**ASP.NET Core 8.0 WEB API HANDSON – WEEK 4**

**HANDS ON 1**

**Code :**

**Controller.cs**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace WebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class ValuesController : ControllerBase

{

private static List<string> values = new List<string> { "value1", "value2" };

[HttpGet]

public IActionResult GetAll()

{

return Ok(values);

}

[HttpGet("{id}")]

public IActionResult GetById(int id)

{

if (id < 0 || id >= values.Count)

return NotFound();

return Ok(values[id]);

}

[HttpPost]

public IActionResult Create([FromBody] string value)

{

values.Add(value);

return Ok(value);

[HttpPut("{id}")]

public IActionResult Update(int id, [FromBody] string newValue)

{

if (id < 0 || id >= values.Count)

return BadRequest("Invalid ID");

values[id] = newValue;

return Ok(newValue);

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

if (id < 0 || id >= values.Count)

return BadRequest("Invalid ID");

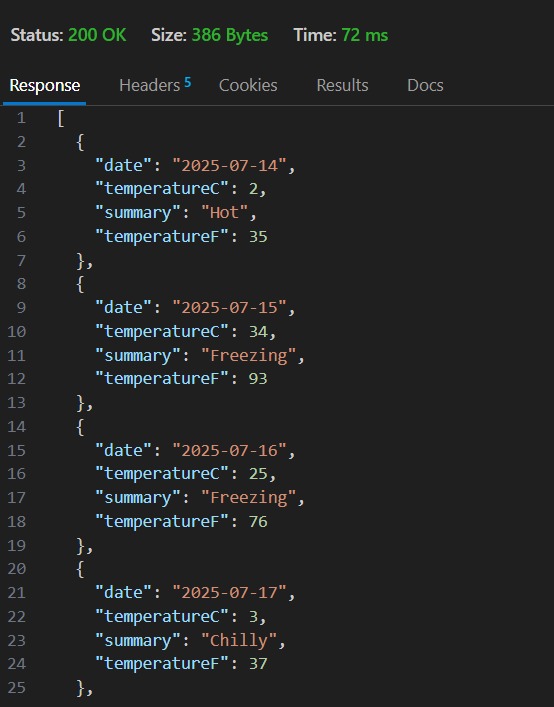
values.RemoveAt(id);

return Ok();

       }

    }

}

****

**program.cs**

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddOpenApi();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.MapOpenApi();

}

app.UseHttpsRedirection();

var summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

app.MapGet("/weatherforecast", () =>

{

var forecast = Enumerable.Range(1, 5).Select(index =>

new WeatherForecast

(

DateOnly.FromDateTime(DateTime.Now.AddDays(index)),

Random.Shared.Next(-20, 55),

summaries[Random.Shared.Next(summaries.Length)]

))

.ToArray();

return forecast;

})

.WithName("GetWeatherForecast");

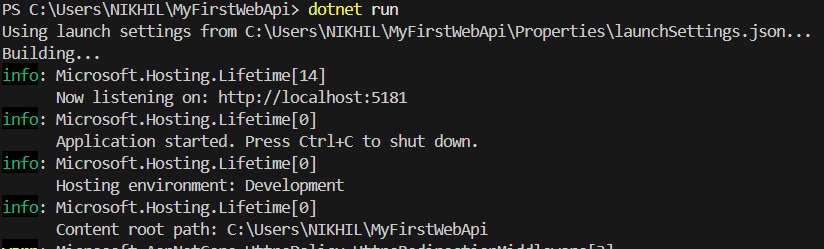
app.Run();

record WeatherForecast(DateOnly Date, int TemperatureC, string? Summary)

{

public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);

}



**HANDS ON 2**

**Program.cs**

using EmployeeApiDemo.Models;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Employee API",

Version = "v1",

Description = "CRUD API for Employee",

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

Url = new Uri("http://example.com")

}

});

});

var app = builder.Build();

// Configure middleware

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Employee API v1");

});

