How to Implement JWT in Spring Boot::

1. Add Dependency

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.thymeleaf.extras</groupId>  
 <artifactId>thymeleaf-extras-springsecurity6</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-mongodb</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
</dependency>  
<dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-test</artifactId>  
 <scope>test</scope>  
</dependency>  
<!-- https://mvnrepository.com/artifact/org.springframework.security/spring-security-jwt -->  
<dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-jwt</artifactId>  
 <version>1.0.10.RELEASE</version>  
</dependency>  
<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>

2.Define User Class and Role

package com.example.Task.Management.model;  
  
import org.springframework.data.annotation.Id;  
import org.springframework.data.mongodb.core.mapping.Document;  
  
import java.util.Set;  
  
@Document(collection = "users")  
public class User {  
  
 @Id  
 private String id;  
  
 private String username;  
 private String email;  
 private String password;  
 private Set<Role> roles;  
 public User() {  
  
 }  
  
 public User(String id, String username, String email, String password, Set<Role> roles) {  
 this.id = id;  
 this.username = username;  
 this.email = email;  
 this.password = password;  
 this.roles = roles;  
 }  
  
 public String getId() {  
 return id;  
 }  
  
 public void setId(String id) {  
 this.id = id;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public Set<Role> getRoles() {  
 return roles;  
 }  
  
 public void setRoles(Set<Role> roles) {  
 this.roles = roles;  
 }  
}

Role :

package com.example.Task.Management.model;  
  
public enum Role {  
 *ROLE\_USER*,  
 *ROLE\_ADMIN*}

3 . JWTUtils Class

package com.example.Task.Management.model;  
  
import io.jsonwebtoken.\*;  
import io.jsonwebtoken.security.Keys;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.stereotype.Component;  
import java.security.Key;  
import java.util.Date;  
import java.util.function.Function;  
  
@Component // ✅ Ensure Spring can inject this  
public class JwtUtils {  
  
 @Value("${jwt.secret}")  
 private String secretKey;  
  
 @Value("${jwt.expiration}")  
 private long expirationTime;  
  
 private Key getSigningKey() {  
 return Keys.*hmacShaKeyFor*(secretKey.getBytes());  
 }  
  
 public String generateToken(String username) {  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(System.*currentTimeMillis*() + expirationTime))  
 .signWith(getSigningKey(), SignatureAlgorithm.*HS256*)  
 .compact();  
 }  
  
 public String extractUsername(String token) {  
 return extractClaim(token, Claims::getSubject);  
 }  
  
 public boolean validateToken(String token, String username) {  
 return (username.equals(extractUsername(token)) && !isTokenExpired(token));  
 }  
  
 private boolean isTokenExpired(String token) {  
 return extractClaim(token, Claims::getExpiration).before(new Date());  
 }  
  
 private <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {  
 Claims claims = Jwts.*parserBuilder*()  
 .setSigningKey(getSigningKey())  
 .build()  
 .parseClaimsJws(token)  
 .getBody();  
 return claimsResolver.apply(claims);  
 }  
}

4 . Auth Service

package com.example.Task.Management.service;  
  
import com.example.Task.Management.model.JwtUtils;  
import com.example.Task.Management.model.Role;  
import com.example.Task.Management.model.User;  
import com.example.Task.Management.repository.UserRepository;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
import java.util.HashSet;  
import java.util.Set;  
  
@Service  
public class AuthService {  
  
 private final UserRepository userRepository;  
 private final JwtUtils jwtUtils;  
 private final PasswordEncoder passwordEncoder = new BCryptPasswordEncoder();  
  
 public AuthService(UserRepository userRepository, JwtUtils jwtUtils) {  
 this.userRepository = userRepository;  
 this.jwtUtils = jwtUtils;  
 }  
  
 public String register(User user) {  
 if (userRepository.findByEmail(user.getEmail()).isPresent()) {  
 throw new RuntimeException("Email already in use");  
 }  
  
 user.setPassword(passwordEncoder.encode(user.getPassword()));  
 user.setRoles(new HashSet<>(Set.*of*(Role.*ROLE\_USER*))); // Default role  
 userRepository.save(user);  
  
 return "User registered successfully!";  
 }  
  
 public String login(String email, String password) {  
 User user = userRepository.findByEmail(email)  
 .orElseThrow(() -> new RuntimeException("User not found"));  
  
 if (!passwordEncoder.matches(password, user.getPassword())) {  
 throw new RuntimeException("Invalid password");  
 }  
  
 return jwtUtils.generateToken(user.getEmail()); // Token generated with email  
 }  
  
}

5.Auth Controller

package com.example.Task.Management.controller;  
  
import com.example.Task.Management.model.User;  
import com.example.Task.Management.service.AuthService;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
@RequestMapping("/auth")  
public class AuthController {  
  
 private final AuthService authService;  
  
 public AuthController(AuthService authService) {  
 this.authService = authService;  
 }  
  
 @PostMapping("/register")  
 public String register(@RequestBody User user) {  
 return authService.register(user);  
 }  
  
 @PostMapping("/login")  
 public String login(@RequestParam String email, @RequestParam String password) {  
 return authService.login(email, password);  
 }  
}

7. Spring Security !

package com.example.Task.Management.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.web.SecurityFilterChain;  
  
@Configuration  
public class SecurityConfig {  
  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
 http  
 .csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/auth/\*\*").permitAll()  
 .requestMatchers("/admin/\*\*").hasRole("ADMIN")  
 .requestMatchers("/user/\*\*").hasRole("USER")  
 .anyRequest().authenticated()  
 )  
 .formLogin(login -> login.disable());  
  
 return http.build();  
 }  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {  
 return config.getAuthenticationManager();  
 }  
}

8.Test Apis

<http://localhost:8080/auth/register>

{

"username": "test",

"email": "test@example.com",

"password": "password"

}

Login :http://locahost:8080/auth/login

Form Data

email :test@example.com

password:password