**Create authentication service that returns JWT**   
**SecurityConfig**

**package** com.cognizant.ormlearn.config;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

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

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

**import** org.springframework.security.config.http.SessionCreationPolicy;

**import** org.springframework.security.core.userdetails.User;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.core.userdetails.UserDetailsService;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

**import** org.springframework.security.crypto.password.PasswordEncoder;

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig **extends** WebSecurityConfigurerAdapter {

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.antMatchers("/h2-console/\*\*").permitAll()

.anyRequest().authenticated()

.and()

.headers().frameOptions().disable()

.and()

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.***STATELESS***);

}

@Override

@Bean

**public** AuthenticationManager authenticationManagerBean() **throws** Exception {

**return** **super**.authenticationManagerBean();

}

@Bean

**public** UserDetailsService userDetailsService() {

UserDetails user = User.*builder*()

.username("user")

.password(passwordEncoder().encode("pwd"))

.roles("USER")

.build();

UserDetails admin = User.*builder*()

.username("admin")

.password(passwordEncoder().encode("admin123"))

.roles("ADMIN")

.build();

**return** **new** InMemoryUserDetailsManager(user, admin);

}

@Bean

**public** PasswordEncoder passwordEncoder() {

**return** **new** BCryptPasswordEncoder();

}

}

**AuthenticationController**

**package** com.cognizant.ormlearn.controller;

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

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.authentication.BadCredentialsException;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

**import** org.springframework.security.core.Authentication;

**import** org.springframework.security.core.userdetails.UserDetails;

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

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

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

**import** com.cognizant.ormlearn.service.JwtService;

**import** com.cognizant.ormlearn.model.AuthResponse;

**import** java.util.Base64;

**import** java.util.HashMap;

**import** java.util.Map;

@RestController

**public** **class** AuthenticationController {

@Autowired

**private** AuthenticationManager authenticationManager;

@Autowired

**private** JwtService jwtService;

@PostMapping("/authenticate")

**public** ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authorization) {

**try** {

String[] credentials = decodeBasicAuth(authorization);

String username = credentials[0];

String password = credentials[1];

Authentication authentication = authenticationManager.authenticate(

**new** UsernamePasswordAuthenticationToken(username, password)

);

UserDetails userDetails = (UserDetails) authentication.getPrincipal();

String token = jwtService.generateToken(userDetails);

Map<String, String> response = **new** HashMap<>();

response.put("token", token);

**return** ResponseEntity.*ok*(response);

} **catch** (BadCredentialsException e) {

Map<String, String> errorResponse = **new** HashMap<>();

errorResponse.put("error", "Invalid credentials");

**return** ResponseEntity.*status*(401).body(errorResponse);

} **catch** (Exception e) {

Map<String, String> errorResponse = **new** HashMap<>();

errorResponse.put("error", "Authentication failed: " + e.getMessage());

**return** ResponseEntity.*status*(401).body(errorResponse);

}

}

**private** String[] decodeBasicAuth(String authorization) {

**if** (authorization == **null** || !authorization.startsWith("Basic ")) {

**throw** **new** IllegalArgumentException("Invalid authorization header format");

}

String base64Credentials = authorization.substring("Basic ".length());

String credentials = **new** String(Base64.*getDecoder*().decode(base64Credentials));

String[] parts = credentials.split(":", 2);

**if** (parts.length != 2) {

**throw** **new** IllegalArgumentException("Invalid credentials format");

}

**return** parts;

}

}

**TestController**

**package** com.cognizant.ormlearn.controller;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.security.core.Authentication;

**import** org.springframework.security.core.context.SecurityContextHolder;

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

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

**import** java.util.HashMap;

**import** java.util.Map;

@RestController

**public** **class** TestController {

@GetMapping("/test")

**public** ResponseEntity<?> test() {

Authentication authentication = SecurityContextHolder.*getContext*().getAuthentication();

Map<String, Object> response = **new** HashMap<>();

response.put("message", "JWT Authentication successful!");

response.put("user", authentication.getName());

response.put("authorities", authentication.getAuthorities());

**return** ResponseEntity.*ok*(response);

}

@GetMapping("/public")

**public** ResponseEntity<?> publicEndpoint() {

Map<String, String> response = **new** HashMap<>();

response.put("message", "This is a public endpoint");

**return** ResponseEntity.*ok*(response);

}

}

**JwtAuthenticationFilter**

**package** com.cognizant.ormlearn.filter;

**import** com.cognizant.ormlearn.service.JwtService;

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

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

**import** org.springframework.security.core.context.SecurityContextHolder;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.core.userdetails.UserDetailsService;

