**Handson 4: Web API**

Superset id: 6363535

[**Program.cs**](http://program.cs)

using EmployeeApiDemo.Filters;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Enable Swagger with Authorization support

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

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

// Add Bearer token support

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter 'Bearer' followed by your token. Example: Bearer mytoken123"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<string>()

}

});

});

// Register your custom filter

builder.Services.AddScoped<CustomAuthFilter>();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using EmployeeApiDemo.Models;

using EmployeeApiDemo.Filters;

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

[ApiController]

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private List<Employee> employees;

public EmployeeController()

{

employees = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

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(1990, 1, 1)

}

};

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> Get()

{

return Ok(employees);

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

employees.Add(emp);

return Ok(emp);

}

[HttpPut("{id}")]

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

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

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

if (existingEmployee == null)

{

return BadRequest("Invalid employee id");

}

existingEmployee.Name = updatedEmployee.Name;

existingEmployee.Salary = updatedEmployee.Salary;

existingEmployee.Permanent = updatedEmployee.Permanent;

existingEmployee.Department = updatedEmployee.Department;

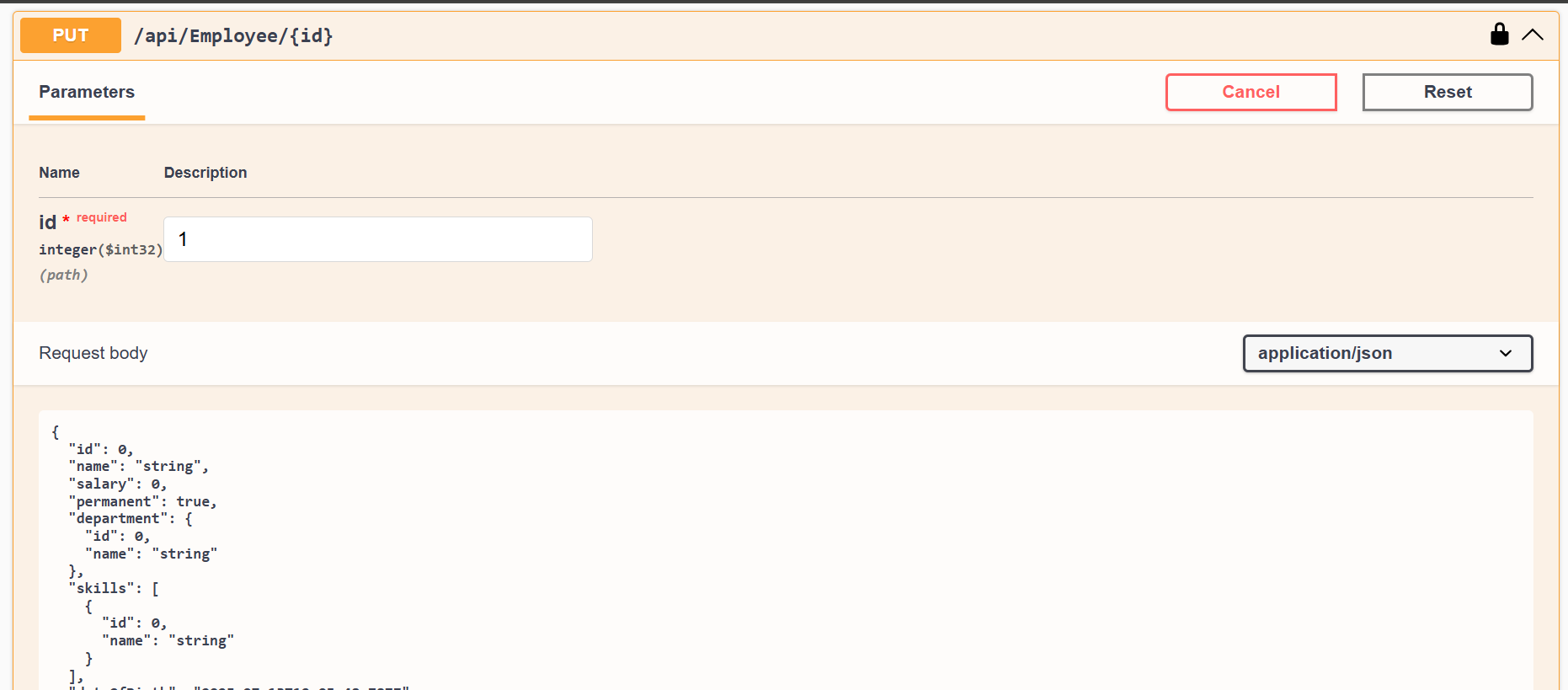
existingEmployee.Skills = updatedEmployee.Skills;

