**Sweet Home**

FSD C6 Course Project: Hotel Room Booking Application

Submitted by-Akash Kumar Sethy

upGrad-[PG Diploma in Software Development - Feb '21](https://learn.upgrad.com/)

**----------------------------------------------------------------------------------------------------------------------------------**

***Code Logics***

* **Database Configuration**

I have used the in-memory database h2. By default, Spring Boot configures the application to connect to an in-memory store with the username “sa” and an empty password. The in-memory database is volatile, and data will be lost when we restart the application.

Configuration in application.yml file

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.userName=sa

spring.datasource.password=

spring.datasource.driver-class-name=org.h2.Driver

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=create

* **Eureka Server Configuration**

Port Number: 8761

Dependencies- Spring Cloud Netflix Eureka Client, Spring Boot Web, Spring Boot Data JPA, H2 Database

For Eureka Clients (BookingService, PaymentService, API Gateway in application.yml file)

eureka:

client:

register-with-eureka: true

fetch-registry: true

service-url:

defaultZone: http://localhost:8761/eureka/

instance:

hostname: localhost

Eureka Server (in application.yml file)

server:

port: 8761

eureka:

client:

register-with-eureka: false

fetch-registry: false

* **API-Gateway Configuration**

Port Number: 9191

Dependencies- Spring Cloud Netflix Eureka Client, Spring Boot Web, Spring Boot Data JPA, H2 Database

server:

port: 9191

spring:

application:

name: API-GATEWAY

cloud:

gateway:

routes:

- id: BOOKING-SERVICE

uri: lb://BOOKING-SERVICE

predicates:

- Path=/hotel/\*\*

- id: PAYMENT-SERVICE

uri: lb://PAYMENT-SERVICE

predicates:

- Path=/payment/\*\*

discovery:

enabled: true

* **Booking Service Configuration**

Port Number: 8081

Dependencies- Spring Cloud Netflix Eureka Client, Spring Boot Web, Spring Boot Data JPA, H2 Database

Controller @RequestMapping(value = "/hotel")

Has the following dependencies

@Autowired

private BookingService bookingService;

This class has two methods for the two endpoints of Booking Service.

1. **bookingDetails** with @PostMapping(value = "/booking")

invokes bookingService.acceptBookingDetails(DTO)

2. **bookingConfirmation** with @PostMapping(value =

"/booking/{id}/transaction")

invokes bookingService.acceptPaymentDetails(DTO)

The second Controller method handles the following two exceptions -

1. “Invalid mode of payment” : checks if payment mode = “UPI” || “CARD”, if true, only

then invokes the bookingService. Otherwise sends back appropriate

ResponseEntity

2. "Invalid Booking Id" : catches **BookingIdNotPresentException** thrown by

bookingService and sends back appropriate ResponseEntity

BookingServiceImpl

Has the following dependencies

@Autowired

private BookingInfoDao bookingInfoDao;

@Autowired

private RestTemplate restTemplate;

@Value("${paymentService.url}") //in applications.properties

private String paymentServiceUrl;

This class contains the following two utility methods

1. **stringToLocalDate** : for converting String to LocalDate

2. **getRandomNumbers** : to generate room numbers based on count

This class contains the following two service methods

1. **acceptBookingDetails** takes fromDate, toDate, aadharNumber, numOfRooms from

DTO. Calculates Room Price, Gets random room numbers from

getRandomNumbers, sets bookedOn to now(), transactionId is set to 0 by default.

2. **acceptPaymentDetails** uses bookingInfoDao to find the bookingInfoEntity

stored in the database. Throws BookingIdNotPresentException if not found.

Uses restTemplate to call Payment Service through API Gateway. Sets the

transactionId. Prints booking confirmation message on console.

DTO Classes

● Booking DTO : To map request body of endpoint 1

● Payment DTO : To map request body of endpoint 2

● Exception DTO : To send back appropriate ResponseEntity in case of Exceptions

* **Payment Service Configuration**

Port Number: 8083

Dependencies- Spring Cloud Netflix Eureka Client, Spring Boot Web, Spring Boot Data JPA, H2 Database

Controller @RequestMapping(value = "/payment")

Has the following dependency

@Autowired

private PaymentService paymentService;

This class has two methods for the two endpoints of Payment Service.

1. **transactionConfirmation** with @PostMapping(value = "/transaction")

invokes paymentService.acceptPaymentDetails(DTO)

2. **transactionDetails** with @GetMapping(value = "/transaction/{id})

invokes paymentService.getTransactionDetails(id)

Catches Exception and sends appropriate message if transaction not found.

PaymentServiceImpl

Has the following dependencies -

@Autowired

private TransactionDetailsDao transactionDetailsDao;

This class contains the following two service methods

1. **acceptPaymentDetails** uses DTO to create a new transactionDetailsEntity

and saves it in the database using transactionDetailsDao.

2. **getTransactionDetails** uses transactionDetailsDao to find the

transactionDetailsEntity.Throws exception if transaction not found.

DTO Class

Payment DTO : To map request body of endpoint 1