Home Devices Control System I

Problem Statement:

Problem | Submissions | Solutions | Doubts

Problem statement Send feedback

As a developer, you are tasked with developing a Smart Home Control System using Spring Boot. The system should allow users to remotely control various smart devices in their homes. To ensure a smooth development process and efficient monitoring, you must integrate Spring Boot DevTools and Actuators into your application.

▼ Tasks:-

1. Complete the **SmartDevice** entity class with relevant annotations with the following attributes.
   - id (Integer)
   - name (String)
   - type (String)
   - status (String)
   - roomId (Integer)
2. Complete the **Room** entity class with relevant annotations with the following attributes.
   - id (Integer)
   - name(String)
3. Complete the **DeviceDTO** class by adding proper Lombok annotations with these attributes.
   - name (String)
   - status (String)
   - roomId (Integer)
4. Complete the **RoomController** class by auto-wiring the necessary dependencies and creating the following APIs by completing the relevant controller methods.
   - GET "/room/id/{id}": This API allows the user to fetch the room record by its ID.
   - GET "/room/all": This API allows the user to fetch all the room records.
   - POST "/room/add": This API allows the to create a room record and save it to the database.
   - DELETE "/id/{id}": This API lets you delete a room record from the database.

   Note: All variables in the APIs are Pathvariables.

Problem | Submissions | Solutions | Doubts

5. Complete the **DeviceController** class by auto-wiring the necessary dependencies and creating the following APIs by completing the relevant controller methods.
   - GET "/device/all": This API allows the user to fetch all records of smart devices from the database.
   - GET "/device/id/{id}": This API allows the user to fetch a record of a smart device from the database.
   - POST "/device/add": This API allows users to create a Smart Device record. (@RequestBody DeviceDTO deviceDto).
   - PUT "/device/changeStatus": This API allows users to change the status of the smart device.

   Note: All variables in the APIs are Pathvariables.
6. The status should have only two values "On" or "Off". An exception should be thrown if a user tries to add or update a smart device with an invalid status. You have been provided with a custom exception **(InvalidStatusException)**; use this in your code wherever required.
7. Complete the **RoomService** class method by adding relevant business logic corresponding to their respective APIs.
8. Complete the **DeviceService** class methods by adding relevant business logic corresponding to their respective APIs.
9. Create **DeviceRepository and RoomRepository** interfaces extending **JpaRepository<>** with proper parameters and add a required annotation.
10. Test the APIs using Postman.
11. Add the DevTools Dependency.

▼ Reference image

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <scope>runtime</scope>
  <optional>true</optional>
</dependency>
```

12. Create a new API in the **DeviceController** class.

- DELETE "/device/id/{id}" - This API allows the user to delete a device record from the database.

13. Save the code and access the delete API without re-running the application to ensure live reloading

▶ **Special Instructions for submitting the solution:**

▶ **Note:-**