Agenda

* Unit Testing Spring based Application
* Spring Security (Token based authentication and Authorization)

Day 5 Revisit

Spring Boot Annotations

@Autowired

@Service

@RestController

@Component

@Bean

@PathVariable and @PathParam

@RequestBody and @ResponseBody

Using Profiles – Dev, Test, Prod

API Documentation using Swagger (It’s a simplest way of understanding all the end points of the web services)

It’s also a simplest way of testing all the end points visually.

It’s a documentation of end points URI, list of parameters with it’s type. Expected response details.

@EnableSwagger2 --- SpringBoot version < 1.5

@OpenAPIDocumentation -- SpringBoot version > 1.5 (Swagger3)

Swagger, Swagger UI. (Two dependencies)

This will automatically generate two end points

1. V1/api-docs
2. Swagger-ui.html

To Configure a Spring Boot based application

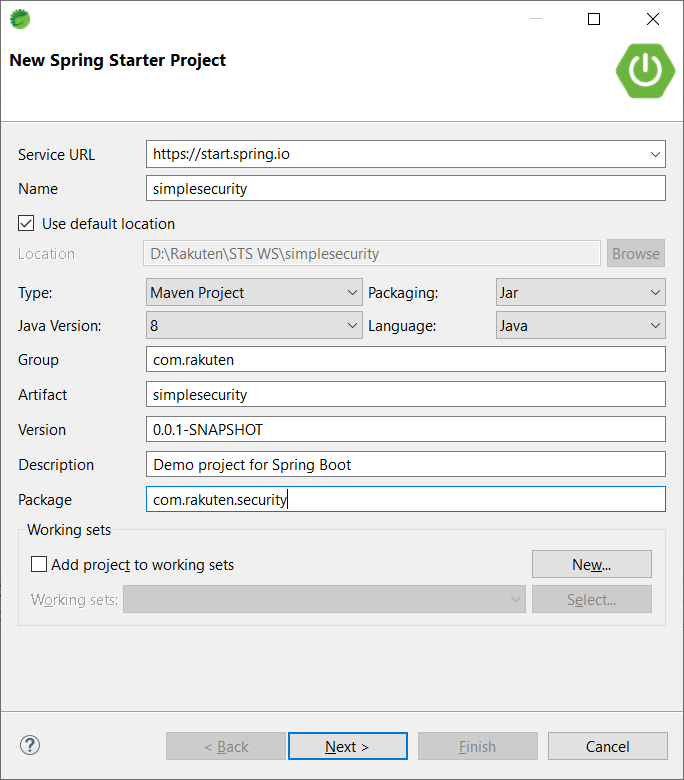
1. Adding the dependency in pom.xml
2. Adding necessary Annotations to the POJO class
3. Configuring it in application.properties (text based file, contains the data in key, value pairs) or in the java class or in application.yml

Adding dependency in the pom.xml

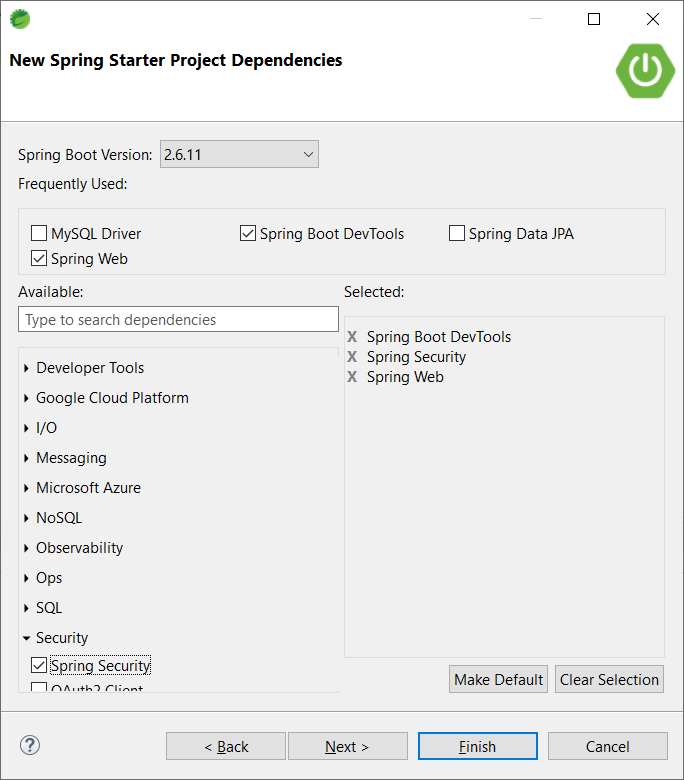
1. Internal Dependency (It will be available in official Spring site--- add it using start.spring.io)
2. External Dependency (Need to add it from mavenrepository.org )

