Bahria University,

Karachi Campus

A picture containing text, emblem, symbol, crown

Description automatically generated

LAB EXPERIMENT NO.

\_\_\_13\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | Employee API with crud |
| 2 | Create a payroll API and perform crud operations on it |

Submitted On:

20-06-2023

(Date: DD/MM/YY)

**LAB 13**

**Task: 1*:* Employee API with crud:**

**Employee controller:**

namespace MVCwithWebAPI.Controllers

{

public class EmployeesController : Controller

{

readonly string apiBaseAddress = ConfigurationManager.AppSettings["apiBaseAddress"];

public async Task<ActionResult> Index()

{

IEnumerable<Employee> employees = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var result = await client.GetAsync("Employees/get");

if (result.IsSuccessStatusCode)

{

employees = await result.Content.ReadAsAsync<IList<Employee>>();

}

else

{

employees = Enumerable.Empty<Employee>();

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

return View(employees);

}

public async Task<ActionResult> Details(string id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Employee employee = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var result = await client.GetAsync($"Employees/details/{id}");

if (result.IsSuccessStatusCode)

{

employee = await result.Content.ReadAsAsync<Employee>();

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Create([Bind(Include = "Id,Name,Address,Gender,Company,Designation")] Employee employee)

{

if (ModelState.IsValid)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var response = await client.PostAsJsonAsync("Employees/Create", employee);

if (response.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

}

return View(employee);

}

public async Task<ActionResult> Edit(string id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Employee employee = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var result = await client.GetAsync($"Employees/details/{id}");

if (result.IsSuccessStatusCode)

{

employee = await result.Content.ReadAsAsync<Employee>();

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Edit([Bind(Include = "Id,Name,Address,Gender,Company,Designation")] Employee employee)

{

if (ModelState.IsValid)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var response = await client.PutAsJsonAsync("Employees/edit", employee);

if (response.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

return RedirectToAction("Index");

}

return View(employee);

}

public async Task<ActionResult> Delete(string id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Employee employee = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var result = await client.GetAsync($"Employees/details/{id}");

if (result.IsSuccessStatusCode)

{

employee = await result.Content.ReadAsAsync<Employee>();

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<ActionResult> DeleteConfirmed(string id)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var response = await client.DeleteAsync($"Employees/delete/{id}");

if (response.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

else

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

return View();

}

}

}

**Employee api:**

namespace MVCwithWebAPI.Controllers

{

public class EmployeesApiController : ApiController

{

private readonly IEmployeeRepository \_iEmployeeRepository = new EmployeeRepository();

[HttpGet]

[Route("api/Employees/Get")]

public async Task<IEnumerable<Employee>> Get()

{

return await \_iEmployeeRepository.GetEmployees();

}

[HttpPost]

[Route("api/Employees/Create")]

public async Task CreateAsync([FromBody] Employee employee)

{

if (ModelState.IsValid)

{

await \_iEmployeeRepository.Add(employee);

}

}

[HttpGet]

[Route("api/Employees/Details/{id}")]

public async Task<Employee> Details(string id)

{

var result = await \_iEmployeeRepository.GetEmployee(id);

return result;

}

[HttpPut]

[Route("api/Employees/Edit")]

public async Task EditAsync([FromBody] Employee employee)

{

if (ModelState.IsValid)

{

await \_iEmployeeRepository.Update(employee);

}

}

[HttpDelete]

[Route("api/Employees/Delete/{id}")]

public async Task DeleteConfirmedAsync(string id)

{

await \_iEmployeeRepository.Delete(id);

}

}

}

**Sqldbcontext:**

namespace MVCwithWebAPI

{

public class SqlDbContext

: DbContext

{

public SqlDbContext() : base("name=SqlConn")

{

}

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

}

}

**Create.cshtml:**

@model MVCwithWebAPI.Models.Employee

@{

ViewBag.Title = "Create Employee";

