**Create authentication service that returns JWT**

In pom.xml:

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

Run:

./mvnw clean install

Create:

src/main/java/com/example/springresthandson/AuthRequest.java

package com.example.springresthandson;

public class AuthRequest {

private String username;

private String password;

// Getters and setters

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

Create:

src/main/java/com/example/springresthandson/JwtUtil.java

package com.example.springresthandson;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import java.security.Key;

import java.util.Date;

@Component

public class JwtUtil {

private final Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256);

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuer("spring-rest-handson")

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 3600000)) // 1 hour

.signWith(key)

.compact();

}

}

Create:

src/main/java/com/example/springresthandson/AuthController.java

package com.example.springresthandson;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/api/auth")

public class AuthController {

@Autowired

private JwtUtil jwtUtil;

@PostMapping("/login")

public ResponseEntity<?> login(@RequestBody AuthRequest authRequest) {

// Hardcoded authentication for learning

if ("user".equals(authRequest.getUsername()) && "password".equals(authRequest.getPassword())) {

String token = jwtUtil.generateToken(authRequest.getUsername());

return ResponseEntity.ok(token);

} else {

return ResponseEntity.status(401).body("Invalid credentials");

}

}

}

src/main/resources/application.properties

spring.security.user.name=

spring.security.user.password=

package com.example.springresthandson;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth.anyRequest().permitAll());

return http.build();

}

}

Run:

./mvnw spring-boot:run

{

"username": "user",

"password": "password"

}

Output:

eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaXNzIjoic3ByaW5nLXJlc3QtaGFuZHNvbiIsImlhdCI6MTY5OTA2Mzc2NCwiZXhwIjoxNjk5MDY3MzY0fQ.cABx7WCl\_OZ5a2lU5QoG9c63ChCRrYZf0pml\_EaUWqU

{

"username": "wrong",

"password": "wrong"

}

Output:

Invalid credentials