

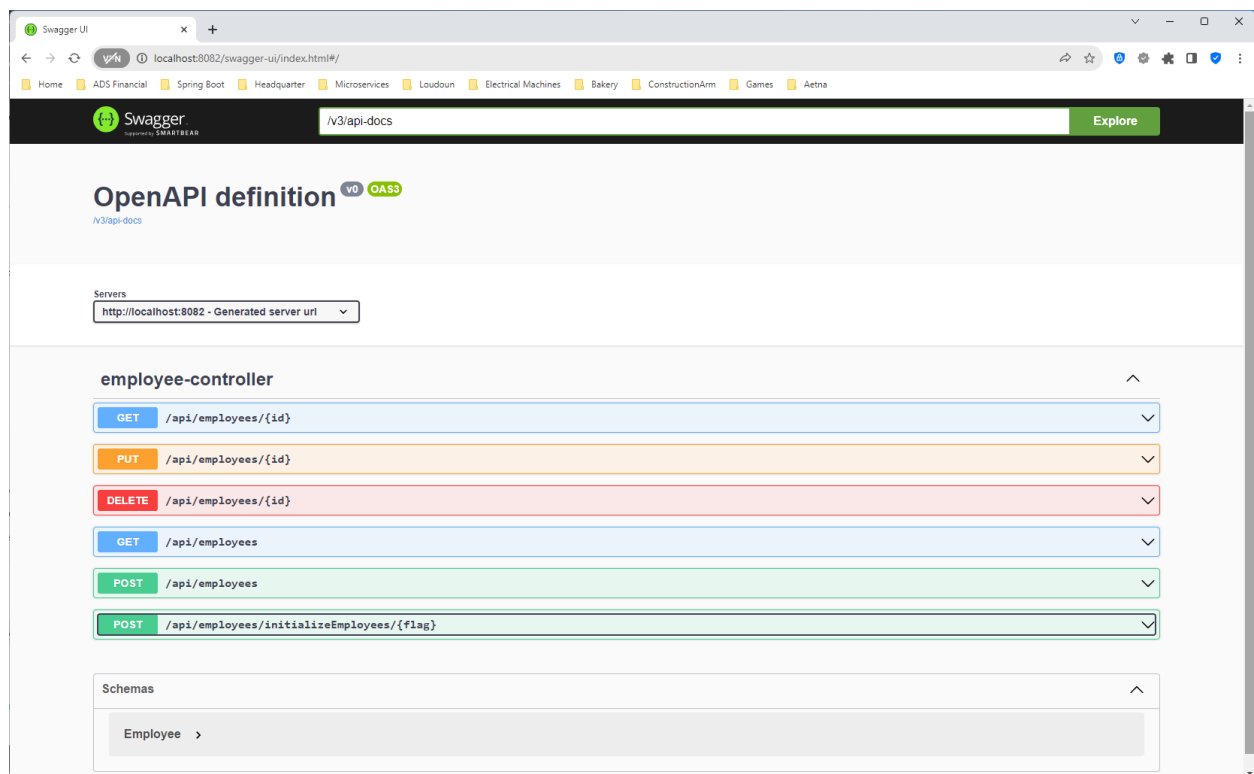
This is a basic report of the Employee API developed by Waseem Nader. The source code for it is located at: <https://github.com/wnader52/spring-boot-employee-rest-api>

The following tools & frameworks are needed to build and run this API:

1. Java 17
2. Spring Boot 3.2.3
3. IntelliJ or Eclipse or any other IDE used for developing Java Spring Boot APIs
4. Please look at the README.md file for usage information