}

<h3>Create Employee</h3>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })

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

@Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Address, htmlAttributes: new { @class = "control-label col-md-2" })

</div>

<div class="form-group">

@Html.LabelFor(model => model.Designation, htmlAttributes: new { @class = "control-label col-md-2" })

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

@Html.EditorFor(model => model.Designation, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Designation, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Create" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

**Delete:**

@model MVCwithWebAPI.Models.Employee

@{

ViewBag.Title = "Delete";

}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>Employee</h4>

<hr />

<dl class="dl-horizontal">

<dt>

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd>

@Html.DisplayFor(model => model.Name)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Address)

</dt>

<dd>

@Html.DisplayFor(model => model.Address)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Gender)

</dt>

<dd>

@Html.DisplayFor(model => model.Gender)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Company)

</dt>

<dd>

@Html.DisplayFor(model => model.Company)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Designation)

</dt>

<dd>

@Html.DisplayFor(model => model.Designation)

</dd>

</dl>

@using (Html.BeginForm()) {

@Html.AntiForgeryToken()

<div class="form-actions no-color">

<input type="submit" value="Delete" class="btn btn-default" /> |

@Html.ActionLink("Back to List", "Index")

</div>

}

</div>

**Database Code:**

USE [task1]

GO

CREATE TABLE [dbo].[Employees](

[Id] [nvarchar](50) NOT NULL,

[Name] [nvarchar](50) NULL,

[Address] [nvarchar](50) NULL,

[Gender] [nvarchar](10) NULL,

[Company] [nvarchar](50) NULL,

[Designation] [nvarchar](50) NULL,

CONSTRAINT [PK\_Employees] PRIMARY KEY CLUSTERED

(

[Id] ASC

)

)

GO

IEmployeerepository:

using MVCwithWebAPI.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MVCwithWebAPI

{

public interface IEmployeeRepository

{

Task Add(Employee employee);

Task Update(Employee employee);

Task Delete(string id);

Task<Employee> GetEmployee(string id);

Task<IEnumerable<Employee>> GetEmployees();

}

}

Output:

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing text, font, line, screenshot

Description automatically generated

**Task: 2:** Create a payroll API and perform crud operations on it.

namespace WebApplication4.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class PayrollController : ControllerBase

{

private static readonly List<Payroll> Payrolls = new List<Payroll>

{

new Payroll { Id = 1, EmployeeId = 1, Salary = 5000, Date = DateTime.Now },

new Payroll { Id = 2, EmployeeId = 2, Salary = 4000, Date = DateTime.Now }

};

[HttpGet]

public IActionResult Get()

{

return Ok(Payrolls);

}

[HttpGet("{id}")]

public IActionResult Get(int id)

{

var payroll = Payrolls.Find(p => p.Id == id);

if (payroll == null)

{

return NotFound();

}

return Ok(payroll);

}

[HttpPost]

public ActionResult<Payroll> Post(Payroll payroll)

{

payroll.Id = GeneratePayrollId();

Payrolls.Add(payroll);

return CreatedAtAction(nameof(Get), new { id = payroll.Id }, payroll);

}

[HttpPut("{id}")]

public IActionResult Put(int id, Payroll updatedPayroll)

{

var payroll = Payrolls.Find(p => p.Id == id);

if (payroll == null)

{

return NotFound();

}

payroll.EmployeeId = updatedPayroll.EmployeeId;

payroll.Salary = updatedPayroll.Salary;

payroll.Date = updatedPayroll.Date;

return NoContent();

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

var payroll = Payrolls.Find(p => p.Id == id);

if (payroll == null)

{

return NotFound();

}

Payrolls.Remove(payroll);

return NoContent();

}

private int GeneratePayrollId()

{

// Logic to generate a unique payroll id

return Payrolls.Count + 1;

}

}

A screenshot of a computer

Description automatically generated with medium confidence}

A screenshot of a computer

Description automatically generated with low confidence