Hotel Room Booking Application Documentation

Booking Service:

Spring Boot project is created with the following dependencies in

POM.xml:

1)Eureka Discovery Client

2)Spring Data JPA

3)MySQL Driver

4)Spring Web

5)Apache Kafka dependency

Application. Properties:

Required configurations are updated in this file like

1)server.port

2)MySql connection

3)paymentServiceUrl configuration

4)kafka topicName

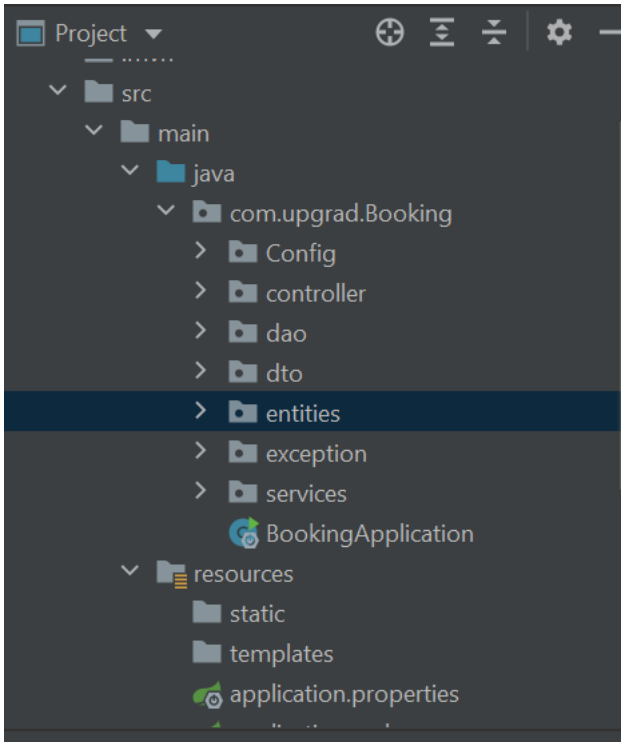
5)Spring application name

Application.yml:

Eureka client related properties are set in this file.

Main-Java:

In Booking Service Spring Boot Application, there are different packages namely Config, Controller, Dao, Dto, entities, exception, services as below.



Entities: entities package has a class “BookingInfoEntity” which has all the fields as per the given schema. Proper annotation is used to make this class an Entity class.

Dto: dto package has classes “BookingDto” and “PaymentDto”. It has all the fields same as entities (table columns). It is used in Controller layer.

Dao: dao Package has an interface “BookingDao” which extends JpaRepository. This dao layer is used to connect to data base.

Services: It has an interface (BookingService) and its implementation class (BookingServiceImpl). BookingService interface has abstract methods which are used to create business logic. All these abstract methods are implemented in BookingServiceImpl class.

This class is Autowired with BookingDAO, Producer and RestTemplate class objects. Proper annotation is applied to make this class a service class.

@Value annotation is used to get the value paymentServiceUrl which is stored in application.properties file.

The below methods are implemented in BookingServieImpl class.

acceptBookingDetails() : Saves all the booking details in database.

updateTransactionId() : Updated the transactionId in database once payment is done.

getBookingDetails() : This method is used to get all the booking info.

getRoomNumbers() : This method logic generates the random room numbers.

getRoomPrice() : This method logic calculates the room price based on fromDate, toDate and numOfRooms.

confirmBooking() : This method calls the PaymentService end point using RestTemplate and gets the transactionId. This transactionId is saved in database by invoking updateTransactionId().

This method also throws RecordNotFoundException “Invalid Booking Id”, if the booking id is invalid.

It also throws InvalidInputException “Invalid mode of payment”, if the payment mode is other than CARD/UPI.

Exception: This package has two classes “RecordNotFoundException” and “InvalidInputException” which extends RunTimeException.

There is another package inside exception, namely handler.

Handler: This package has a class “CustomExceptionHandler”, where in the logic is implemented to send the error message based on the input provided.

