Software Design Document

Created By

Sujit Kumar

Table of Content

1.Reference

2. Requirement

3.Starting of Project

4.Dependencies

5.Writing Test Cases for Spring Boot based Web Service

6.Writing Program for Web Service

7.Conclusion

Reference

This Service is developed by referring

1.YouTube

2.Dzone Website

Requirement

Project 1: Account Management

Account is associated with a user.

Create: Account – newly created Id in response  
Get: Accounts; filter account by accountId & userId – array of object(s) in response  
Update: Account; based on accountId – no response payload  
Delete: Account; based on accountId – no response payload

Project 2: User Management

Each User has an Account.

Create: User – newly created Id in response  
Get: User filter account by email & userId – array of object(s) in response  
Update: User based on userId – no response payload  
Delete: User based on userId – no response payload

Sample Input

Models to be used :

User.json

{

"id": "int",

"username": "string", unique value

"status": "string", - enum{active,inactive}

"firstName": "string",

"lastName": "string",

"email": "string", - valid email

"createdBy": "string", - userId of the user created

"createdDate": "string", timestamp

"lastModifiedBy": "string", - userId of the user modified

"lastModifiedDate": "string" - timestamp

}

Account.json

{

"id": "Int",

"accountName": "string",

"owner": "Int", - userId of the owner

"createdBy": "string", - userId of the user created

"createdDate": "string", timestamp

"lastModifiedBy": "string", - userId of the user modified

"lastModifiedDate": "string" - timestamp

}

Starting of Project

1.Start with the [Spring Initiali](https://start.spring.io/)zer

2.Add Dependencies

1.Spring web

2.Lombok

3.mockito

4.junit

5.H2

6.Spring Boot DevTools

7.Spring Data JPA

3. Downloading and importing to eclipse IDE

Test Cases

1.In test case we have to write the what we are giving in the request and what we expect in the response based on input and respond our test case will execute we need to go inside src/test/main in UserdataApplicationTests.java

2.Write test case for user and account

3.There are each test case for get, post, update, delete for user and account In test class we need to mention order of test execution order So, there will be total of 8 test cases we can write more test case according to condition

Writing code for microservice

1. In java/main/resource application.property db configuration is written 3 packages for model repository and for controller are created
2. user class and account class in model package and mention the entity in both the class and generate getter and setter based on which our table will be created in h2 database
3. repository interface for both account and user is created and extended to jpa repository
4. controller class for user and account is created and business logic is written for get, put and update and delete
5. We can run the application using main class

Given below are the sample request uri and and body for the microservice we can tested using postman

localhost:8080/info/createUser

create user

{

"id": "5"

"username": "ghost",

"status":"active",

"firstName":"sujit",

"lastName":"kumar",

"createdBy":"1",

"lastModifiedBy":"1",

"email":"sujit@gmail.com"

}

localhost:8080/updateUser/{5}

Update user

{

"id": "5"

"username": "ghost",

"status":"active",

"firstName":"gourav",

"lastName":"singh",

"createdBy":"1",

"lastModifiedBy":"1",

"email":"gourav@gmail.com"

}

Get Account

localhost:8080/info/getUser/5/gourav@gmail.com

Delete Account

localhost:8080/info/deleteUser/5

localhost:8080/info/createAccount

create account

{

"lastModifiedBy": "1",

"accountName": "SBI",

"createdBy": "1",

"owner":"5",

"id" : "1005"

}

localhost:8080/info/updateAcount/{1005}

Update account

{

"lastModifiedBy": "1",

"accountName": "PNB",

"createdBy": "1",

"owner":"5",

"id" : "1005"

}

Get Account

localhost:8080/info/getAccount/1005/5

Delete Account

localhost:8080/info/deleteAccount/5

Swagger API documentation can we seen by visiting

http://localhost:8080/v2/api-docs

http://localhost:8080/swagger-ui.html

Conclusion

1.Microservice is developed based on the requirement

2.All test case is passed Successfully