**Spring Product Management**

This documentation includes details about prerequisites, data model, RESTful endpoints, validation, HTTP methods, exception handling, pagination, and unit tests along with snapshots.

**Prerequisites**

* Java 17
* IntelliJ
* Postman or Insomnia
* Git

**Setup project:**

1. Clone the project from the GitHub repository:

* git clone https://github.com/Abirami15/SpringProductManagement.git

1. Open IntelliJ IDEA and import the project as a Maven project:

* Select "File" > "Open" or "Import" and choose the 'productmanagement' directory.
* Make sure to select the pom.xml file during the import process.

1. Set up the JDK to Java 17:

* Navigate to "File" > "Project Structure."
* Under "Project," set the "Project SDK" to Java 17.

1. After downloading dependencies, the application will be configured to run. Run the application:

* Look for the main class (usually annotated with @SpringBootApplication).
* Right-click on the main class and select "Run."

1. Access the H2 database using the link <http://localhost:9090/h2>:

* During the initial connection setup, input the JDBC URL as 'jdbc:h2:mem:pmdb'.
* Set the username as 'sa' and leave the password field empty.

1. Verify that the application has initialized on port number 9090.
2. To test the RESTful endpoints, use either Postman or Insomnia:

* Utilize the provided RESTful endpoints documentation to interact with the application and test various functionalities.

**Data Model**

1. Product Entity

* Table Name: Product
* Attributes: product\_id, product\_desc, product\_name

1. Sub Product Entity

* Table Name: Sub\_Product
* Attributes: sub\_product\_id, product\_id, sub\_product\_desc, sub\_product\_name

**RESTful Endpoints**

1. **Create a New Product and SubProduct**

* Endpoint: <http://localhost:9090/products/create>
* Http Method: POST
* Body: Json Format

{

"Name": "Laptop",

"Description": "High-performance laptop for professionals",

"SubProducts": [

{

"Name": "Dell XPS 15",

"Description": " laptop with Intel Core iseven"

},

{

"Name": "HP Spectre x360",

"Description": "Convertible laptop with touch screen"

}

]

}

* Response: HTTP Status Code: 201 Created

Body: "Product and Sub Product created successfully"

1. **Retrieve Details of a Product and Associated SubProducts by Product ID**

* Endpoint: http://localhost:9090/products/1
* Http Method: GET
* Response: HTTP Status Code: 200 OK

Body : {

"Product Id": 1,

"Name": "Laptop",

"Description": "High-performance laptop for professionals",

"SubProducts": [

{

"SubProduct Id": 1,

"Name": "Dell XPS 15",

"Description": " laptop with Intel Core iseven"

},

{

"SubProduct Id": 2,

"Name": "HP Spectre x360",

"Description": "Convertible laptop with touch screen"

}

]

}

1. **Update the Details of a Product and Associated SubProducts by Product ID**

* Endpoint: http://localhost:9090/products/update/1
* Http Method: PUT
* Body: {

"Name": "Laptop",

"Description": "High-performance laptop for professionals",

"SubProducts": [

{

"Name": "Dell XPS 15",

"Description": " Laptop with Intel Core iFive"

},

{

"Name": "Lenevo Think Pad",

"Description": "Mostly used for Office WOrk"

}

]

}

* Response: Http Status Code: 200 OK

Body: Product and SubProduct updated successfully

1. **Delete a Product and Associated SubProducts by Product ID**

* Endpoint: http://localhost:9090/products/delete/1
* Http Method: DELETE
* Response: Http Status Code: 200 OK

Body: Product and SubProduct deleted successfully

1. **Retrieve a List of All Products with Pagination**

* Endpoint: <http://localhost:9090/products/all>

http://localhost:9090/products/all?page=0&size=2

* Http Method: GET
* Response: Http Status Code: 200 OK

Body: {

"content": [

{

"Product Id": 2,

"Name": "Laptop",

"Description": "High-performance laptop for professionals",

"SubProducts": [

{

"SubProduct Id": 4,

"Name": "Dell XPS 15",

"Description": " laptop with Intel Core iseven"

},

{

"SubProduct Id": 5,

"Name": "HP Spectre x360",

"Description": "Convertible laptop with touch screen"

}

]

}

],

"pageable": {

"pageNumber": 0,

"pageSize": 2,

"sort": {

"empty": true,

"sorted": false,

"unsorted": true

},

"offset": 0,

"unpaged": false,

"paged": true

},

"totalPages": 1,

"totalElements": 1,

"last": true,

"size": 2,

"number": 0,

"sort": {

"empty": true,

"sorted": false,

"unsorted": true

},

"numberOfElements": 1,

"first": true,

"empty": false

}

**Validation**

Stringent validation measures have been implemented for both the "name" and "description" fields to guarantee the provision of mandatory information and adherence to the prescribed data formats.

