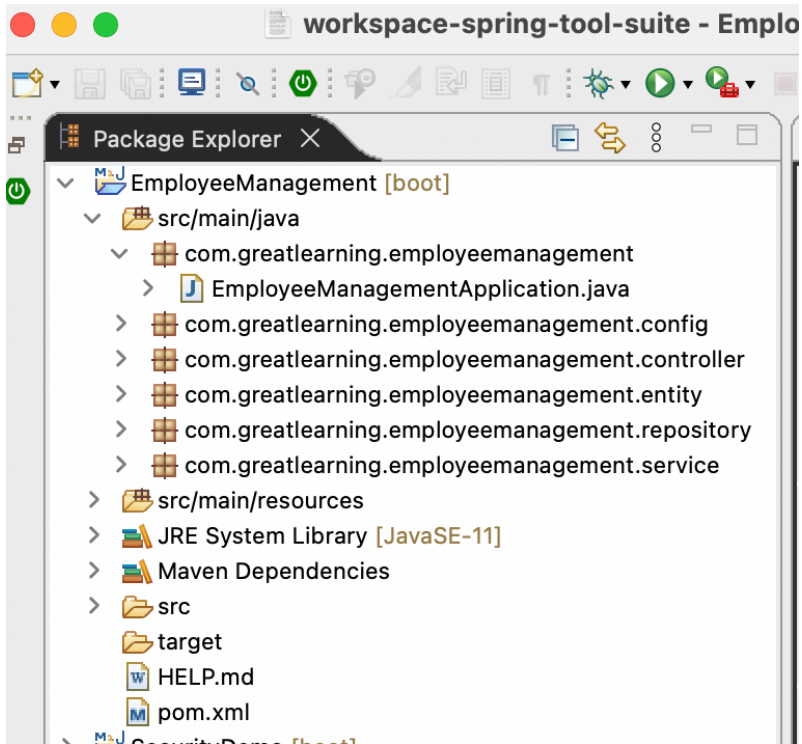


RestAPI assignment

Project structure



Prompts for login. Login as ADMIN - user1.

Please sign in

Sign in

Add a new employee api, employee successfully added

The screenshot shows a Postman interface for a POST request. The URL is `http://localhost:8080/EmployeeManagement/addEmployee?firstName=Dig&lastName=K&email=dig@test.com`. The request body is empty. The response status is 200 OK, and the body contains the text "Employee has been added/updated".

Key	Value	Description
firstName	Dig	
lastName	K	
email	dig@test.com	

Authorization: No Auth

Body: Employee has been added/updated

Get all employees api - gets all employee from H2 DB

<http://localhost:8080/EmployeeManagement/getAllEmployees>

The screenshot shows a Postman interface for a GET request. The URL is `http://localhost:8080/EmployeeManagement/getAllEmployees`. The response status is 200 OK, and the body contains a JSON array of two employee objects.

Authorization: No Auth

Body: JSON

```
[{"id": 1, "firstName": "Dig", "lastName": "K", "email": "dig@test.com"}, {"id": 2, "firstName": "Tej", "lastName": "K", "email": "tej@test.com"}]
```

Get employee by ID. <http://localhost:8080/EmployeeManagement/getEmployeeById?id=1>

http://localhost:8080/

+

...

GET

http://localhost:8080/EmployeeManagement/getEmployeeById?id=1

key	value
<input checked="" type="checkbox"/> id	1
New key	Value

Authorization

Headers

Body

Pre-request Script

Tests

Type

No Auth

Body

Cookies (1)

Headers (12)

Test Results

Pretty

Raw

Preview

JSON

1

{

2

" id": 1,

3

"firstName": "Dig",

4

"lastName": "K",

5

"email": "dig@test.com"

6

}

Get employee by first name. Gets all the employees whose first name starts with “test” (2 employee in this case)

The screenshot shows a REST client interface with the following components:

- URL Bar:** `http://localhost:8080/`
- Method:** `GET`
- URL:** `http://localhost:8080/EmployeeManagement/getEmployeesByName?name=test`
- Query Parameters Table:**

key	value
<input checked="" type="checkbox"/> name	test
New key	Value
- Authorization:** No Auth
- Body:** JSON
- Response:**

```
[
  {
    "id": 4,
    "firstName": "test",
    "lastName": "d",
    "email": "test@test.com"
  },
  {
    "id": 5,
    "firstName": "test",
    "lastName": "p",
    "email": "testp@test.com"
  }
]
```

Get all employees sorted by first name in particular order.