Example for Spring Security

1. Create a Spring Starter project in STS (File 🡪 New 🡪 Spring Starter Project)

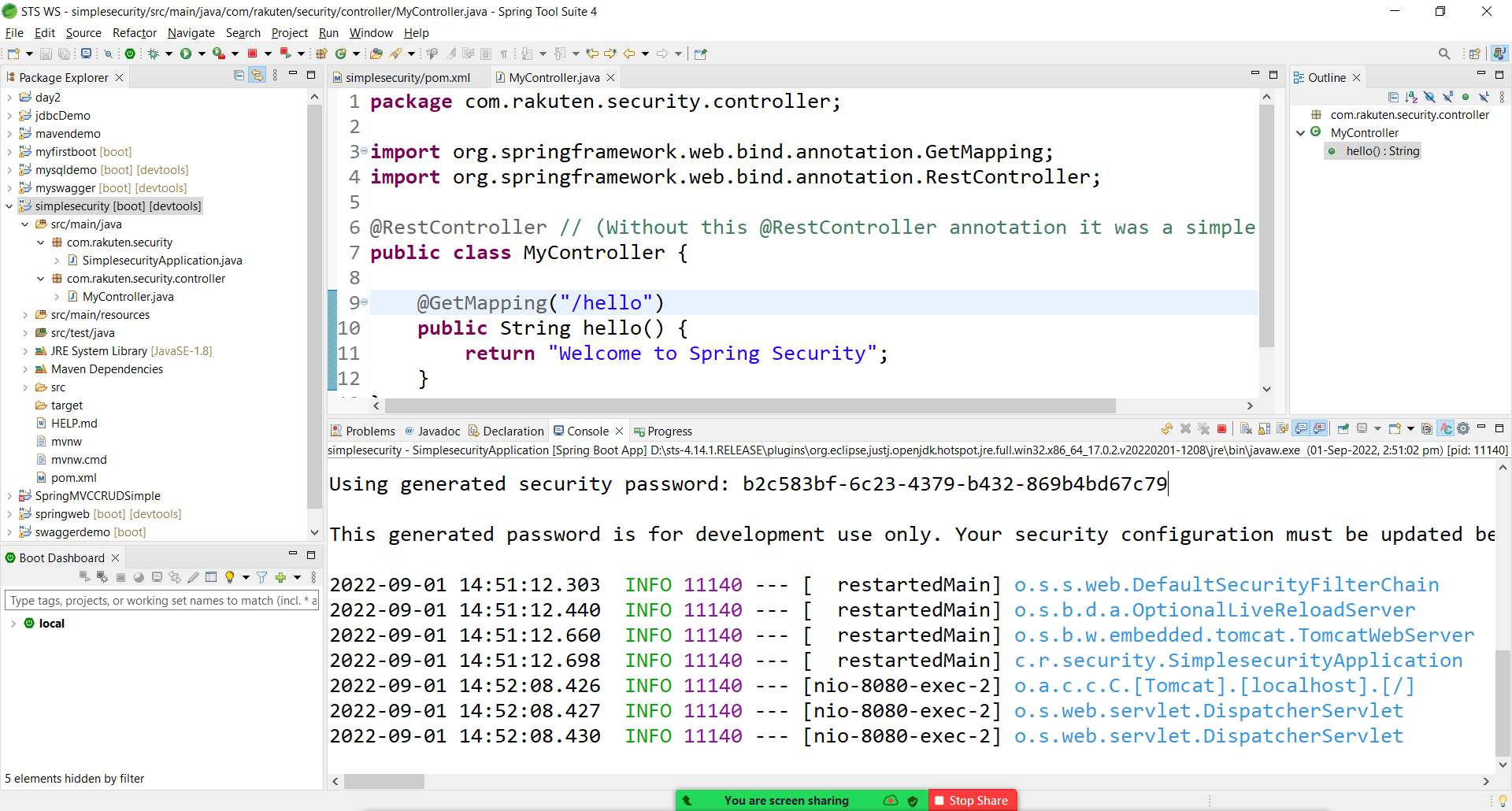


1. Add the necessary dependencies (spring web, dev tools, spring security)



Adding Spring security dependency automatically adds two ends points

1. /login (Responsive Web Page)
2. /logout (Responsive Web Page)



Default user name is : user

Basic (simple) Security module will automatically generate a random password and display it in the console.

Authentication & Authorization

Authentication – checking valid credentials (if the user is a registered user or not)

Authorization – Checking the accessible level of the application (Role based authorization)

(Checking accessibility of a user based on the user role)

Token Based Authentication & Authorization

Spring Boot application will automatically generate a token by validating the user name and password.

JWT – JSON Web Token

<https://jwt.io/>

JWT security module uses some algorithm. (Hashing Algorithm)

1. HSA ( 256,384, 512)