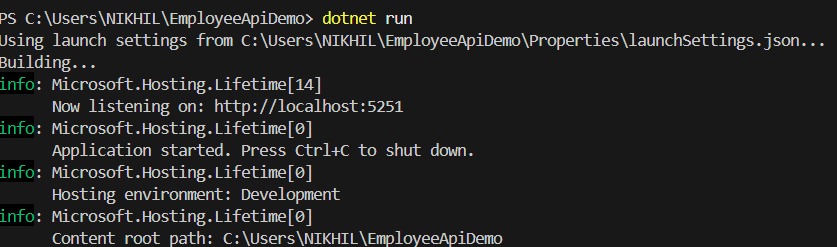
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();



**employee.cs**

namespace EmployeeApiDemo.Models

{

public class Employee

{

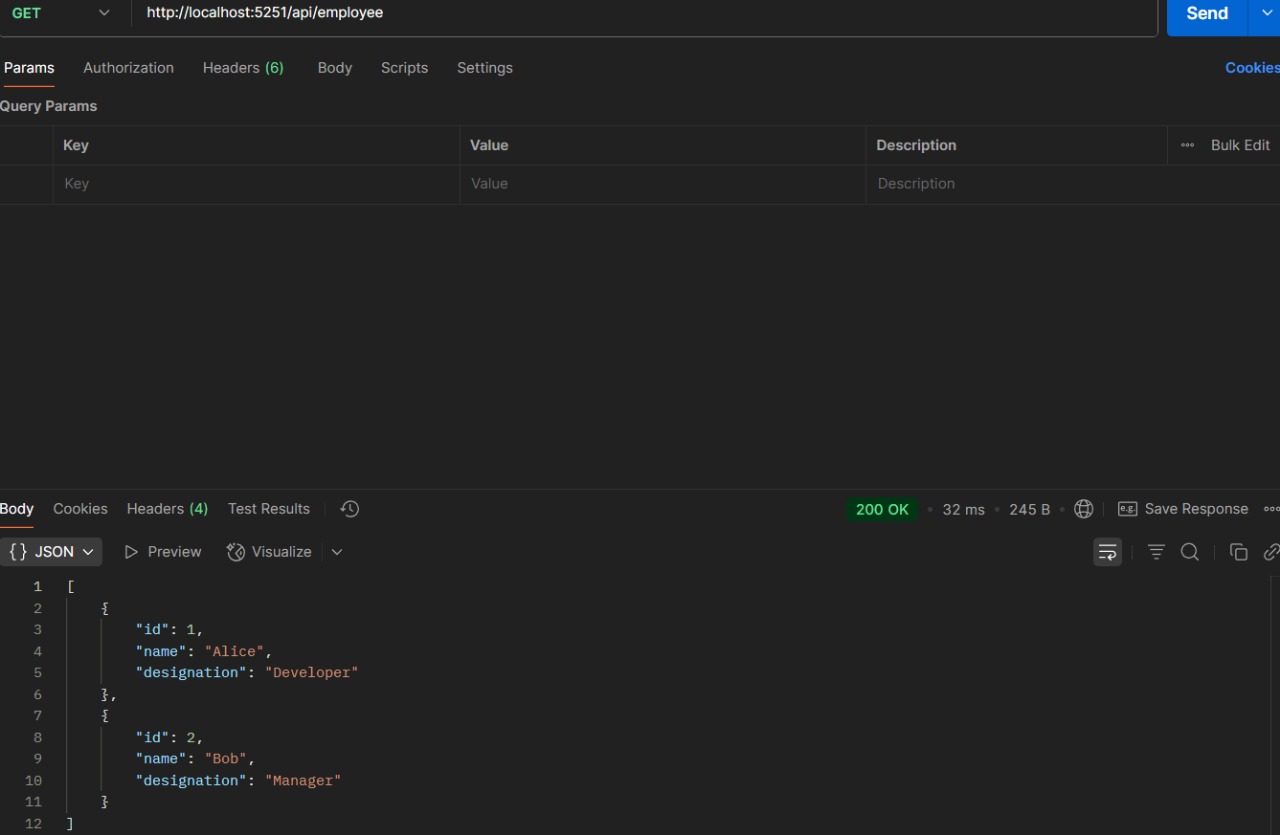
public int Id { get; set; }

public string Name { get; set; } = string.Empty;

public string Designation { get; set; } = string.Empty;