**HTTP Methods and Status Codes**

Each operation uses appropriate HTTP methods (POST, GET, PUT, DELETE).

Proper HTTP status codes are returned for responses (e.g., 201 Created, 200 OK, 404 Not Found).

**Exception Handling**

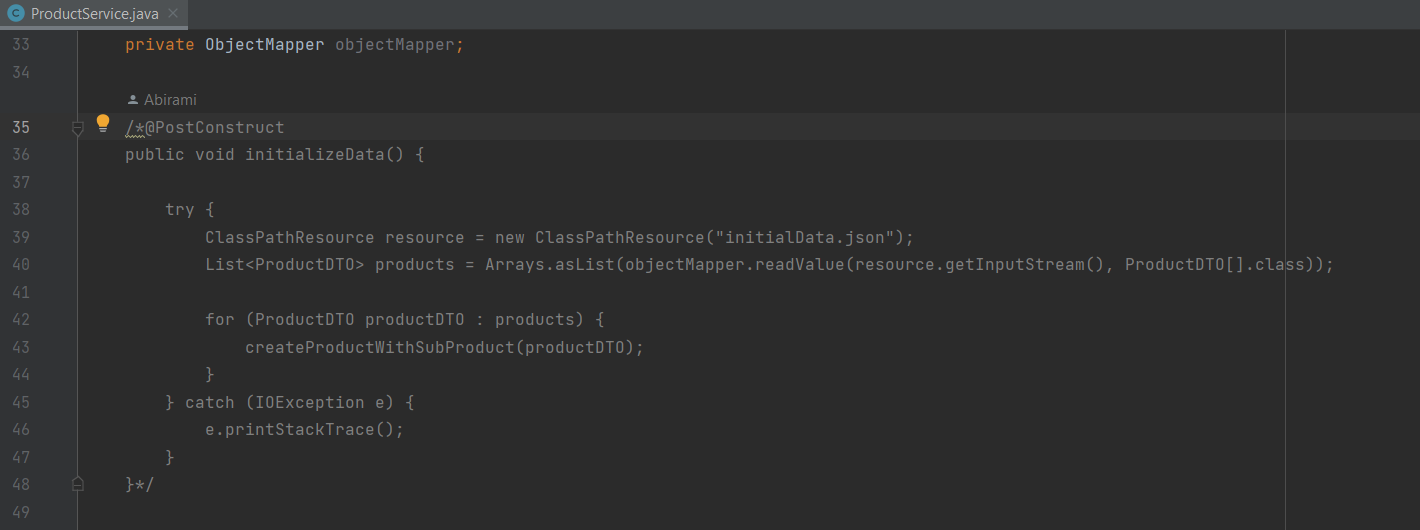
Exception handling is systematically incorporated through the creation of a user-defined exception class, specifically tailored to capture and manage not-found exceptions.

**Pagination**

Pagination functionality has been implemented to facilitate the retrieval of lists encompassing both products and sub-products. To leverage this feature, simply include the parameters 'page' and 'size' alongside the "/all" endpoint.

**Unit Tests**

Unit tests are crafted to verify the functionality of each endpoint, encompassing both positive and negative test scenarios. When executing unit test cases, it is imperative to ensure that the @PostConstruct method is appropriately commented. This method facilitates the initialization of data into the database table during the application's startup, serving application purposes. However, during unit testing, managing this pre-initialized data can pose challenges. Therefore, kindly comment out the @PostConstruct method before initiating the test cases to ensure a smooth and controlled testing environment. Tests cover various scenarios, including successful operations, invalid requests, and error handling.



**Database**

For the database configuration, H2 database has been utilized in this application. It is accessible through the URL http://localhost:9090/h2. During the initial connection setup, the system will prompt for the JDBC URL, which should be set to 'jdbc:h2:mem:pmdb'. The username is 'sa', and the password should be left empty.

Below is a snapshot illustrating the H2 database connection details:

Ensure to follow these specifications to establish a successful connection to the H2 database for your application.

