Prerequisites:

1. Eclipse IDE: Make sure you have Eclipse IDE installed on your machine. If not, you can download it from https://eclipseide.org/.

2. Java JDK: Ensure that you have Java JDK (Java Development Kit) installed on your machine. Download and install the latest version from https://www.oracle.com/in/java/technologies/downloads/

Steps:

1. Clone Your GitHub Repository:

Clone your GitHub repository to your local machine using the following command:

git clone <repository_url>

2. Open Project in Eclipse:

- Open Eclipse IDE.
- Click on `File` -> `Open Projects from File System`.
- Navigate to your cloned repository directory and select the project.

3. Set Up Eclipse for Maven:

- Right-click on your project in Eclipse.
- Navigate to `Properties` -> `Java Build Path`.
- Select the `Libraries` tab and ensure that Maven Dependencies are included.

4. Build Project with Maven:

- Right-click on your project in Eclipse.
- Navigate to `Run As` -> `Maven Build`.
- In the dialog, set `clean install` as goals and click `Run`.

5. Run the Spring Boot Application:

- Right-click on your main class (usually a class annotated with `@SpringBootApplication`).

- Choose `Run As` -> `Java Application`.

Your Spring Boot application should now be running.

6. Test APIs:

Use Postman or any other API testing tool to test your APIs as per the previous instructions.

Assuming application is running locally on `http://localhost:8080`.

1. Add Item

- Method: POST

- URL: http://localhost:8080/items - Headers:

- Content-Type: application/json

- Request Body: json

```
{
    "name": "Sample Item",
    "description": "This is a sample item."
}
```

- Response:

Item added successfully.

2. Get Item by ID

- Method: GET

- URL: http://localhost:8080/items/1

- Replace `1` with the actual ID of the item you added.

- Response:

```
{
  "id": 1,
  "name": "Sxample Item",
```

"description": "This is a sample item."

}

Note:

- Ensure that your Spring Boot application is running on http://localhost:8080.
- Replace the placeholder data (e.g., item name, ID) with your actual data.
- Use the correct HTTP method for each request (GET, POST).
- Make sure you have the required headers, especially `Content-Type: application/json` for POST requests.
- Verify the response from your Spring Boot application for success or error messages.

I have integrated Thymeleaf for frontend data input, storing in-memory, and attempted deployment on AWS, but currently, it's not functioning; however, you can test the application locally using the provided localhost instructions.

Page Name: Add Item

Purpose:

This page allows users to add a new item by providing a name and description.

- URL: http://localhost:8080/items/addItem

Thymeleaf Template: addItem.html

Usage:

Access the page using the URL /items/addItem.

Fill in the name and description in the form.

Click the "Add Item" button to submit the form.

Success or error messages will be displayed based on the result.

Page Name: View Item Options

Purpose:

This page provides options to view all items or view an item by ID.

- URL: /items/viewItemOptions

Thymeleaf Template: viewItemOptions.html

Usage:

Access the page using the URL /items/viewItemOptions.

Click on the links to navigate to view all items or view an item by ID.