}

}



**employee controller.cs**

using Microsoft.AspNetCore.Mvc;

using EmployeeApiDemo.Models;

namespace EmployeeApiDemo.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static List<Employee> employees = new()

{

new Employee { Id = 1, Name = "Alice", Designation = "Developer" },

new Employee { Id = 2, Name = "Bob", Designation = "Manager" }

};

[HttpGet]

public ActionResult<IEnumerable<Employee>> Get() => employees;

[HttpGet("{id}")]

public ActionResult<Employee> Get(int id)

{

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

return emp is null ? NotFound() : Ok(emp);

}

[HttpPost]

public ActionResult<Employee> Post(Employee emp)

{

employees.Add(emp);

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

}

[HttpPut("{id}")]

public IActionResult Put(int id, Employee updatedEmp)

{

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

if (emp is null) return NotFound();

emp.Name = updatedEmp.Name;

emp.Designation = updatedEmp.Designation;

return Ok(emp);

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

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

if (emp is null) return NotFound();

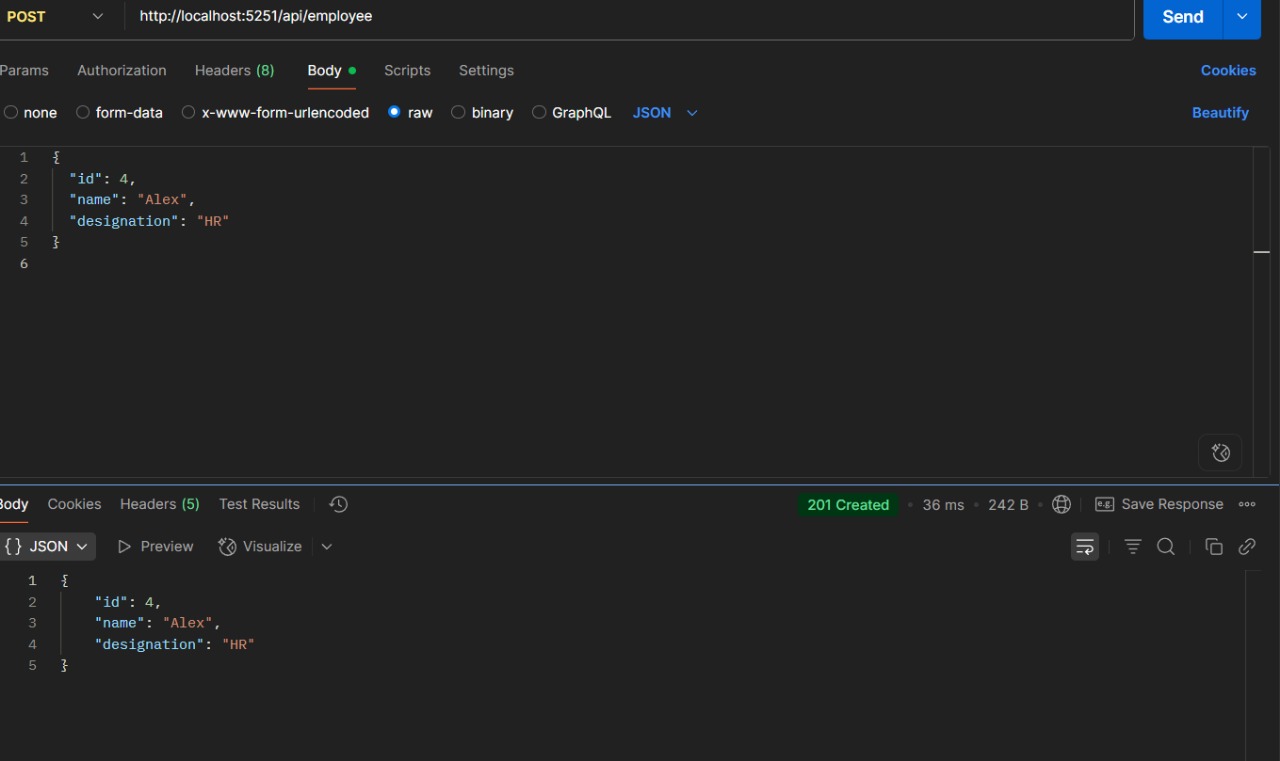
employees.Remove(emp);