2. RSA ( 256,384, 512)
3. ESA ( 256,384, 512)
4. PSA ( 256,384, 512)

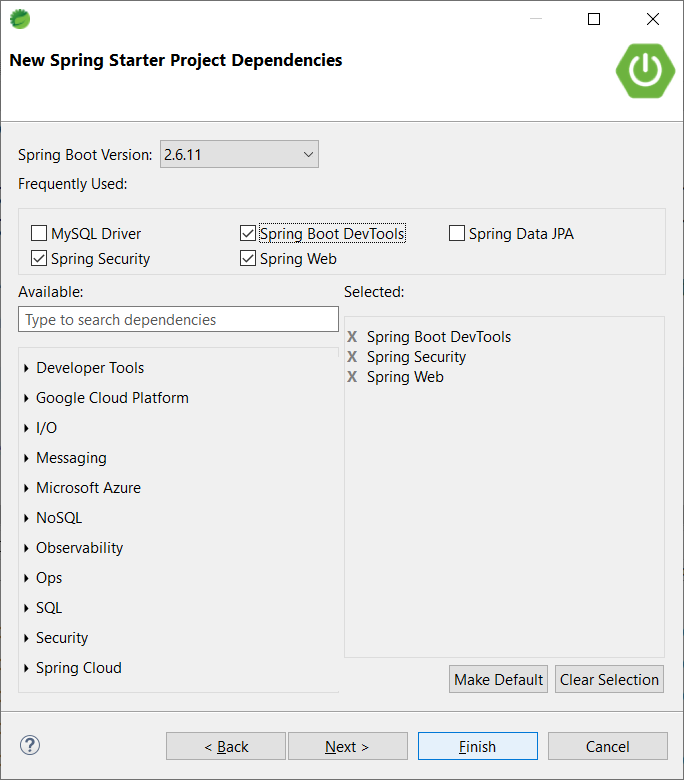
JWT Parts

1. Header
2. Pay Load (Detail)
3. Signature

<https://github.com/syskantechnosoft/JWTAuthwithoutJPA>

<https://www.javainuse.com/spring/boot-jwt>





<https://www.codejava.net/frameworks/spring-boot/spring-security-jwt-authentication-tutorial>

Role based Auth

<https://github.com/syskantechnosoft/role-based-auth-jwt/tree/main/SpringRoleBasedsecurity-jwt-authentication>

Testing the Web Services using Spring Boot

JUNIT & Mockito

Junit – Unit Testing framework for Java. @Test, @Before , @After, @BeforeAll, @AfterAll , @Ignore

TDD – Test Driven Development (Write the Test cases/ Testing code first before even starting the application code)

<https://junit.org/junit5/docs/current/user-guide/>

BDD – Behavior Driven Development ( Gherkhin, Cucumber )

Customer API – Create Customer, FindAll(), findById(),update(),deleteById(), create customers(List<Customer> customer), deleteCustomers(List<Integer> ids)

MySQL Database