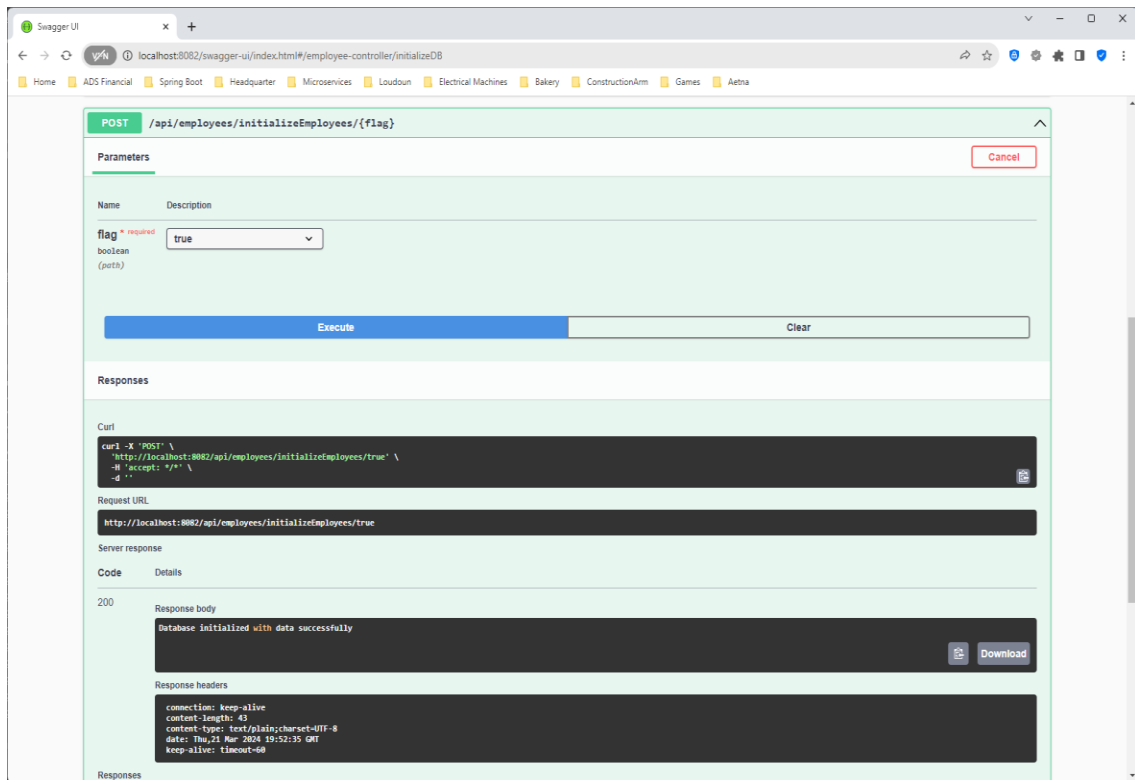
Download the API from the above location and build it so that it can run locally. Once fully built using either Eclipse or IntelliJ IDE, you would be able to run it. Once you run it, access the local URL as shown below via Chrome or any other browser.

Go to <http://localhost:8082/swagger-ui/index.html> to see the swagger schema for its usage.