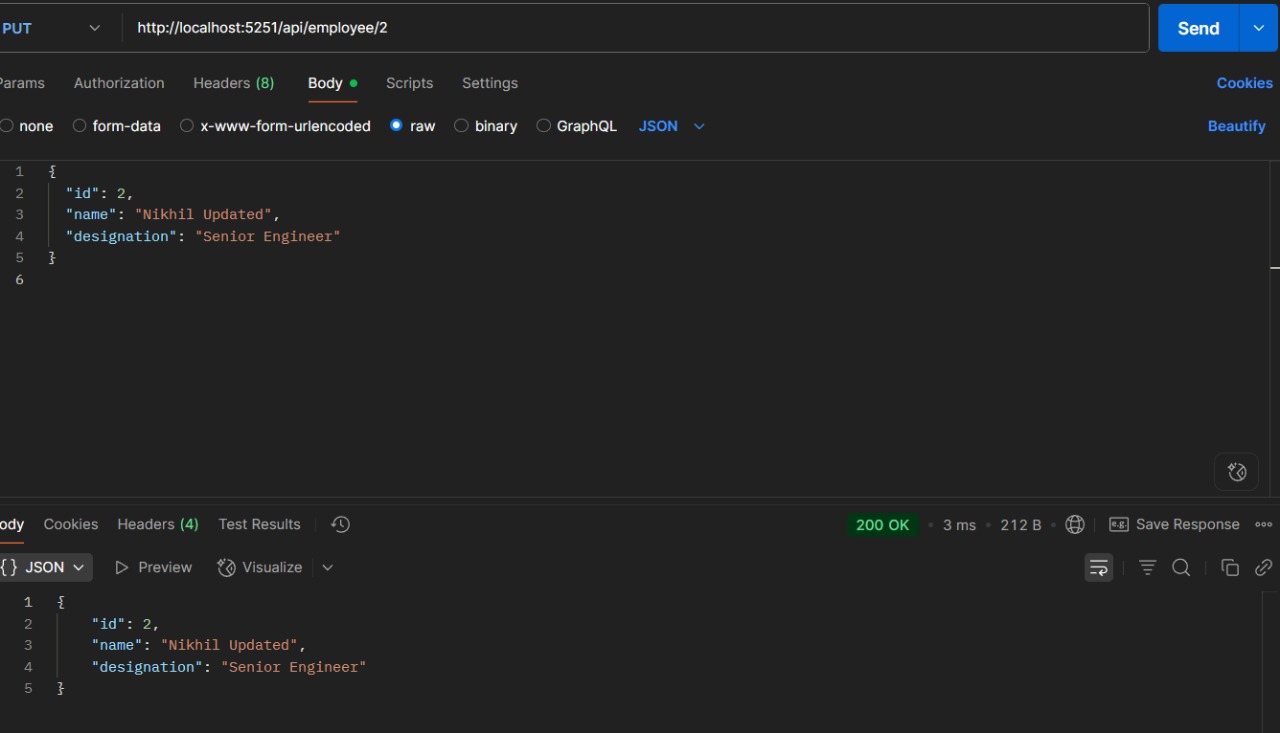
return NoContent();

       }

    }

}





**HANDS ON 3**

**program.cs**

using EmployeeWebApi.Filters;

using EmployeeWebApi.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.OpenApi.Models;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers(options =>

{

options.Filters.Add(typeof(CustomExceptionFilter)); // Custom exception filter

});

builder.Services.AddScoped<CustomAuthFilter>(); // Register your custom auth filter

builder.Services.AddSingleton<EmployeeRepository>();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "Employee API", Version = "v1" });

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter 'Bearer' followed by space and your token. Example: Bearer token123"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<string>()

}

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

appUseAuthorization();

app.MapControllers();

app.Run();

**custom.auth.filter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeWebApi.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

if (!context.HttpContext.Request.Headers.ContainsKey("Auth"))

{

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

return;

}

var token = context.HttpContext.Request.Headers["Auth"].ToString();

if (string.IsNullOrEmpty(token) || !token.Contains("pass123"))

{

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

return;

}

base.OnActionExecuting(context);

       }

    }

}

**customexceptionfilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

namespace EmployeeWebApi.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

string path = Path.Combine(Directory.GetCurrentDirectory(), "logs.txt");

File.AppendAllText(path, $"[{DateTime.Now}] {context.Exception.Message}\n");

context.Result = new ObjectResult("An internal error occurred.")