**import** org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

**import** org.springframework.stereotype.Component;

**import** org.springframework.web.filter.OncePerRequestFilter;

**import** javax.servlet.FilterChain;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** java.io.IOException;

@Component

**public** **class** JwtAuthenticationFilter **extends** OncePerRequestFilter {

@Autowired

**private** JwtService jwtService;

@Autowired

**private** UserDetailsService userDetailsService;

@Override

**protected** **void** doFilterInternal(HttpServletRequest request, HttpServletResponse response,

FilterChain filterChain) **throws** ServletException, IOException {

**final** String authorizationHeader = request.getHeader("Authorization");

String username = **null**;

String jwt = **null**;

**if** (authorizationHeader != **null** && authorizationHeader.startsWith("Bearer ")) {

jwt = authorizationHeader.substring(7);

**try** {

username = jwtService.extractUsername(jwt);

} **catch** (Exception e) {

logger.error("Unable to get JWT Token or JWT Token has expired");

}

}

**if** (username != **null** && SecurityContextHolder.*getContext*().getAuthentication() == **null**) {

UserDetails userDetails = **this**.userDetailsService.loadUserByUsername(username);

**if** (jwtService.validateToken(jwt, userDetails)) {

UsernamePasswordAuthenticationToken authenticationToken = **new** UsernamePasswordAuthenticationToken(userDetails, **null**, userDetails.getAuthorities());

authenticationToken.setDetails(**new** WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.*getContext*().setAuthentication(authenticationToken);

}

}

filterChain.doFilter(request, response);

}

}

**AuthResponse**

**package** com.cognizant.ormlearn.model;

**public** **class** AuthResponse {

**private** String token;

**private** String type = "Bearer";

**private** String username;

**public** AuthResponse() {}

**public** AuthResponse(String token) {

**this**.token = token;

}

**public** AuthResponse(String token, String username) {

**this**.token = token;

**this**.username = username;

}

**public** String getToken() {

**return** token;

}

**public** **void** setToken(String token) {

**this**.token = token;

}

**public** String getType() {

**return** type;

}

**public** **void** setType(String type) {

**this**.type = type;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

}

**JwtService**

**package** com.cognizant.ormlearn.service;

**import** io.jsonwebtoken.Claims;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.stereotype.Service;

**import** java.util.Date;

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.function.Function;

@Service

**public** **class** JwtService {

**private** String SECRET\_KEY = "mySecretKey123456789012345678901234567890"; // Use a longer key in production

**private** **int** JWT\_EXPIRATION = 1200000; // 20 minutes in milliseconds

**public** String generateToken(UserDetails userDetails) {

Map<String, Object> claims = **new** HashMap<>();

claims.put("roles", userDetails.getAuthorities());

**return** createToken(claims, userDetails.getUsername());

}

**private** String createToken(Map<String, Object> claims, String subject) {

**return** Jwts.*builder*()

.setClaims(claims)

.setSubject(subject)

.setIssuedAt(**new** Date(System.*currentTimeMillis*()))

.setExpiration(**new** Date(System.*currentTimeMillis*() + JWT\_EXPIRATION))

.signWith(SignatureAlgorithm.***HS256***, SECRET\_KEY)

.compact();

}

**public** String extractUsername(String token) {

**return** extractClaim(token, Claims::getSubject);

}

**public** Date extractExpiration(String token) {

**return** extractClaim(token, Claims::getExpiration);

}

**public** <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {

**final** Claims claims = extractAllClaims(token);

**return** claimsResolver.apply(claims);

}

**private** Claims extractAllClaims(String token) {

**return** Jwts.*parser*().setSigningKey(SECRET\_KEY).parseClaimsJws(token).getBody();

}

**private** Boolean isTokenExpired(String token) {

**return** extractExpiration(token).before(**new** Date());

}

**public** Boolean validateToken(String token, UserDetails userDetails) {

**final** String username = extractUsername(token);

**return** (username.equals(userDetails.getUsername()) && !isTokenExpired(token));

}

}

**application.properties**

# Server Configuration

server.port=8090

# Database Configuration

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

# JPA Configuration

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

# H2 Console (for development)

spring.h2.console.enabled=true

# =============================================================================

// FILE 3: src/main/java/com/cognizant/ormlearn/OrmLearnApplication.java

// =============================================================================

package com.cognizant.ormlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OrmLearnApplication {

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

}

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<parent>

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

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.0</version>

<relativePath/>

</parent>

<properties>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- Spring Boot Web Starter -->

<dependency>

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

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Spring Boot Security Starter -->

<dependency>

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

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<!-- Spring Boot JPA Starter -->

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- H2 Database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- JWT Library -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<!-- Spring Boot Test Starter -->

<dependency>

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

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**Output:**