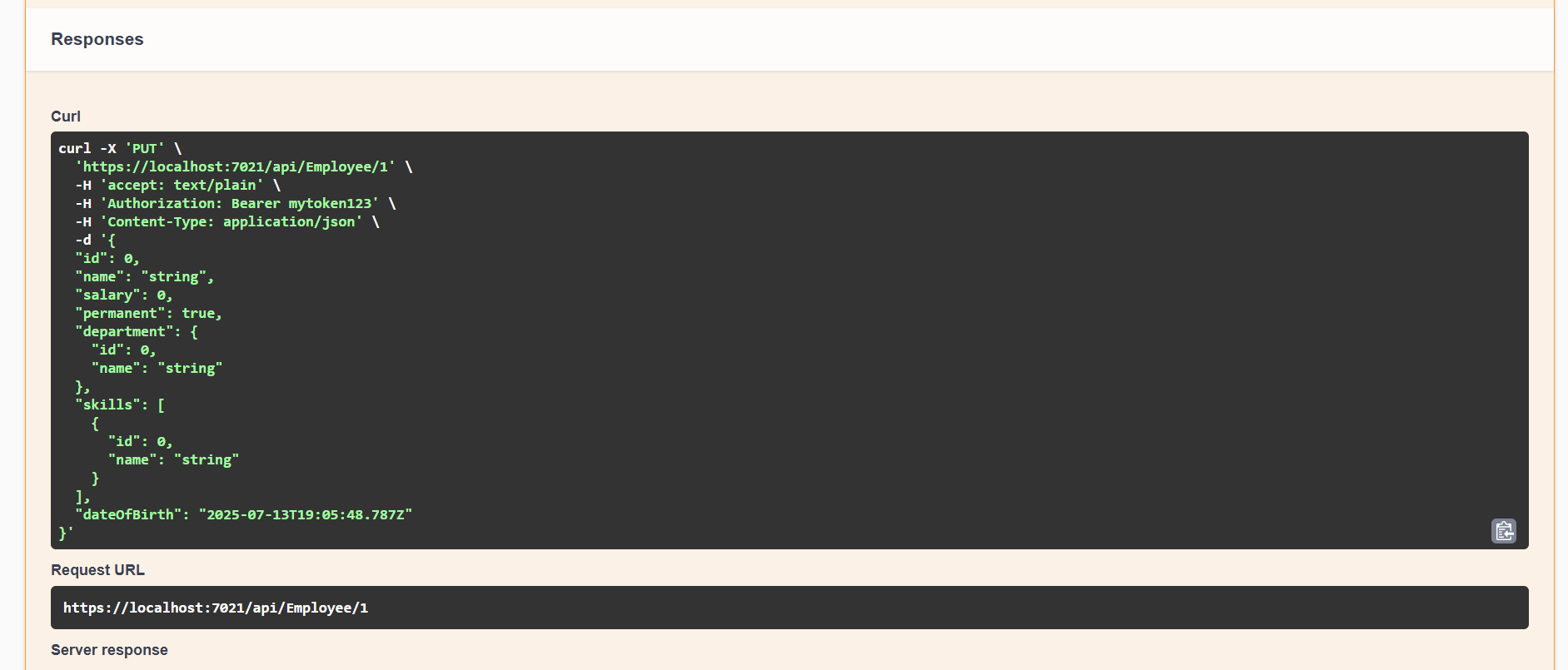
existingEmployee.DateOfBirth = updatedEmployee.DateOfBirth;

return Ok(existingEmployee);

}

}







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

