**Handson 5: Web API**

Superset id: 6363535

## **What is CORS?**

**CORS** stands for **Cross-Origin Resource Sharing**.

By default, browsers block JavaScript requests from a frontend app (e.g., http://localhost:3000) to a different origin backend API (e.g., https://localhost:5001) for security reasons.

CORS is a browser security feature that **controls which domains are allowed to access resources** from your Web API.

**Controller-**

**EmployeeController.cs**

﻿using Microsoft.AspNetCore.Mvc;

using WebApiLab3.Models;

namespace WebApiLab3.Controllers

{

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

[ApiController]

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees = new List<Employee>

{

new Employee

{

Id = 1,

Name = "Anisha",

Salary = 50000,

Permanent = true,

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

Skills = new List<Skill>

{

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

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

},

DateOfBirth = new DateTime(1999, 5, 12)

},

new Employee

{

Id = 2,

Name = "Ravi",

Salary = 45000,

Permanent = false,

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

Skills = new List<Skill>

{

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

},

DateOfBirth = new DateTime(1998, 11, 20)

}

};

// GET: api/Employee

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), StatusCodes.Status200OK)]

public ActionResult<List<Employee>> Get()

{

return Ok(\_employees);

}

// GET: api/Employee/1

[HttpGet("{id}")]

[ProducesResponseType(typeof(Employee), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public ActionResult<Employee> Get(int id)

{

var employee = \_employees.FirstOrDefault(e => e.Id == id);

if (employee == null)

return NotFound();

return Ok(employee);

}

// POST: api/Employee

[HttpPost]

[ProducesResponseType(typeof(Employee), StatusCodes.Status201Created)]

public ActionResult<Employee> Post([FromBody] Employee emp)

{

emp.Id = \_employees.Max(e => e.Id) + 1;

\_employees.Add(emp);

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

}

// PUT: api/Employee/1

[HttpPut("{id}")]

[ProducesResponseType(typeof(Employee), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public ActionResult<Employee> Put(int id, [FromBody] Employee updatedEmp)

{

if (id <= 0)

return BadRequest("Invalid employee id");

var emp = \_employees.FirstOrDefault(e => e.Id == id);

if (emp == null)

return BadRequest("Invalid employee id");

emp.Name = updatedEmp.Name;

emp.Salary = updatedEmp.Salary;

emp.Permanent = updatedEmp.Permanent;

emp.Department = updatedEmp.Department;

emp.Skills = updatedEmp.Skills;

emp.DateOfBirth = updatedEmp.DateOfBirth;

return Ok(emp);

}

// DELETE: api/Employee/1

[HttpDelete("{id}")]

[ProducesResponseType(StatusCodes.Status204NoContent)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public IActionResult Delete(int id)

{

var emp = \_employees.FirstOrDefault(e => e.Id == id);

if (emp == null)

return NotFound();

\_employees.Remove(emp);

return NoContent();

}

}

}

**Filter-**

**CustomAuthFilter.cs**

**﻿**using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebApiLab3.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var headers = context.HttpContext.Request.Headers;

if (!headers.ContainsKey("Authorization"))

{

context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

return;

}

var token = headers["Authorization"].ToString();

if (!token.Contains("Bearer"))

{

context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

return;

}

base.OnActionExecuting(context);

}

}

}

[**CustomExceptionFilter.cs**](http://customexceptionfilter.cs)

﻿using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System;

using System.IO;

namespace WebApiLab3.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

var logPath = "ExceptionLog.txt";

var logMessage = $"[{DateTime.Now}] Exception: {exception.Message}\nStackTrace: {exception.StackTrace}\n\n";

File.AppendAllText(logPath, logMessage);

context.Result = new ObjectResult("Something went wrong. Please contact support.")

{

StatusCode = 500

};

context.ExceptionHandled = true;

}

}

}

**Models-**

[**Department.cs**](http://department.cs)

namespace EmployeeApiDemo.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Skill**[**.cs**](http://department.cs)

namespace EmployeeApiDemo.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Employee.cs**

namespace EmployeeApiDemo.Models

{

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public int Salary { get; set; }

public bool Permanent { get; set; }

public Department Department { get; set; }

public List<Skill> Skills { get; set; }

public DateTime DateOfBirth { get; set; }

}

}

