Eksemplarisk Web Application

# 1.Architecture

Opret følgende projekter

* **WebApp** (Razor Pages Web App)
* **ServiceLayer** (.NET Standard class library)
* **DataLayer** (.NET Standard class library)

Opret project-referencene:

* WebApp -> DataLayer
* ServiceLayer -> DataLayer

Installér følgende NuGet-pakker vha. PMC:

Install-Package Microsoft.VisualStudio.Web.BrowserLink -ProjectName WebApp

Install-Package Microsoft.EntityFrameworkCore.SqlServer -ProjectName DataLayer

Install-Package Microsoft.EntityFrameworkCore.Tools -ProjectName DataLayer

WebApp

Tilføj til StartUp.cs:

app.UseBrowserLink();

For InMemory database:

services.AddDbContext<AppDbContext>(options => options.UseInMemoryDatabase("InMemoryDb"));

DataLayer

Opret AppDbContext.cs

public class AppDbContext : DbContext

{

    public AppDbContext(DbContextOptions options)

        : base(options)

    {

    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)

    {

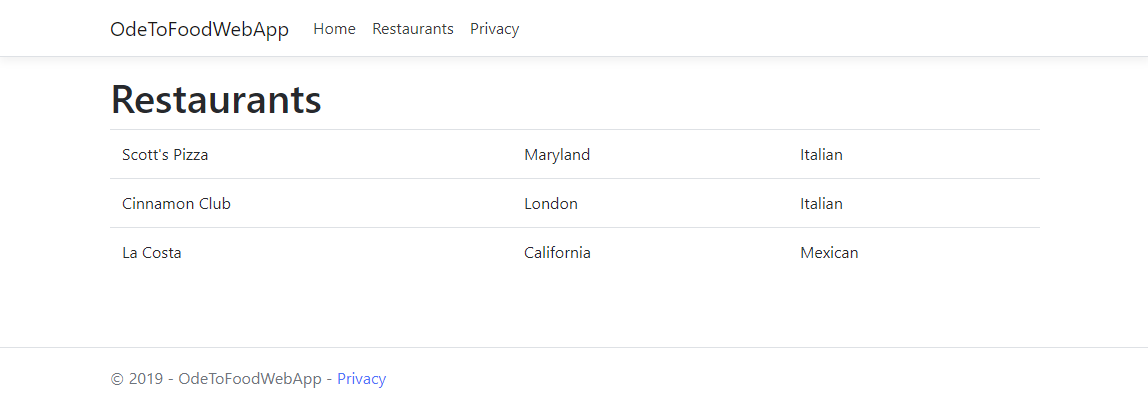
    }

}

Opret folder kaldet **Entities**.

ServiceLayer

2.ListOfRestaurants



DataLayer

Opret klasserne i Entities folderen:

public class Restaurant

{

    public int Id { get; set; }

    public string Name { get; set; }

    public string Location { get; set; }

    public CuisineType Cuisine { get; set; }

}

public enum CuisineType

{

    None,

    Mexican,

    Italian,

    Indian

}

Opret DbSet<Restaurant> og DataSeeding i AppDbContext.cs

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

    modelBuilder.Entity<Restaurant>().HasData(

        new Restaurant { Id = 1, Name = "Scott's Pizza", Location = "Maryland", Cuisine = CuisineType.Italian },

        new Restaurant { Id = 2, Name = "Cinnamon Club", Location = "London", Cuisine = CuisineType.Italian },

        new Restaurant { Id = 3, Name = "La Costa", Location = "California", Cuisine = CuisineType.Mexican }

    );

}

ServiceLayer

Opret interface og implementering af service:

public interface IRestaurantService

{

    IQueryable<Restaurant> GetRestaurants();

}

public class RestaurantService : IRestaurantService

{

    private readonly AppDbContext \_ctx;

    public RestaurantService(AppDbContext ctx)

    {

ctx.Database.EnsureCreated();

        \_ctx = ctx;

    }

    public IQueryable<Restaurant> GetRestaurants()

    {

        return \_ctx.Restaurants;

    }

}

Bemærk ctx.Database.EnsureCreated(), som sikrer at OnModelCreating() bliver kørt og InMemory databasen bliver initialiseret. Fjernes ved brug af manuel Migration.

WebApp

Opret folder Pages | Restaurants

Opret Razor Page kaldet List:

<table class="table">

    @foreach (var restaurant in Model.Restaurants)

    {

        <tr>

            <td>@restaurant.Name</td>

            <td>@restaurant.Location</td>

            <td>@restaurant.Cuisine</td>

        </tr>

    }

</table>

public class ListModel : PageModel

{

    public string Message { get; set; }

    public IEnumerable<Restaurant> Restaurants { get; set; }

    private readonly IRestaurantService \_restaurantService;

    public ListModel(IRestaurantService restaurantService)

    {

        \_restaurantService = restaurantService;

    }

    public void OnGet()

    {

        Restaurants = \_restaurantService.GetRestaurants().ToList();

