Create operation in Blazor

In this video we will implement Create operation in Blazor i.e add a new employee record to the Employees database table through a blazor web application.

Edit Employee Component Class (EditEmployeeBase.cs)

using AutoMapper;

using EmployeeManagement.Models;

using EmployeeManagement.Web.Models;

using EmployeeManagement.Web.Services;

using Microsoft.AspNetCore.Components;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace EmployeeManagement.Web.Pages

{

public class EditEmployeeBase : ComponentBase

{

[Inject]

public IEmployeeService EmployeeService { get; set; }

private Employee Employee { get; set; } = new Employee();

public EditEmployeeModel EditEmployeeModel { get; set; } =

new EditEmployeeModel();

[Inject]

public IDepartmentService DepartmentService { get; set; }

public List<Department> Departments { get; set; } =

new List<Department>();

[Parameter]

public string Id { get; set; }

[Inject]

public IMapper Mapper { get; set; }

[Inject]

public NavigationManager NavigationManager { get; set; }

protected async override Task OnInitializedAsync()

{

int.TryParse(Id, out int employeeId);

if (employeeId != 0)

{

Employee = await EmployeeService.GetEmployee(int.Parse(Id));

}

else

{

Employee = new Employee

{

DepartmentId = 1,

DateOfBrith = DateTime.Now,

PhotoPath = "images/nophoto.jpg"

};

}

Departments = (await DepartmentService.GetDepartments()).ToList();

Mapper.Map(Employee, EditEmployeeModel);

}

protected async Task HandleValidSubmit()

{

Mapper.Map(EditEmployeeModel, Employee);

Employee result = null;

if (Employee.EmployeeId != 0)

{

result = await EmployeeService.UpdateEmployee(Employee);

}

else

{

result = await EmployeeService.CreateEmployee(Employee);

}

if (result != null)

{

NavigationManager.NavigateTo("/");

}

}

}

}

Copy

Edit Employee Component View (EditEmployee.razor)

@page "/editemployee/{id}"

@page "/editemployee"

@inherits EditEmployeeBase

<EditForm Model="@EditEmployeeModel" OnValidSubmit="HandleValidSubmit">

<ObjectGraphDataAnnotationsValidator />

<h3>Edit Employee</h3>

<hr />

<ValidationSummary />

<div class="form-group row">

<label for="firstName" class="col-sm-2 col-form-label">

First Name

</label>

<div class="col-sm-10">

<InputText id="firstName" class="form-control" placeholder="First Name"

@bind-Value="EditEmployeeModel.FirstName" />

<ValidationMessage For="@(() => EditEmployeeModel.FirstName)" />

</div>

</div>

<div class="form-group row">

<label for="lastName" class="col-sm-2 col-form-label">

Last Name

</label>

<div class="col-sm-10">

<InputText id="lastName" class="form-control" placeholder="Last Name"

@bind-Value="EditEmployeeModel.LastName" />

<ValidationMessage For="@(() => EditEmployeeModel.LastName)" />

</div>

</div>

<div class="form-group row">

<label for="email" class="col-sm-2 col-form-label">

Email

</label>

<div class="col-sm-10">

<InputText id="email" class="form-control" placeholder="Email"

@bind-Value="EditEmployeeModel.Email" />

<ValidationMessage For="@(() => EditEmployeeModel.Email)" />

</div>

</div>

<div class="form-group row">

<label for="confirmEmail" class="col-sm-2 col-form-label">

Confirm Email

</label>

<div class="col-sm-10">

<InputText id="confirmEmail" class="form-control"

@bind-Value="EditEmployeeModel.ConfirmEmail" />

<ValidationMessage For="@(() => EditEmployeeModel.ConfirmEmail)" />

</div>

</div>

<div class="form-group row">

<label for="department" class="col-sm-2 col-form-label">

Department

</label>

<div class="col-sm-10">

<CustomInputSelect @bind-Value="EditEmployeeModel.DepartmentId"

class="form-control">

@foreach (var dept in Departments)

{

<option value="@dept.DepartmentId">

@dept.DepartmentName

</option>

}

</CustomInputSelect>

</div>

</div>

<div class="form-group row">

<label for="gender" class="col-sm-2 col-form-label">

Gender

</label>

<div class="col-sm-10">

<InputSelect @bind-Value="EditEmployeeModel.Gender" class="form-control">

@foreach (var gender in Enum.GetValues(typeof(Gender)))

{

<option value="@gender">@gender</option>

}

</InputSelect>

</div>

</div>

<div class="form-group row">

<label for="dateOfBirth" class="col-sm-2 col-form-label">

Date Of Birth

</label>

<div class="col-sm-10">

<InputDate @bind-Value="EditEmployeeModel.DateOfBrith" class="form-control" />

</div>

</div>

@\*<div class="form-group row">

<label for="deptName" class="col-sm-2 col-form-label">

Department Name

</label>

<div id="deptName" class="col-sm-10">

<InputText @bind-Value="EditEmployeeModel.Department.DepartmentName"

class="form-control" />

<ValidationMessage For="@(() => EditEmployeeModel.Department.DepartmentName)" />

</div>

</div>\*@

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

</EditForm>

Copy

NavMenu.razor

<li class="nav-item px-3">

<NavLink class="nav-link" href="/editemployee">

<span class="oi oi-file" aria-hidden="true"></span> Create

</NavLink>

</li>

Copy

IEmployeeService

public interface IEmployeeService

{

Task<IEnumerable<Employee>> GetEmployees();

Task<Employee> GetEmployee(int id);

Task<Employee> UpdateEmployee(Employee updatedEmployee);

Task<Employee> CreateEmployee(Employee newEmployee);

}

Copy

EmployeeService

public class EmployeeService : IEmployeeService

{

private readonly HttpClient httpClient;

public EmployeeService(HttpClient httpClient)

{

this.httpClient = httpClient;

}

public async Task<Employee> CreateEmployee(Employee newEmployee)

{

return await httpClient.PostJsonAsync<Employee>("api/employees", newEmployee);

}

}

Copy

nophoto.jpg used in the demo