{

StatusCode = 500

};

       }

    }

}

**controller.cs**

using EmployeeWebApi.Filters;

using EmployeeWebApi.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Authorization;

namespace EmployeeWebApi.Controllers

{

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

[ApiController]

[ServiceFilter(typeof(CustomAuthFilter))] // This triggers our custom filter

public class EmployeeController : ControllerBase

{

private readonly List<Employee> \_employees

public EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

[HttpGet]

[AllowAnonymous]

[ProducesResponseType(typeof(List<Employee>), 200)]

[ProducesResponseType(500)]

public ActionResult<List<Employee>> GetStandard()

{

return \_employees;

}

[HttpPost]

public IActionResult Post([FromBody] Employee employee)

{

\_employees.Add(employee);

return Ok(employee);

}

[HttpPut]

public IActionResult Put([FromBody] Employee employee)

{

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

if (existing != null)

{

existing.Name = employee.Name;

existing.Salary = employee.Salary;

existing.Permanent = employee.Permanent;

existing.Department = employee.Department;

existing.Skills = employee.Skills;

existing.DateOfBirth = employee.DateOfBirth;

}

return Ok(existing);

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

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

},

DateOfBirth = new DateTime(1990, 1, 1)

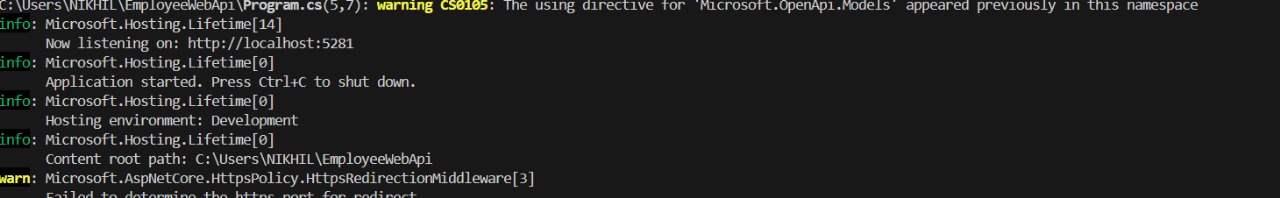
}

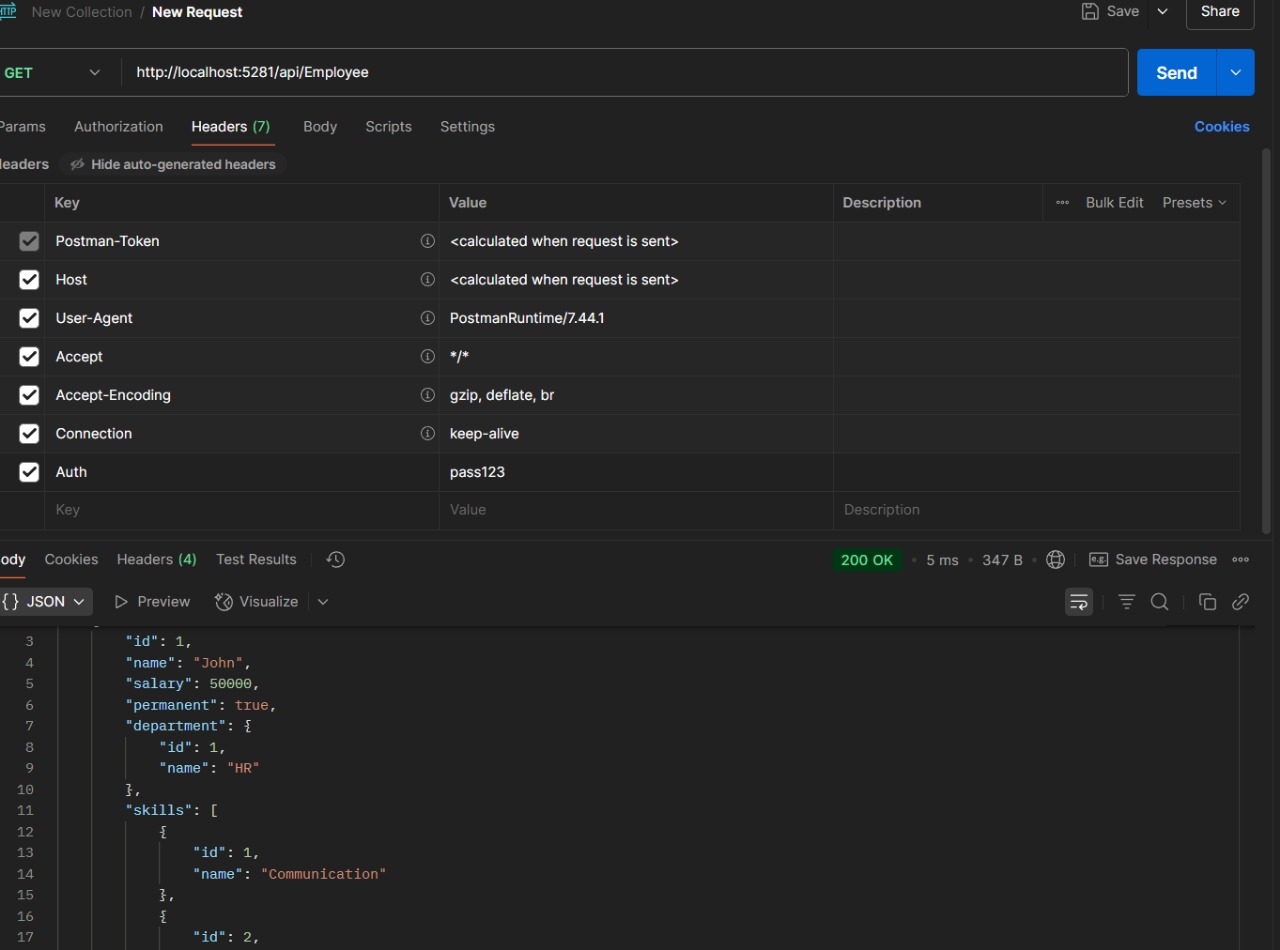
};