Controller: This package has a class “BookingController”. It contains two endpoints (PostMapping) one for saving booking details in database and other for payment (getting transactionId from confirmBooking() of BookingService class) if user wishes to confirmBooking by paying.

This class is Autowired with BookingService and ModelMapper class objects. Proper annotations are used to make this class a controller class.

Config: This package has a class named KafkaConfig. This class creates a producer bean and returns a kafkaProducer for which all the kafka properties are set.

Booking Application: In the main BookingApplication, ModelMapper and RestTemplate Beans are created.

Payment Service:

Spring Boot project is created with the following dependencies in

POM.xml:

1)Eureka Discovery Client

2)Spring Data JPA

3)MySQL Driver

4)SpringWeb

Application.properties:

Required configurations are updated in this file like

1)server.port

2)MySql connection

3)Spring application name

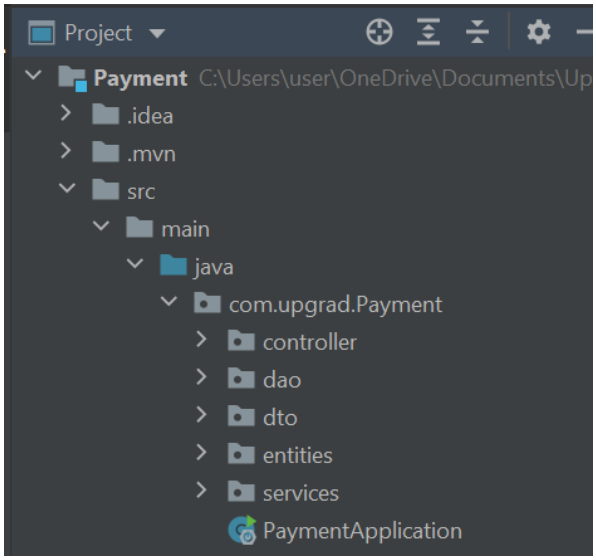
Application.yml:

Eureka client related properties are set in this file.

Main-Java:

In Payment Service Spring Boot Application, there are different packages namely

Controller, Dao, Dto, entities, services as below.



Entities: entities package has a class “TransactionDetailsEntity” which has all the fields as per the given schema. Proper annotation is used to make this class an Entity class.

Dto: dto package has classes “PaymentDTO”. It has all the fields same as entities (table columns). It is used in Controller layer.

Dao: dao Package has an interface “PaymentDAO” which extends JpaRepository. This dao layer is used to connect to data base.

Services: It has an interface (PaymentService) and its implementation class (PaymentServiceImpl). PaymentService interface has abstract methods which are used to create business logic. All these abstract methods are implemented in PaymentServiceImpl class.

This class is Autowired with PaymentDAO class object. Proper annotation is applied to make it a service class.

This class has the below methods.

acceptTransactionDetails() : This method is used to save the transaction details in transaction table.

getTransactionDetails() : This method gets the transaction details based on transaction id.

Controller: This package has a class named “PaymentController” in which there are two endpoints (PostMapping and GetMapping). PostMapping end point is used by BookingService to get the transaction id. GetMapping endpoint is used to get transaction details based on transaction id.

This class is Autowired with PaymentService and ModelMapper class objects. Proper annotations are used to make this class a controller class.

Payment Application: In the main PaymentApplication, ModelMapper and RestTemplate Beans are created.

Eureka Server:

Spring Boot project is created with the following dependencies in

POM.xml:

1)Eureka Server

Application.yml:

Eureka server related properties are set in this file.

EurekaApplication: The main EurekaApplication is annotated with @EnableEurekaServer.

Notification Service:

Java-Maven Project is created. “Consumer” Class is created under the package “com.upgrad.Notification.KafkaConsumer”

Inside the Consumer class, logic is implemented to set all the properties for Kafka consumer and subscribe to the topic and consume messages in forever loop.