Installing DevTools

Add DevTools Dependency: Make sure you have the spring-boot-devtools

dependency in your pom.xml or build.gradle file.

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<optional>true</optional>

</dependency>

After this, set the IntelliJ settings to allow restarting.

Automatic Restart

* Automatically restarts the application whenever files on the
* classpath change. This is faster than a full restart and only
* reloads the changed parts.
* Uses two class loaders: one for the base classes that don't change
* (typically libraries) and another for the classes that do change
* (application code). When a change is detected, only the latter is
* reloaded, making the restart faster.

Useful configurations for Dev-tools

* spring.devtools.restart.enabled=false
* spring.devtools.restart.exclude=static/\*\*,public/\*\*
* spring.devtools.restart.pollInterval = 20
* spring.devtools.restart.quietPeriod=10

Auditing in Spring Boot

Auditing in Spring boot allows you to automatically populate certain fields, such as creation and modification timestamps, as well as the user who created or modified the entity.

Steps to Add Auditing

1. Create Auditable base Entity using the following annotations

@EntityListeners(AuditingEntityListener.class). This way you will get access to

@CreatedBy, @CreatedDate, @LastModifiedBy, and @LastModifiedDate annotations.

1. Extend all Entities from the Superclass.
2. Create a class and Implement the AuditorAware interface to provide

the current authenticated user from Spring Security.

1. Enable JPA Auditing by adding the @EnableJpaAuditing annotation to

a configuration class.

1. Pass the reference of the AuditorAware class bean to the

@EnableJpaAuditing interface.

RestClient

* The RestClient is a synchronous HTTP client that offers a modern, fluent API. It offers an abstraction over HTTP libraries that allows for convenient conversion from a Java object to an HTTP request, and the creation of objects from an HTTO response.

Building RestClient

* RestClient restClient = RestClient.builder().baseUrl(BASE\_URL).defaultHeader(HttpHeaders.AUTHORIZATION,encodeBasic(properties.getUsername(),properties.getPassword())).build();