Search in ASP.NET Core REST API

In this video we will discuss, **how to implement search feature in ASP.NET Core REST API**.

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

[ApiController]

public class EmployeesController : ControllerBase

{

private readonly IEmployeeRepository employeeRepository;

public EmployeesController(IEmployeeRepository employeeRepository)

{

this.employeeRepository = employeeRepository;

}

[HttpGet("{search}")]

public async Task<ActionResult<IEnumerable<Employee>>> Search(string name, Gender? gender)

{

try

{

var result = await employeeRepository.Search(name, gender);

if (result.Any())

{

return Ok(result);

}

return NotFound();

}

catch (Exception)

{

return StatusCode(StatusCodes.Status500InternalServerError,

"Error retrieving data from the database");

}

}

}

Copy

Code Explanation

As per the [Route] attribute on the EmployeesController, the route to reach this controller is /api/employees

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

[ApiController]

public class EmployeesController : ControllerBase

{

}

Copy

The following [HttpGet] attribute specified an extension to the base route /api/employees. So the route to reach this Search() method is /api/employees/search. The values for the two method parameters come from the query strings in the URL.

[HttpGet("{search}")]

public async Task<ActionResult<IEnumerable<Employee>>> Search(string name, Gender? gender)

{

}

Copy

If you want the search terms (name and gender) to be included as route parameters instead of query strings, change the route template on the HttpGet attribute as shown below. However, this approach is not recommended if you have many search parameters.

[HttpGet("{search}/{name}/{gender?}")]

public async Task<ActionResult<IEnumerable<Employee>>> Search(string name, Gender? gender)

{

}

Copy

Since the route parameters and method parameters are mapped by name, even if the order does not match, they will still be correctly mapped i.e the URL parameter name is mapped to the method parameter name and the gender URL parameter is mapped to the gender method parameter.

[HttpGet("{search}/{name}/{gender?}")]

public async Task<ActionResult<IEnumerable<Employee>>> Search(Gender? gender, string name)

{

}

Copy

IEmployeeRepository Interface

public interface IEmployeeRepository

{

Task<IEnumerable<Employee>> Search(string name, Gender? gender);

}

Copy

EmployeeRepository

public class EmployeeRepository : IEmployeeRepository

{

private readonly AppDbContext appDbContext;

public EmployeeRepository(AppDbContext appDbContext)

{

this.appDbContext = appDbContext;

}

public async Task<IEnumerable<Employee>> Search(string name, Gender? gender)

{

IQueryable<Employee> query = appDbContext.Employees;

if (!string.IsNullOrEmpty(name))

{

query = query.Where(e => e.FirstName.Contains(name)

|| e.LastName.Contains(name));

}

if(gender != null)

{

query = query.Where(e => e.Gender == gender);

}

return await query.ToListAsync();

}

}