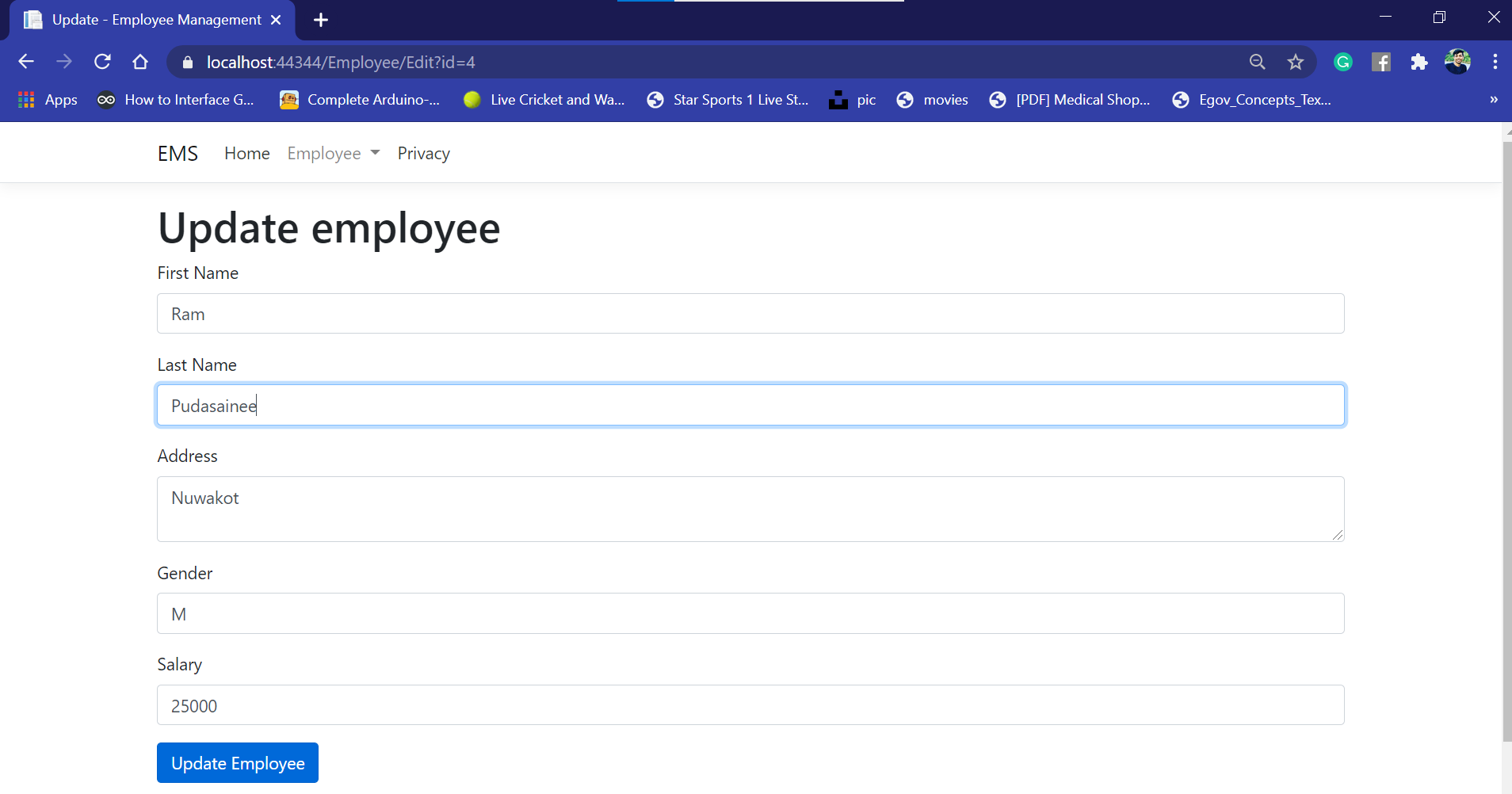
**Add page:**

****

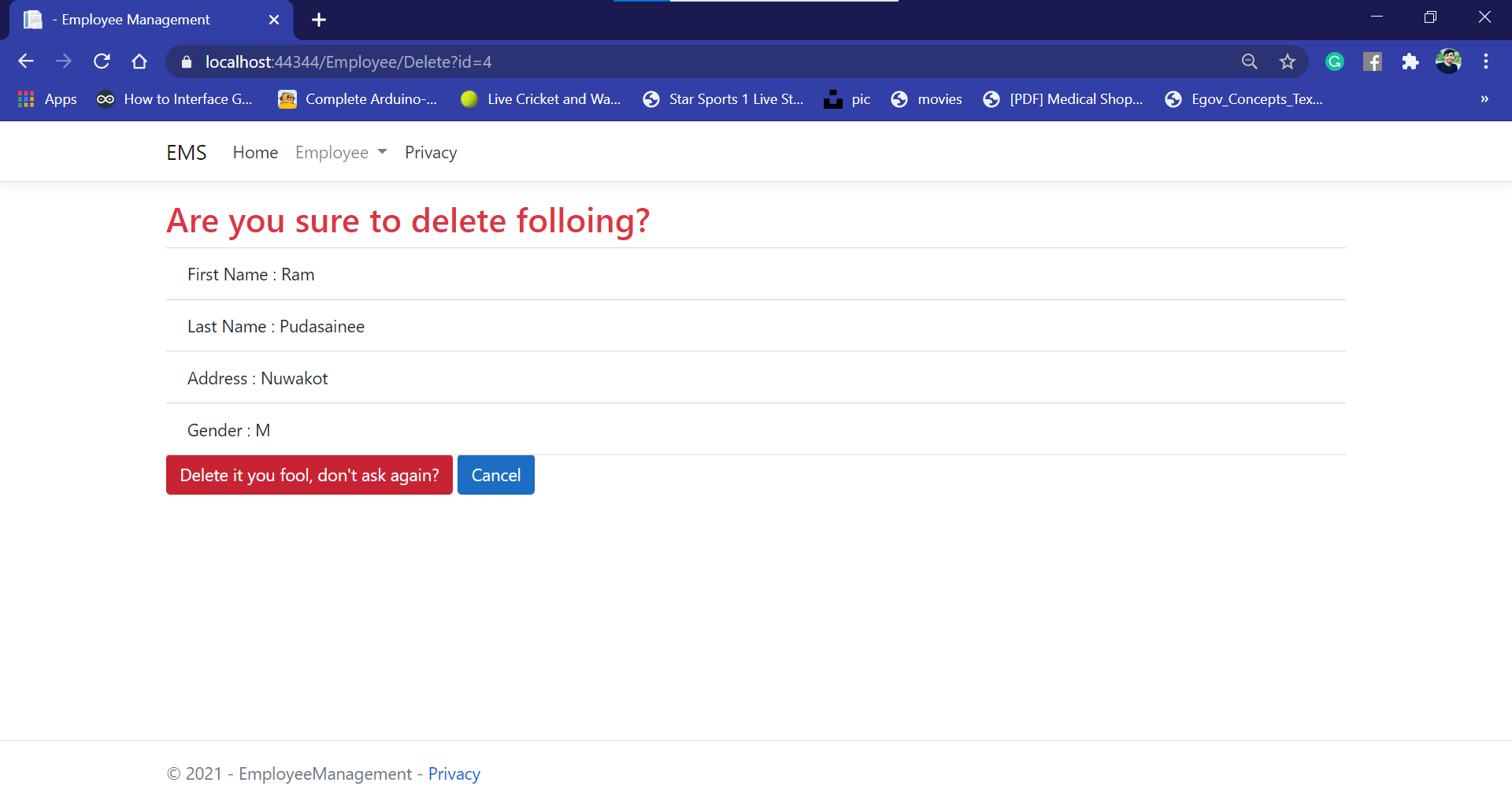
**Employee list page which shows the list of all employees:**

****

**Update page:**

****

**Delete page:**

****

**Conclusions:**

In this lab of Net Centric Computing, we learned about the CRUD using DbContext in EF core on ASP.NET core. In the we create Employee Management System to illustrate the concept of DbContext.