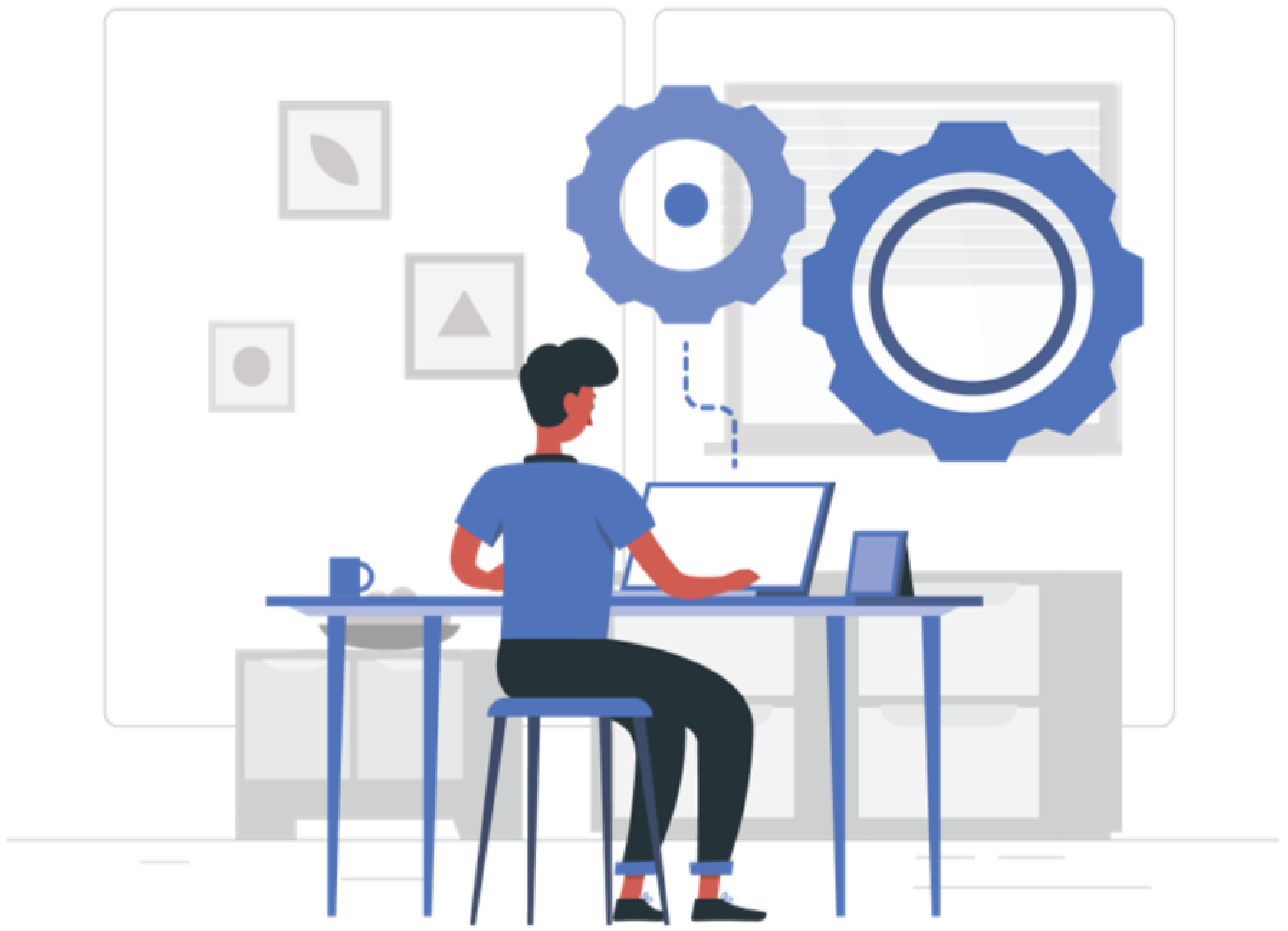
Press Esc to exit full screen



Challenge

Manage Semi-structured and Unstructured Data and Handle Exceptions Within a RESTful Service by Using Mongo Repository

