Spring Boot MongoDB REST API and git integration

28 October 2021 16:55

1) Go to https://start.spring.io.
2) Select Maven Project as Project type, Java as the Language, latest Spring Boot version(non-snapshot), fill in the details for Project Metadata, choose Jar as the packaging type, select Java 11.
3) Add the following dependencies by clicking on Add Dependencies:
    a. Spring Web - to build the web app
    b. Spring Data MongoDB - to access MongoRepository
    c. Lombok - auto create no-args and all-args constructors, getters, setters, and toString methods.
4) Click on Generate. A .zip folder will be downloaded. Unzip the folder.
5) Open Eclipse IDE. Click on File > Import > Search for Maven > Existing Maven Projects. Click on browse(root directory) and select the folder you had unzipped in Step 4.
6) A pom.xml file will be detected in Projects. Click on Finish. Your project will be visible in the Project Explorer Tab.
7) Go to https://github.com and Log In to your GitHub account.
8) Click on the '+' icon on the top right(next to your account name) and then click on 'New Repository.'
9) Enter in the Repository name(name of your project) and then click on Create Repository.
10) Copy the Repository URL from the Quick setup tab(HTTPS).
11) In your Eclipse IDE, right click on your project directory in the Project Explorer and click on Team, then click on Share Project, then check the 'Use or create repository in parent folder of project' option, then click on create repository, and then click on Finish(equivalent of git init). This will create a local repository for your project.
12) Right click on your project, then click on Team, and then click on Add to Index(equivalent of git add.).
13) Right click on your project, then click on Team, then click on Commit, enter in commit message, and then click on the Commit button(equivalent of git commit).
14) Right click on your project, then click on Team, and then click on Remote, then click on Push.
15) In the URI field under Location, paste in the link you had copied in Step 10.
16) Under authentication, enter in your username or email address for User field, and then paste your Git Token(Follow these steps to get a Git Token: Creating a personal access token - GitHub Docs) for Password field.
17) Click on Finish(equivalent of git remote add origin <remote-repo-uri> and git push -u origin master/ git push origin master).
18) Your project will now be visible under your Git Repository in your GitHub Page.
19) Create these packages by right clicking on the src/main/java directory in your project(in the Project Explorer- Eclipse IDE) and then clicking on New > Package:
    a. {Root package name}.models
    b. {Root package name}.repositories
    c. {Root package name}.services
    d. {Root package name}.services.impl
    e. {Root package name}.resources
20) Open application.properties(under project directory > src/main/resources) and type in these MongoDB connection strings:
    a. spring.data.mongodb.host=127.0.0.1
    b. spring.data.mongodb.port=27017
    c. spring.data.mongodb.database=jobs
21) {Root package name}.models package: Represent fields of a MongoDB document as a Java Class(Bean). Use these lombok annotations for the class: @Data(Generate getters, setters, toString), @NoArgsConstructor, @AllArgsConstructor. Use @Document data.mongodb annotation. Use @Id data.annotation for the id field variable(unique identifier field of a

document).

22) {Root package name}.repositories: Create an interface that extends MongoRepository. MongoRepository<T, ID>: T => model, ID => Unique field. With this, we get access to many methods that interact with the MongoDB database and perform CRUD operations.
23) {Root package name}.services: Create an interface that has methods for CRUD operation.
24) {Root package name}.services.impl: Create a class that implements the service you had created in Step 23. Use @Service stereotype annotation on the class. Create a private object of the repository that you created it. Use @Autowired annotation on that object, or use constructor-based dependency injection. Provide implementation for all the methods using the inbuilt repository methods to interact with the database and perform CRUD operations.
25) {Root package name}.resources: Create a class that handles HTTP Requests by using the @RestController web.bind annotation. Use @RequestMapping, @PostMapping, @GetMapping, @PutMapping, and @DeleteMapping annotations on top of the methods to handle the respective requests. Provide the path as a String in parentheses next to the mapping annotations(Eg: @GetMapping("/cars/{modelNumber}"). Create a private Object of the service and use constructor-based dependency injection. Handle the HTTP requests sent to the endpoints with methods that utilize the methods written in service implementation in order to perform the relevant CRUD operations and return relevant response and status code to the client.
26) Run the application by right clicking on {Project name}Application.java(under project > src/main/java) and then by clicking on Run As > Java Application.
27) Test your application using Postman Desktop Client.
28) Redo steps 12 to 17. Your GitHub repository of the project will now have the updated version of your project.