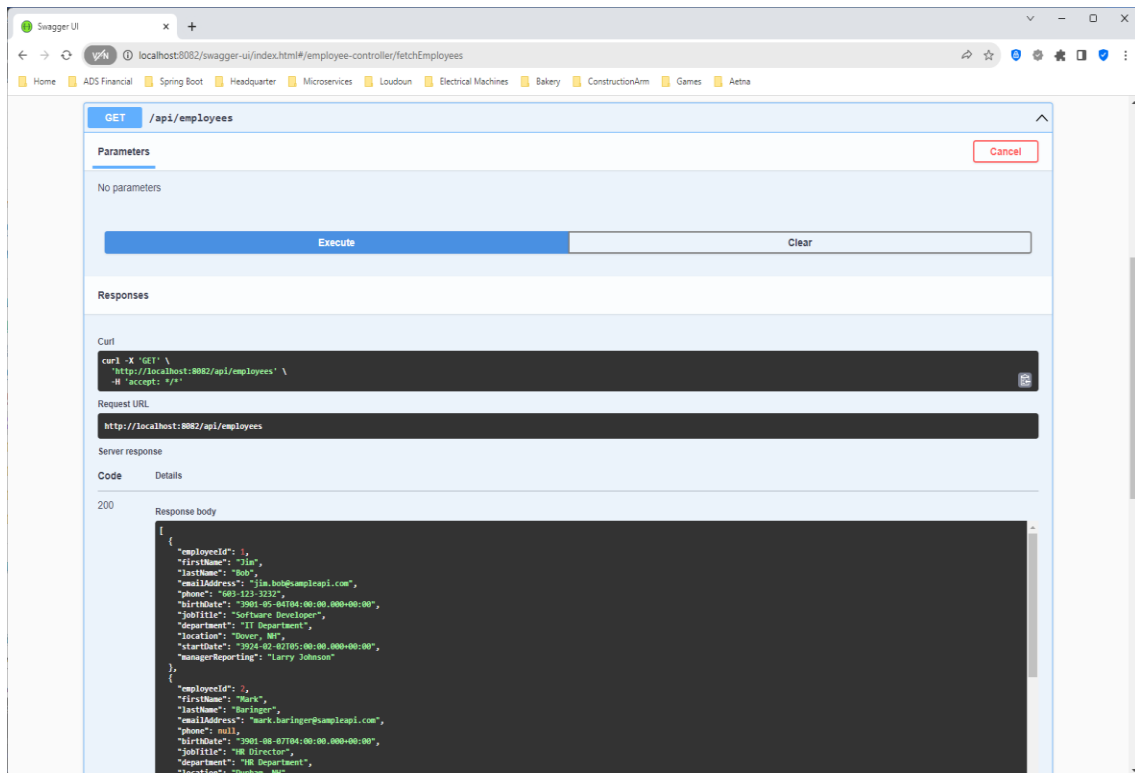


You need to run the `../api/initializeEmployees/{flag}` first so that some dummy data can be generated and stored in memory. Then all subsequent operations can be performed.

1. Running the `../api/initializeEmployees/true` command (create dummy employee objects)



## 2. Executing `../api/employees` (fetch all employees using HTTP GET)



### 3. Executing ../api/employees/1 (fetch individual employee by Id using HTTP GET)

The image shows the Swagger UI interface for an API named "employee-controller". The selected endpoint is `GET /api/employees/{id}`. The parameters section shows a required integer parameter `id` with a value of `1`. The "Execute" button has been clicked, and the response is displayed below.

**Parameters**

Name	Description
<code>id</code> * required	integer(\$int64) (path)

**Responses**

200

Response body

```
{
  "employeeId": 1,
  "firstName": "Jim",
  "lastName": "Bob",
  "emailAddress": "jim.bob@sample1.com",
  "phone": "(603)-123-3232",
  "birthDate": "1961-05-08T04:00:00.000+00:00",
  "jobTitle": "Software Developer",
  "department": "IT Department",
  "location": "Dover, NH",
  "startDate": "1994-02-02T05:00:00.000+00:00",
  "managerReporting": "Larry Johnson"
}
```

### 4. Executing ../api/employees/1 (update using HTTP PUT)

The top screenshot shows the Swagger UI interface for the PUT /api/employees/{id} endpoint. The URL bar shows the local host address. The endpoint is labeled "PUT /api/employees/{id}". The parameters section shows a required integer parameter "id" with a value of 2. The request body is set to "application/json" and contains a JSON object representing an employee. The "Execute" button is visible. The responses section shows a 200 OK status with no links.

The bottom screenshot shows the response body and headers for the PUT /api/employees/{id} endpoint. The URL bar shows the local host address. The response body is a JSON object representing an employee. The response headers include "connection: keep-alive", "content-type: application/json", "date: Thu, 21 Mar 2024 19:56:36 GMT", "keep-alive: timeout=60", and "transfer-encoding: chunked". The responses section shows a 200 OK status with no links. The media type is set to "application/json".

## 5. Executing ../api/employees (creating employee object and saving using HTTP POST)

The image displays two screenshots of the Swagger UI interface, showing the configuration and execution of a REST API endpoint.

