Ryan Anderson
Documentation for Mindex Code Challenge
7-10-2022

Table Of Contents

Overview:

This documentation will review the changes that were made to the code challenge ASP.Net Core project **sr-code-challenge-dotnet**. This project hosts an employee database which contains various employee information that can be returned via API calls. The database is virtual and is persistent only during the duration of the project running. Once the project is not running, the database is reset to its original state.

This project was submitted to me with two tasks:

- *Task 1: Create an endpoint which returns how many employees work under a specific employee*
- *Task 2: Create two endpoints which creates and also separately returns data involving how much an employee makes (salary) and its effective date.*

Task 1

Requirements

Create a new type, ReportingStructure, that has two properties: employee and numberOfReports. For the field "numberOfReports", this should equal the total number of reports under a given employee. The number of reports is determined to be the number of directReports for an employee and all of their direct reports. This new type should have a new REST endpoint created for it. This new endpoint should accept an employeeId and return the fully filled out ReportingStructure for the specified employeeId. The values should be computed on the fly and will not be persisted.

General Overview and Syntax

This endpoint receives an employeeID as its only variable and returns the number of employees who work under that employee. You are only returning data in Task1 and you do so by querying the reportingstructure endpoint. The endpoint is called like so:

- HTTP METHOD: GET
- URL: /api/reportingstructure/{employeeid}
- RESPONSE: ReportingStructure

If you are running this from localhost and using your browser, you will call this reporting structure endpoint as seen below:

- http://localhost:8080/api/reportingstructure/16a596ae-edd3-4847-99fe-c4518e82c86f

The return will contain two properties:

- **employeeId:** The ID of the employee, this is the same as your input variable.

- **numberOfreports:** This is the number of other employees who work under the employee you are querying.

Considerations

These are the considerations that you must account for while using this endpoint.

1. If your employee ID is not valid then you will not receive a valid return.
2. If you are seeing a zero in the number of reports, the employee which you are querying does not have any direct reports. No employees work under that employee.
3. If your employee self references (i.e. his direct report is himself), the return will be zero.
4. If an employee references a direct report which is in their tree and located higher than they are, this direct report will still count as one but will not count continually.

Technical Overview

Controller:

ReportingStructureController

Database:

Queries the Employee Database
No new database was created for this task.

Model

ReportingStructure

Service:

ReportingStructureService

Tests

ReportingStructureControllerTests

Notes

The Employee Data Access Layer did not natively return the number of direct reports so I created a new method which returns an employee object populated with the direct reports. I then call this method from ReportingStructureService. The new method that I created is in **EmployeeRepository.cs** and named **GetFullEmployee.**

Task 2

Requirements

Create a new type, Compensation. A Compensation has the following fields: employee, salary, and effectiveDate. Create two new Compensation REST endpoints. One to create and one to read by employeeId. These should persist and query the Compensation from the persistence layer.

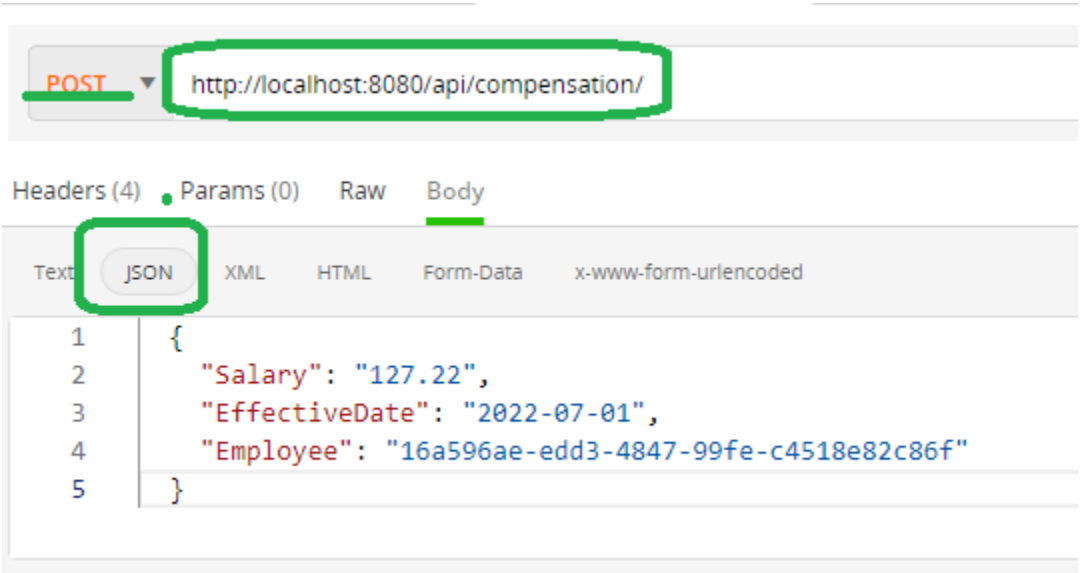
General Overview

This both adds data to the database and also separately retrieves it. As stated in the overview of this document, no data is permanently persisted in these databases therefore you must first create a record before retrieving it.

Creating a Record

HTTP Method: POST
URL: /api/compensation/
PAYLOAD: Compensation
RESPONSE: Compensation

This is an example using fiddler:



The return will contain four properties, three of which you specified in your POST request.

- **CompensationId:** Unique key to this record and primary key for the compensation database.
- **Employee:** ID of the employee who the compensation record is tied to.
- **Salary:** Employee's salary.
- **EffectiveDate:** The date you specified as effective.

Retrieving a Record

Retrieving a record can only be done for a record that has already been created. After you have created a record, you can then retrieve a record by performing the following:

HTTP Method: GET
URL: /api/compensation/{employeeID}
RESPONSE: Compensation

If you are running this from localhost and using your browser, you will call the compensation record as seen below. This example uses the same employeeID as shown in the Create Record example above.

- http://localhost:8080/api/compensation/16a596ae-edd3-4847-99fe-c4518e82c86f

Considerations

These are the considerations that you must account for while using these endpoints.

1. If you do not first create a record, you will not be able to return a record
    a. This is because the database does not permanently persist. Once you close the project, the database resets.
2. If you are not entering a valid date or valid salary value, you will receive null return
3. There is no validation to determine whether or not you have entered a valid employeeID.
    a. You are still allowed to enter an employee id which may not coexist in the employee database.

Technical Overview

Controller:

CompensationController

Database:

CompensationDB (CompensationContext)

Model:

Compensation

Service:

CompensationService

Tests

CompensationControllerTests

Notes

No special notes at this time.

ChangeLog

2022-07-17

- Recreated method of retrieving full employee in EmployeeRepository.
- Recreated method of building ReportingStructure in ReportingStructureService.
- Added new tests to verify expected results when recursive direct report employees exist. An example of this is when an employee references himself as a direct report or when an employee has a direct report that then references the original employee.
- Enhanced null testing in various sections.
- General code cleanup.
- No updates were made to Compensation service.