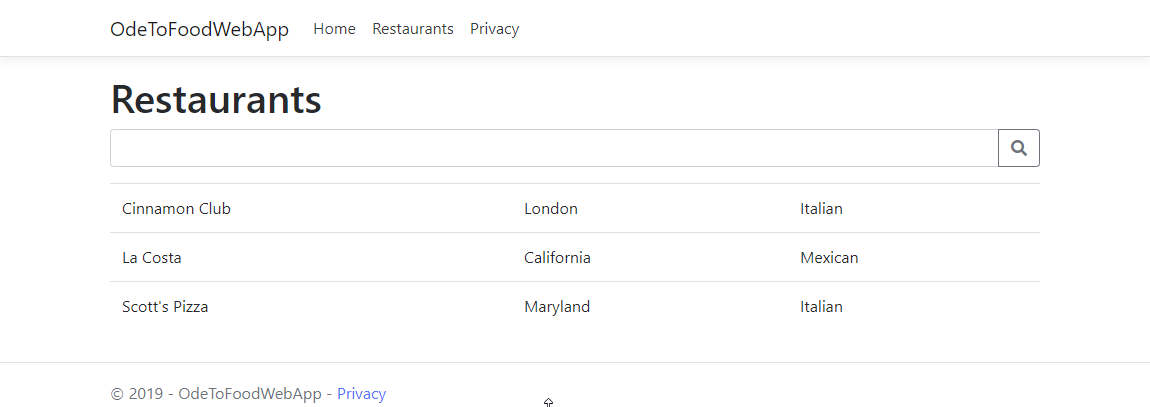
    }

}

Tilføj evt. Restaurants til menuen i \_Layout.cshtml.

3.ModelsModelBinding

Der oprettes en mulighed for at søge (første del af navnet og case-sensitivt):



WebApp

Tilføj en searchbox til List view. Med BootStrap 4 laves en searchbox:

<**form** method="get">

    <div class="input-group mb-3">

        <**input** **type**="search" **asp-for**="SearchTerm" class="form-control">

        <div class="input-group-append">

            <button class="btn btn-outline-secondary" type="submit">

                <i class="fas fa-search"></i>

            </button>

        </div>

    </div>

</**form**>

Bemærk at der er linket til CSS-filen fra Font Awesome i \_Layout.

Man kan evt. demonstrere en input uden brug af asp-for, der så vil kræve både id, name og value:

<input type="search" id="SearchTerm" name="SearchTerm" value="@Model.SearchTerm">

public class ListModel : PageModel

{

    public string Message { get; set; }

    public IEnumerable<Restaurant> Restaurants { get; set; }

    [BindProperty(SupportsGet = true)]

    public string SearchTerm { get; set; }

    private readonly IRestaurantService \_restaurantService;

    public ListModel(IRestaurantService restaurantService)

    {

        \_restaurantService = restaurantService;

    }

    public void OnGet()

    {

        Restaurants = \_restaurantService.GetRestaurantsByName(SearchTerm);

    }

}

Bemærk at der laves en TwoWay Databinding ved at tilføje BindProperty til SearchTerm. Og at den bringes til at supportere Get.

Man man også vise to andre (og mindre smarte) måder:

Uden Model Binding:

string searchTerm = HttpContext.Request.QueryString.Value.Split('=').LastOrDefault();

One-Way binding via parameter:

public void OnGet(string searchTerm)

{

    Restaurants = \_restaurantService.GetRestaurantsByName(searchTerm).ToList();

}

ServiceLayer

RestaurantService udvides:

public interface IRestaurantService

{

    IQueryable<Restaurant> GetRestaurants();

    IQueryable<Restaurant> GetRestaurantsByName(string name = null);

}

public IQueryable<Restaurant> GetRestaurantsByName(string name = null)

{

    return \_ctx.Restaurants

        .Where(r => string.IsNullOrEmpty(name) || r.Name.StartsWith(name))

        .OrderBy(r => r.Name);

}

# Detail Page

Opret en Razor Page kaldet **Detail** i Restaurant-folderen.

@page

@model WebApp.Pages.Restaurants.DetailModel

@{

    ViewData["Title"] = "Detail";

}

<h2>@Model.Restaurant.Name</h2>

<div>

    Id: @Model.Restaurant.Id

</div>

<div>

    Id: @Model.Restaurant.Location

</div>

<div>

    Id: @Model.Restaurant.Cuisine

</div>

<**a** **asp-page**="./List" class="btn btn-info">All Restaurants</**a**>

public class DetailModel : PageModel

{

    public Restaurant Restaurant { get; set; }

    private readonly IRestaurantService \_restaurantService;

    public DetailModel(IRestaurantService restaurantService)

    {

        \_restaurantService = restaurantService;

    }

    public void OnGet(int restaurantId)

    {

        Restaurant = \_restaurantService.GetRestaurantById(restaurantId);

    }

}

Tilføj GetRestarantById(int) til RestarantService:

public Restaurant GetRestaurantById(int restaurantId)

{

    return \_ctx.Restaurants.Find(restaurantId);

}

Tilføj ikon med link til List pagen:

<table class="table">

    @foreach (var restaurant in Model.Restaurants)

    {

    <tr>

        <td>@restaurant.Name</td>

        <td>@restaurant.Location</td>

        <td>@restaurant.Cuisine</td>

        <td>

            <**a** **asp-page**="./Detail" **asp-route-restaurantId**="@restaurant.Id">

                <i class="fas fa-info-circle"></i>

            </**a**>

        </td>

    </tr>

    }

</table>

Tilføjelse af Route i toppen af Details.cs:

@page "{restaurantId:int}"

Hvilket giver en URL: Restaurants/Detail/3 i stedet for Restaurants/Detail?restaurantId=3

# Håndtering af Bad Requests

Hvis brugeren ændrer URL’en til et ID, der ikke findes, så kastes en exception. Dette skal undgås.

Først oprettes en NotFound.cshtml page (uden PageModel):

@page

@{

    ViewData["Title"] = "NotFound";

}

<h2>Your restaurant was not found</h2>

<**a** **asp-page**="./List" class="btn btn-info">See All Restaurants</**a**>

Der laves en control i OnGet() og returntypen ændres til IActionResult:

public IActionResult OnGet(int restaurantId)

{

    Restaurant = \_restaurantService.GetRestaurantById(restaurantId);

    if (Restaurant == null)

    {

        return RedirectToPage("./NotFound");

    }

    return Page();

}