Implementation Environment

- Refer to the documentation below before starting the challenge.
  - Spring Data Mongo
- If MongoDB does not start automatically, follow these steps in Windows:
  - ***Goto -> Control Panel -> Administrative Tools -> Services -> double click -> search for MongoDB Server(MongoDB) -> right click and start service***

Muzix Application

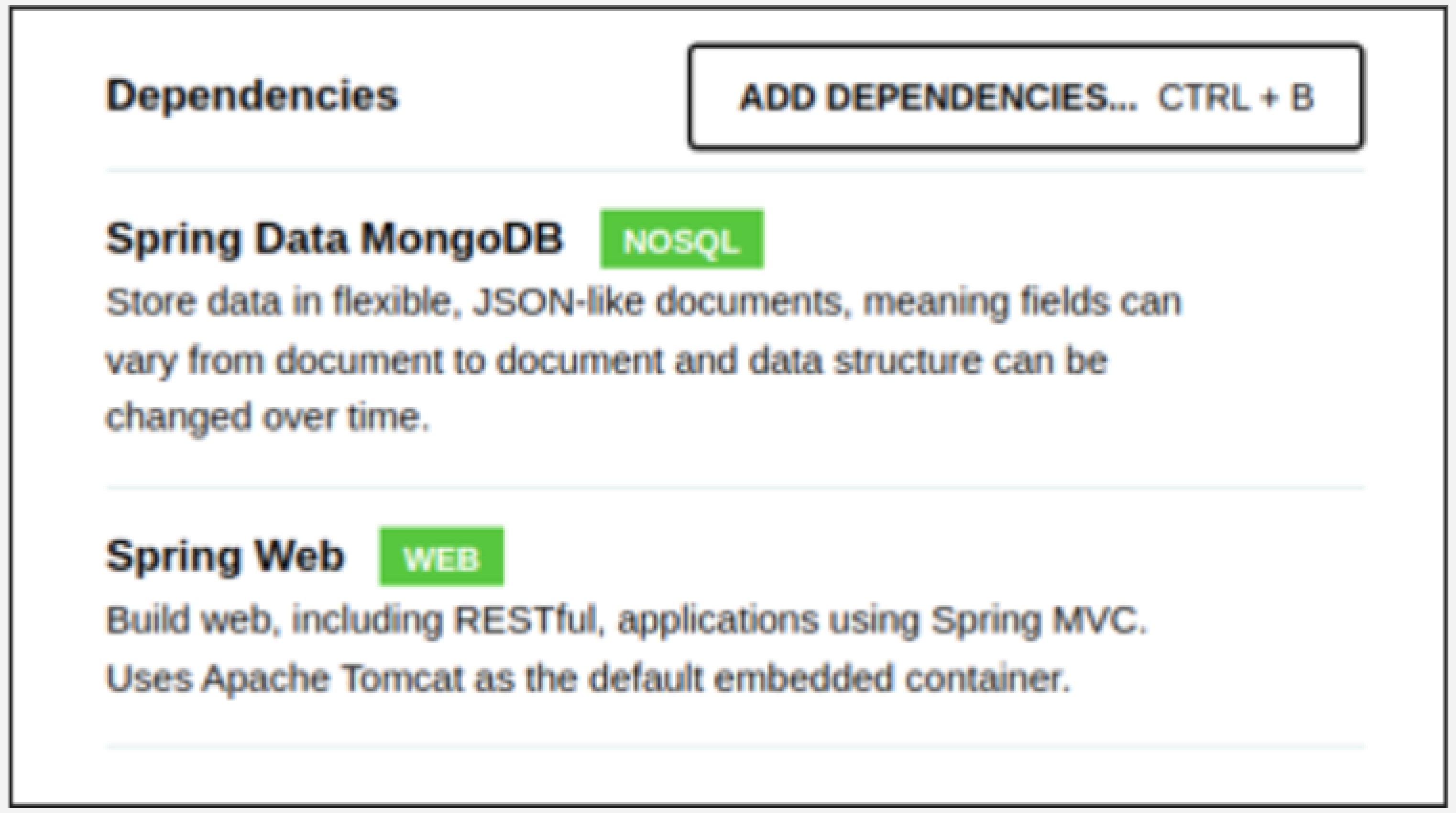
A music streaming application enables users to listen to music on the go. The streaming application requires the track details of all the songs streamed on the application.

Create a Spring Boot application that has the artist and the track as domain classes. Implement the different layers of the application.
Use the MongoDB database for storing the data.
Ensure that all the end points are tested using Postman.

CHALLENGE

Challenge Instructions # 1

- Create a Spring Boot application from the Spring Initializr.
- Add the necessary dependencies in pom.xml.
- Download the project into your local machine.
- Extract the zip file.
- Export the project in your local IDE.

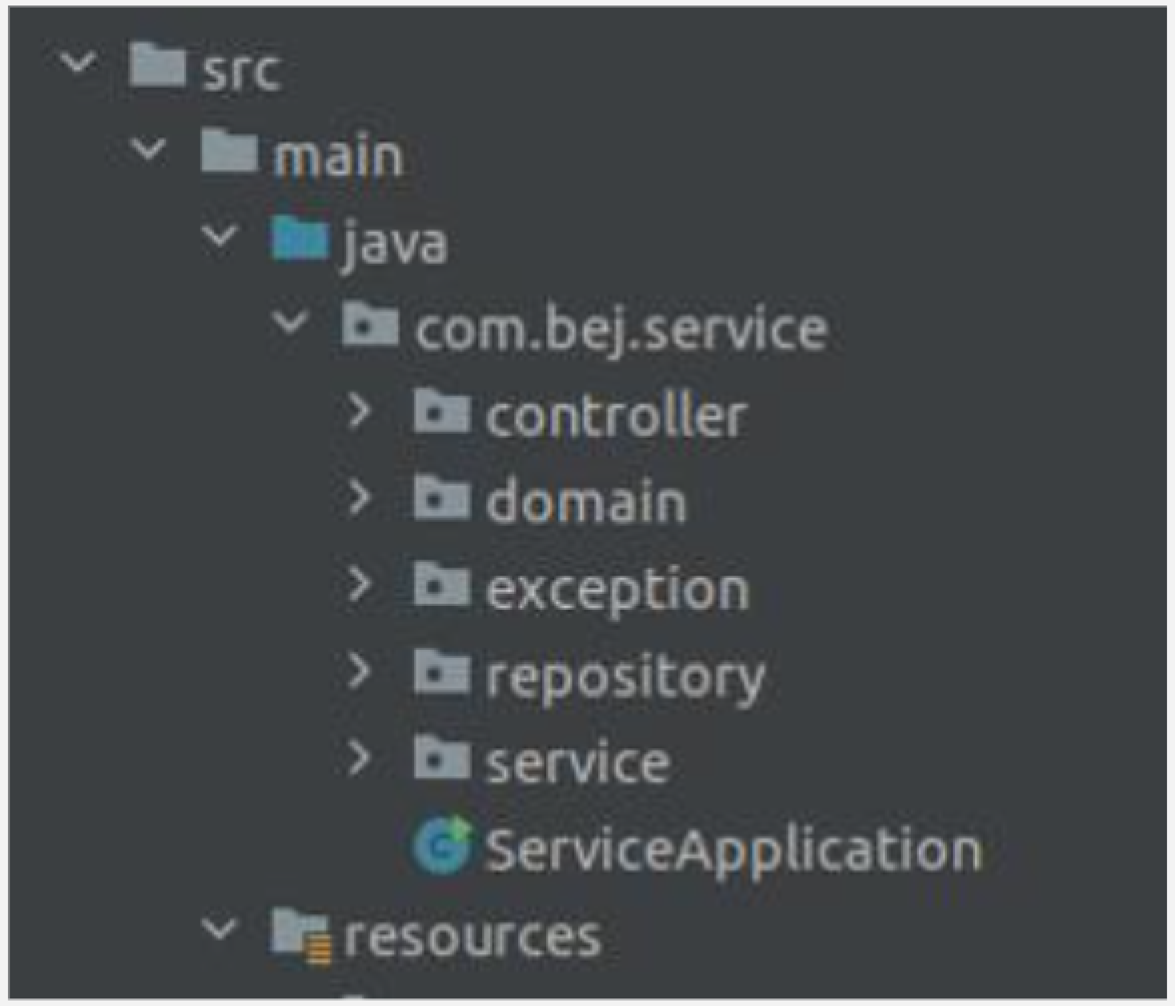


Spring Data MongoDB NOSQL
Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Task # 1 – Domain Classes

- Define the domain classes within the `domain` package
  - `Track` with attributes `trackId`, `trackName`, `trackRating`, `trackArtist` of type Artist
  - `Artist` with attributes `artistId`, `artistName`
- Provide appropriate `@Document` and `@Id` annotations for the `Track` class.
- Generate:
  - Getter and setter methods
  - Constructors – no argument and parameterized
- Override the `toString()` method.



```
src
  main
    java
      com.bej.service
        controller
        domain
        exception
        repository
        service
        ServiceApplication
    resources
```

Structure of the Project

Task # 2 – Repository Layer

- Define a `TrackRepository` interface that will inherit the `MongoRepository` inside the `repository` package.
- The `TrackRepository` will have two parameters:
  - The class annotated with `@Document`
  - The datatype of the attribute annotated with `@Id` attribute
- Define the following methods within the `TrackRepository:`
  - Method that will fetch all the details of the track where `trackRating` is greater than 4.
  - Method that will fetch all the details of tracks for the artist (Justin Bieber).
- Both methods will return a `List` of Track objects.
- Annotate the method with `@Query` and write the query.

