**4. WebApi\_Handson**

**Objectives:**

* Demonstrate creation of an Action method to perform data create, update & delete operation
  + Use FromBody attribute, extract data to custom model class using FromBody attribute, use hardcoded data to update & delete data, Use Swagger and POSTMAN to test

1. **Web Api CRUD operation**

Update Employee data as per the input thru Web API PUT action method call

Employee information has to be updated based on the user input. Use Swagger tool to invoke the action method mapped with Http PUT action verb to update an employee data.

Modify the action method to return Employee data thru ActionResult.

Check if the id value is lesser than or equal to 0. If true, throw BadRequest action result with the message ‘Invalid employee id’

If the value is greater than 0 but not available in the list of employee ids that is there in the hardcoded list of employees, throw BadRequest action result with the same message as stated above.

If the id value is valid, use the JSON data from the input body and update the hardcoded list. Filter the employee list data for the input id and return that as the output

Program:

using EmployeeApiDemo.Filters;

using EmployeeApiDemo.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

namespace EmployeeApiDemo.Controllers

{

[ApiController]

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

[CustomAuthFilter]

public class EmployeeController : ControllerBase

{

// Step 1: Static hardcoded employee list to simulate a database

private static List<Employee> employees = new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 50000,

Permanent = true,

DateOfBirth = new DateTime(1995, 01, 01),

Department = new Department { Id = 1, Name = "IT" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "SQL" }

}

},

new Employee

{

Id = 2,

Name = "Bob",

Salary = 60000,

Permanent = false,

DateOfBirth = new DateTime(1990, 05, 10),

Department = new Department { Id = 2, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 3, Name = "Communication" },

new Skill { Id = 4, Name = "Recruitment" }

}

}

};

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

[AllowAnonymous]

public ActionResult<List<Employee>> GetStandard()

{

// Uncomment below to simulate exception

//throw new Exception("Simulated exception for testing");

return Ok(employees);

} [HttpPost]

public IActionResult CreateEmployee([FromBody] Employee emp)

{

return Ok($"Employee {emp.Name} created successfully!");

}

[HttpPut]

public ActionResult<Employee> UpdateEmployee([FromBody] Employee updatedEmp)

{

if (updatedEmp == null || updatedEmp.Id <= 0)

{

return BadRequest("Invalid employee id");

}

var existingEmp = employees.FirstOrDefault(e => e.Id == updatedEmp.Id);

if (existingEmp == null)

{

return BadRequest("Invalid employee id");

}

existingEmp.Name = updatedEmp.Name;

existingEmp.Salary = updatedEmp.Salary;

existingEmp.Permanent = updatedEmp.Permanent;

existingEmp.DateOfBirth = updatedEmp.DateOfBirth;

existingEmp.Department = updatedEmp.Department;

existingEmp.Skills = updatedEmp.Skills;

return Ok(existingEmp);

}

}

}

OUTPUT:

