01 - Spring Data Rest Introduction

Spring Data REST is a part of the Spring framework that automatically exposes RESTful endpoints for repositories defined by Spring Data.

- It eliminates the need to manually create controller and service layers.
- It automatically handles basic CRUD operations for entities based on repository interfaces, making the development process faster and more efficient.

👉**Advantages of Spring Data Rest:**

1. Automatic RESTful API Generation:

- Spring Data REST automatically exposes REST endpoints for repositories.
- There's no need to create Controller and Service classes manually.

2. Repository Exposure:

- It works directly on repository interfaces like CrudRepository, JpaRepository, and PagingAndSortingRepository, exposing common operations such as:
  - GET for fetching resources.
  - POST for creating new resources.
  - PUT/ PATCH for updating resources.
  - DELETE for removing resources.

3. HATEOAS:

- Hypermedia As The Engine of Application State (HATEOAS) allows clients to navigate between related resources easily by following hypermedia links included in the response.

4. Customization:

- While Spring Data REST provides default implementations, but still we can customize the endpoints, and add validations using event listeners, projections, and custom queries if needed.

5. Paging and Sorting:

- By using PagingAndSortingRepository, Spring Data REST can handle pagination and sorting out of the box.

6. Repository Events:

- It allows you to capture and handle events such as BeforeCreate, AfterCreate, BeforeSave, and AfterDelete to add custom logic at various stages of the repository lifecycle.

7. Ease of Use:

- It eliminates the need for boilerplate code.
- REST endpoints can be customized through annotations like @RepositoryRestResource.

02 - Creating A Data Rest Project

👉 **Steps to Create a Spring Data REST Project:**

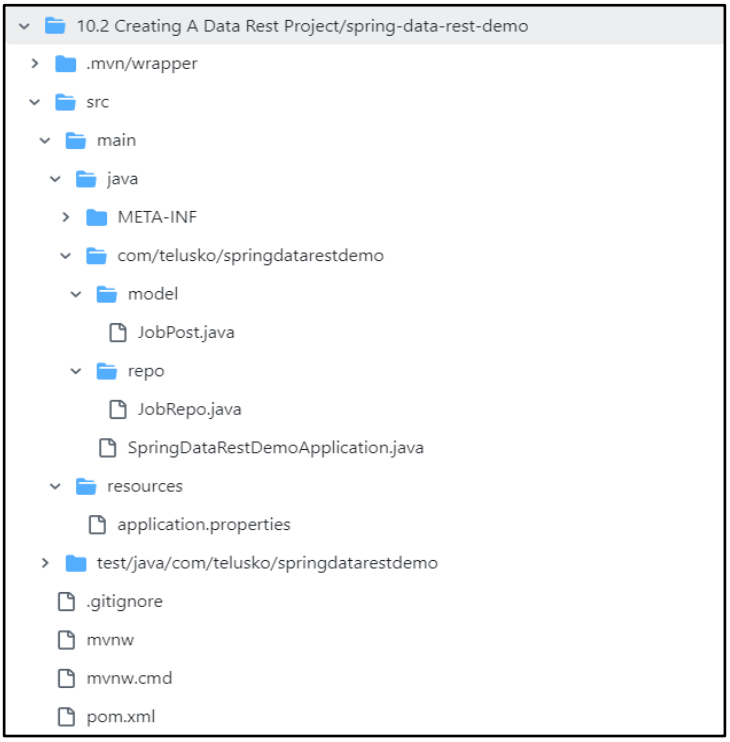
1. **Create a Maven Project:**
   - Create a new Maven project.
   - Provide the necessary Group ID and Artifact ID.
2. **Add Required Dependencies in pom.xml:**
   - Add the necessary dependencies in the pom.xml file to include Spring Data JPA, Spring Data REST repositories, Lombok, and PostgreSQL driver.

```
<dependencies>
    <!-- Spring Data JPA -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <!-- Spring Data REST -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-rest</artifactId>
    </dependency>

    <!-- Lombok -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <scope>provided</scope>
    </dependency>

    <!-- PostgreSQL Driver -->
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <scope>runtime</scope>
    </dependency>
```

3. **Database Configuration in application.properties:**
   - Add the PostgreSQL database configuration in the application.properties file located in the src/main/resources directory.

```
# Database configuration
spring.datasource.url=jdbc:postgresql://localhost:5432/your_database_name
spring.datasource.username=postgres
spring.datasource.password=your_password
spring.datasource.driver-class-name=org.postgresql.Driver

# JPA and Hibernate configuration
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

👉Project Structure:



Link for the Code:
https://github.com/navinreddy20/spring6-course/tree/c6690e4f2c70d8f530d70623f13d14ff0ffd7e7d/10%20Spring%20Data%20Rest/10.2%20Creating%20A%20Data%20Rest%20Project/spring-data-rest-demo

TELUSKO

03 - Running the Project

In Spring Data REST, there is no need to create controllers manually. The repository layer is automatically exposed as RESTful web services, making it simpler to interact with data entities.

👉 Accessing Data via URL:

- Once the project is running, we can access and retrieve data by visiting specific URLs.
- For example, if we have a JobPost entity and repository, we can retrieve all job posts by visiting:

  http://localhost:8080/jobPosts

👉 HATEOAS Support:

- Spring Data REST automatically provides **HATEOAS (Hypermedia as the Engine of Application State)**. This means that the API responses not only provide the requested data but also include links to related resources, such as:
  - Self-links to the resource.
  - Links to related entities or actions (e.g., GET, POST, DELETE).
- It makes it easier to discover related resources and perform other operations.

👉 Steps to Run the Spring Data REST Project:

1. **Run the Spring Boot Application:**
   - Use the IDE to run the Spring Boot application.
2. **Access the Data:**
   - Open the browser or a tool like Postman and visit the appropriate URLs (e.g., http://localhost:8080/entityName) to interact with the automatically exposed RESTful endpoints for entities.

3. **HATEOAS Links:**
    - In the responses, we will notice **HATEOAS links**, which provide additional URLs to related resources, making it easier to navigate between different entities or actions in the application.

Link for the Code:
https://github.com/navinreddy20/spring6-course/tree/c6690e4f2c70d8f530d70623f13d14ff0ffd7e7d/10%20Spring%20Data%20Rest/10.3%20Running%20The%20Project/spring-data-rest-demo

04 - Update And Delete

👉 **Update Data**:

- HATEOAS (Hypermedia as the Engine of Application State) provides links in response to related resources.
- To update an entity, follow the HATEOAS link provided in the response for the specific resource (e.g., self link).
- Send a **PUT** request to the link's URL and include the updated data in the request body. Spring Data REST automatically handles the update operation.

👉 **Delete Data**:

- To delete an entity, follow the HATEOAS link in the response for the resource.
- Send a **DELETE** request to the link's URL (e.g., self link of a resource). The entity will be deleted by Spring Data REST without requiring a controller.

👉 **No Need for Controllers**:

- If your repository is properly defined, Spring Data REST automatically generates endpoints with HATEOAS links. There is no need to manually create controllers, making RESTful interactions streamlined and standardized.

Link for the Code:
https://github.com/navinreddy20/spring6-course/tree/c6690e4f2c70d8f530d70623f13d14ff0ffd7e7d/10%20Spring%20Data%20Rest/10.4%20Update%20And%20Delete/spring-data-rest-demo