<http://localhost:8080/EmployeeManagement/getEmployeesSortedByName?direction=DESC>

The screenshot shows a web browser window with a REST client interface. The address bar displays the URL: `http://localhost:8080/`. The method dropdown is set to `GET`, and the request URL is `http://localhost:8080/EmployeeManagement/getEmployeesSortedByName?direction=DESC`. The response body is displayed in a tab labeled `Body`, with sub-tabs for `Cookies (1)`, `Headers (12)`, and `Test Results`. The response is formatted as JSON and is displayed in a tab labeled `Pretty`. The JSON response is an array of five employee objects, sorted by first name in descending order. The response is as follows:

```
[
  {
    "id": 6,
    "firstName": "user",
    "lastName": "one",
    "email": "user1@test.com"
  },
  {
    "id": 4,
    "firstName": "test",
    "lastName": "d",
    "email": "test@test.com"
  },
  {
    "id": 5,
    "firstName": "test",
    "lastName": "p",
    "email": "testp@test.com"
  },
  {
    "id": 2,
    "firstName": "Tej",
    "lastName": "D",
    "email": "tej@test.com"
  },
  {
    "id": 3,
    "firstName": "Rikhil",
    "lastName": "R",
    "email": "rikhil@test.com"
  },
  {
    "id": 1,
    "firstName": "Dig",
    "lastName": "K",
    "email": "dig@test.com"
  }
]
```


Update existing employee record

Before sending PUT request for employee id = 6 to change email address from user1@test.com to user.one@test.com

The screenshot shows a REST client interface with the following details:

- URL: `http://localhost:8080/EmployeeManagement/updateEmployee?id=6&firstName=user&lastName=one&em...`
- Method: **PUT**
- Params: `id=6, firstName=user, lastName=one, email=user.one@test.com`
- Authorization: **No Auth**
- Body:

```
{  "id": 6,  "firstName": "user",  "lastName": "one",  "email": "user1@test.com"}
```

After PUT updated record fetched through ID

The screenshot shows a REST client interface with the following details:

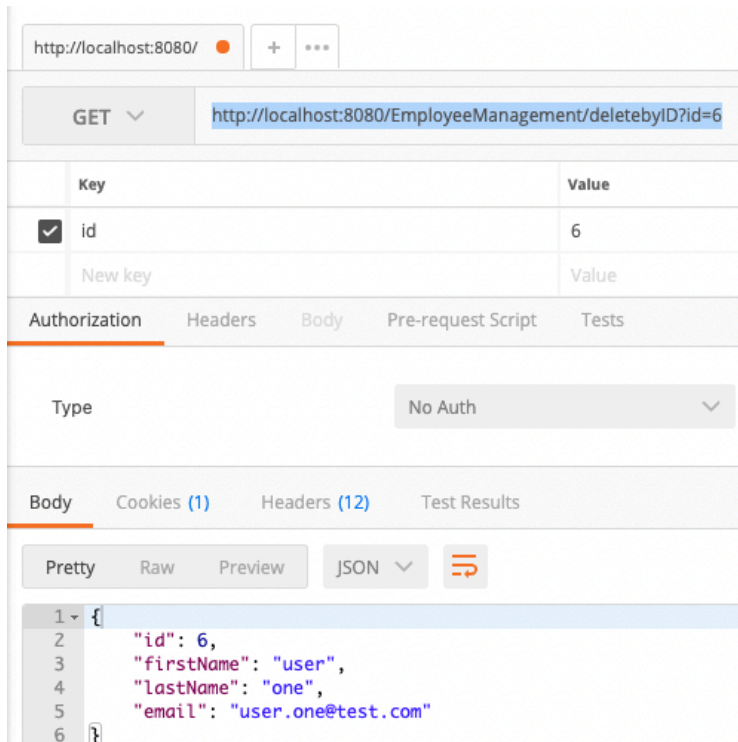
- URL: `http://localhost:8080/EmployeeManagement/getEmployeeById?id=6`
- Method: **GET**
- Params: `id=6`
- Authorization: **No Auth**
- Body:

```
{  "id": 6,  "firstName": "user",  "lastName": "one",  "email": "user.one@test.com"}
```