Task # 3 – Service Layer

- Create the `ITrackService` interface and `TrackServiceImpl` class inside the `service` package to provide the business logic for the application.
- Annotate the `TrackServiceImpl` class with the `@Service` annotation.
- Create methods in the `ITrackService` interface for the actions below:
  - Save a Track and return the saved Track object.
  - Delete a Track and return true if success, and false, if failure.
  - Retrieve all the Tracks present in the database as a List.
  - Retrieve all the Tracks where trackRating is greater than 4.
  - Retrieve all the Tracks for the artist (Justin Bieber)
- Exception handling should be done for all the methods.

Task # 3 – Service Layer (contd.)

- The `TrackServiceImpl` class must implement the `ITrackService` interface and provide implementation for all its methods.
- Autowire `TrackRepository` within the `TrackServiceImpl` class.
- Make calls to the appropriate methods of the `TrackRepository` methods in the `TrackServiceImpl` class.

Task # 4 – Controller Layer

- Create the `TrackController` class in the controller package.
- Annotate the class with the `@RestController` and `@RequestMapping` annotations.
- Autowire the `TrackServiceImpl` class in the TrackController.
- Define all the handler methods `(@PostMapping, @GetMapping, @DeleteMapping)` for handling the requests from the client.
- Call the appropriate service layer methods to process all the responses.
- Send the response back to the client.
- Set up the MongoDB database configuration details in the `application.properties` file.
- Run the boot application by using the Spring way of execution.
- Open the Postman and call all the REST API.

Submission Instructions

- Create a new repository on Git named **BEJ_C2_S2_REST_API_MONGODB_MC_1.**
- Push your code into the repository.
- There is no boilerplate for this challenge.