**Top Screenshot: POST /api/employees**

- Method:** POST
- Path:** /api/employees
- Parameters:** No parameters
- Request body:** Required, set to application/json. The body contains a JSON object representing an employee:

```
{  "employeeId": 0,  "firstName": "Stephen",  "lastName": "Waldo",  "emailAddress": "s.waldo@gmail.com",  "phone": "786-123-0909",  "birthDate": "1993-03-21",  "jobTitle": "Software Tester",  "department": "R&D Department",  "location": "Baltimore, MD",  "startDate": "2023-03-03",  "managerReporting": "Dilan Thomas"}
```
- Buttons:** Execute, Clear, Cancel, Reset

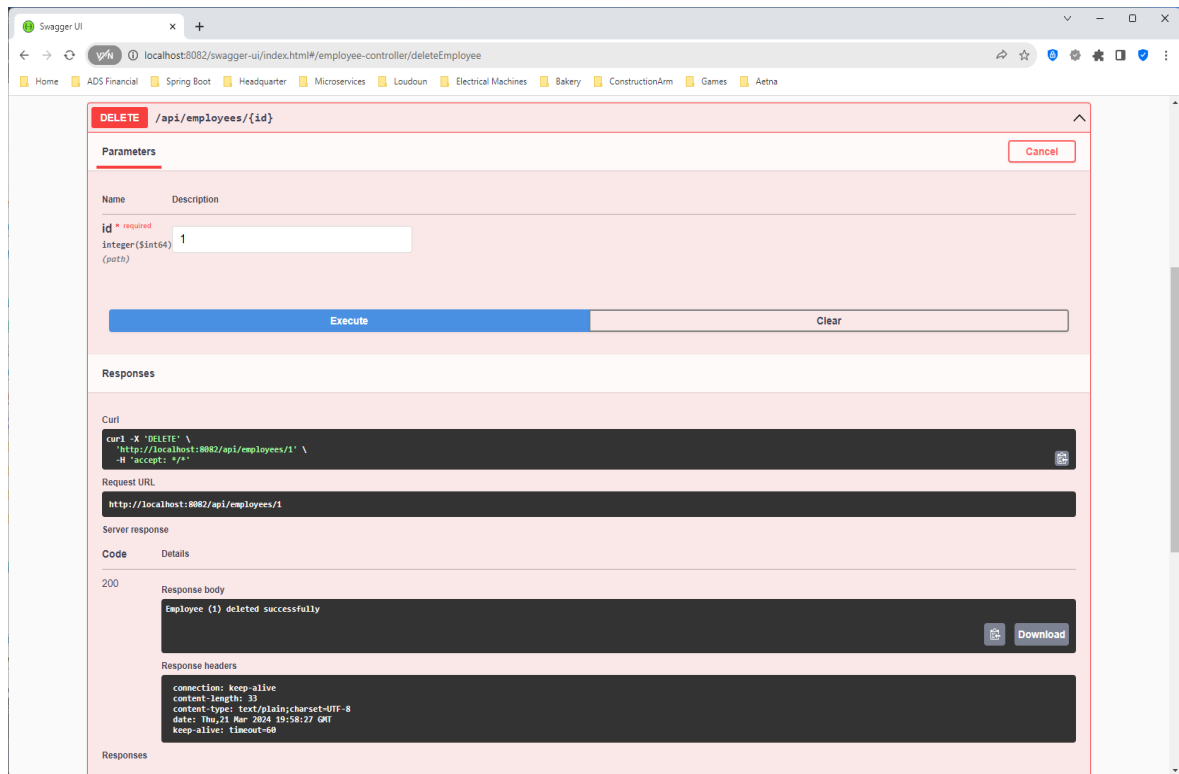
**Bottom Screenshot: Responses**

- Method:** POST
- Path:** /api/employees
- Response Code:** 200
- Response Body:** The response body is a JSON object representing the created employee:

```
{  "employeeId": 1,  "firstName": "Stephen",  "lastName": "Waldo",  "emailAddress": "s.waldo@gmail.com",  "phone": "786-123-0909",  "birthDate": "1993-03-21T00:00:00.000+00:00",  "jobTitle": "Software Tester",  "department": "R&D Department",  "location": "Baltimore, MD",  "startDate": "2023-03-03T00:00:00.000+00:00",  "managerReporting": "Dilan Thomas"}
```
- Response Headers:**

```
connection: keep-alive
content-type: application/json
date: Thu, 21 Mar 2024 20:00:22 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

## 6. Executing ../api/employees/1 (deleting employee by Id using HTTP DELETE)



The Employees API also has some Junit 5 tests that can be executed as shown below.