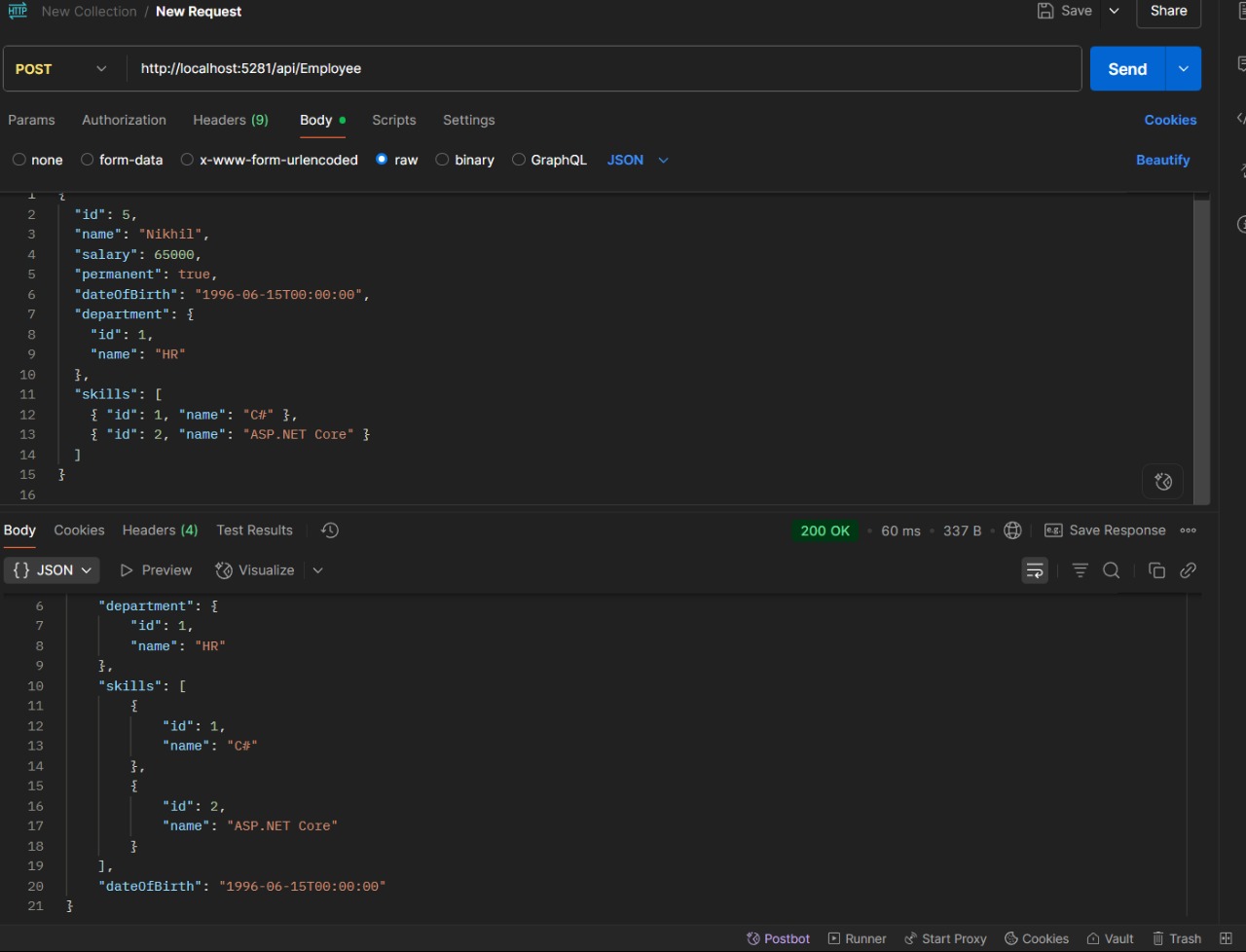
       }

    }

}







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

if (employee == null)

return NotFound("Employee not found");

employees.Remove(employee);

return NoContent(); // 204 **HANDS ON 4**

**Program.cs**

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**controller.py**

using Microsoft.AspNetCore.Mvc;

using EmployeeWebApi.Models;

namespace EmployeeWebApi.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

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

{

new Employee { Id = 1, Name = "Alice", Department = "HR", Salary = 50000 },

new Employee { Id = 2, Name = "Bob", Department = "IT", Salary = 60000 },

new Employee { Id = 3, Name = "Charlie", Department = "Finance", Salary = 55000 }

};

// GET: api/employee

[HttpGet]

public ActionResult<IEnumerable<Employee>> GetAllEmployees()

{

return Ok(employees);

}

// GET: api/employee/2

[HttpGet("{id}")]

public ActionResult<Employee> GetEmployee(int id)

{

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

if (employee == null)

return NotFound("Employee not found");

return Ok(employee);

}

[HttpPost]

public ActionResult<Employee> CreateEmployee([FromBody] Employee newEmployee)

{

if (newEmployee.Id <= 0)

return BadRequest("Invalid employee id");

if (employees.Any(e => e.Id == newEmployee.Id))

return BadRequest("Employee with this id already exists");

employees.Add(newEmployee);

return CreatedAtAction(nameof(GetEmployee), new { id = newEmployee.Id }, newEmployee);

}

// PUT: api/employee/2

[HttpPut("{id}")]

public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updatedEmployee)

{

if (id <= 0)

return BadRequest("Invalid employee id");

if (id != updatedEmployee.Id)

return BadRequest("Employee id in URL and body do not match");

var existingEmployee = employees.FirstOrDefault(e => e.Id == id);

if (existingEmployee == null)

return BadRequest("Invalid employee id");

existing mployee.Name = updatedEmployee.Name;

existingEmployee.Department = updatedEmployee.Department;

existingEmployee.Salary = updatedEmployee.Salary;

return Ok(existingEmployee);

}

[HttpDelete("{id}")]

public ActionResult DeleteEmployee(int id)

{

       }

    }

}

**employee.cs**

namespace EmployeeWebApi.Models

{

public class Employee

{

public int Id { get; set; }

public string? Name { get; set; }

public string? Department { get; set; }

public double Salary { get; set; }

    }

}