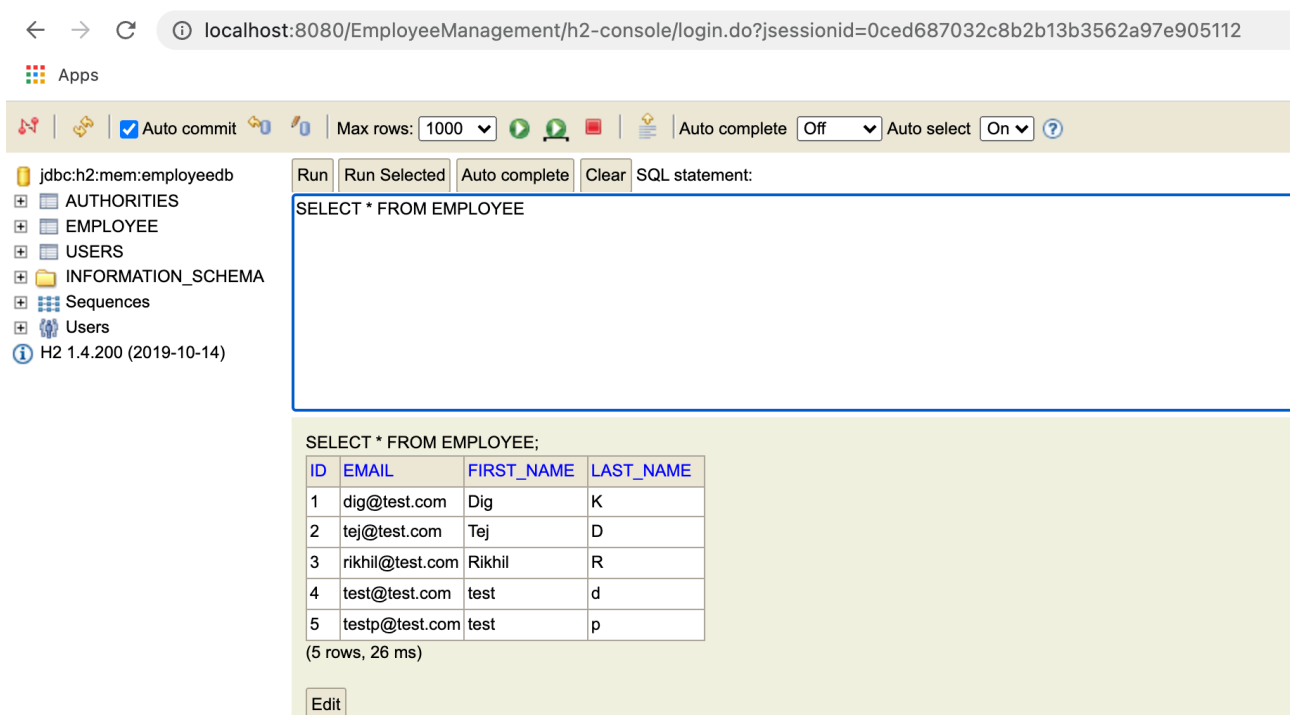
Delete Employee by ID

<http://localhost:8080/EmployeeManagement/deletebyID?id=6>

Before



After - screenshot of H2 DB



Login as user2 with USER role access

Please sign in

user2

.....

Sign in

Tries to add an employee through /addEmployee API which requires ADMIN access and receives 403 error

http://localhost:8080/ + ... No Environment

POST http://localhost:8080/EmployeeManagement/addEmployee?firstName=user&lastName=one&email=user1... Params

Key	Value	Description
<input checked="" type="checkbox"/> firstName	user	
<input checked="" type="checkbox"/> lastName	one	
<input checked="" type="checkbox"/> email	user1@test.com	
New key	Value	Description

Authorization Headers Body Pre-request Script Tests

Type No Auth

Body Cookies (1) Headers (12) Test Results Status

Pretty Raw Preview JSON

```
1 {
2   "timestamp": "2021-12-30T08:10:13.086+00:00",
3   "status": 403,
4   "error": "Forbidden",
5   "path": "/EmployeeManagement/addEmployee"
6 }
```