WEEK -4

Create authentication service that returns JWT

Nipuna

Superset id:6432842

As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.  
  
Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.  
  
**Request**

curl -s -u user:pwd http://localhost:8090/authenticate

**Response**

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}

This can be incorporated as three major steps:

* Create authentication controller and configure it in SecurityConfig
* Read Authorization header and decode the username and password
* Generate token based on the user retrieved in the previous step

Let incorporate the above as separate hands on exercises.

Solution:As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.

**SecurityConfig.java:**

package com.cognizant.spring\_learn.config;  
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 securityFilterChain(HttpSecurity http) throws Exception {  
        http  
                .csrf(csrf -> csrf.disable())  
                .authorizeHttpRequests(auth -> auth  
                        .requestMatchers("/authenticate").permitAll()  
                        .anyRequest().authenticated()  
                )  
                .httpBasic();  
  
        return http.build();  
    }  
}

**AuthenticationController.java:**

package com.cognizant.spring\_learn.controller;  
  
import com.cognizant.spring\_learn.util.JwtUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
import jakarta.servlet.http.HttpServletRequest;  
import java.util.Base64;  
@RestController  
public class AuthenticationController {  
    @Autowired  
    private JwtUtil jwtUtil;  
    @RequestMapping(value = "/authenticate", method = RequestMethod.*GET*)  
    public String authenticate(HttpServletRequest request) {  
        String authHeader = request.getHeader("Authorization");  
        if (authHeader != null && authHeader.startsWith("Basic ")) {  
            String base64Credentials = authHeader.substring("Basic ".length());  
            byte[] decoded = Base64.*getDecoder*().decode(base64Credentials);  
            String credentials = new String(decoded);  
            String[] values = credentials.split(":", 2);  
            if ("user".equals(values[0]) && "pwd".equals(values[1])) {  
                String token = jwtUtil.generateToken(values[0]);  
                return "{\"token\":\"" + token + "\"}";  
            }  
        }  
        throw new RuntimeException("Invalid credentials");  
    }  
}

**JwtUtil.java:**

package com.cognizant.spring\_learn.util;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import org.springframework.stereotype.Component;  
import java.util.Base64;  
import java.util.Date;  
@Component  
public class JwtUtil {  
    private final String secretKey = Base64.*getEncoder*().encodeToString("secret-key".getBytes());  
    public String generateToken(String username) {  
        return Jwts.*builder*()  
                .setSubject(username)  
                .setIssuedAt(new Date(System.*currentTimeMillis*()))  
                .setExpiration(new Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60 \* 10)) // 10 hours  
                .signWith(SignatureAlgorithm.*HS256*, Base64.*getDecoder*().decode(secretKey))  
                .compact();  
    }  
}

**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> <https://maven.apache.org/xsd/maven-4.0.0.xsd>">  
    <modelVersion>4.0.0</modelVersion>  
    <parent>  
       <groupId>org.springframework.boot</groupId>  
       <artifactId>spring-boot-starter-parent</artifactId>  
       <version>3.5.3</version>  
       <relativePath/> <!-- lookup parent from repository -->  
    </parent>  
    <groupId>com.cognizant</groupId>  
    <artifactId>spring-learn</artifactId>  
    <version>0.0.1-SNAPSHOT</version>  
    <name>spring-learn</name>  
    <description>Demo project for Spring Boot</description>  
    <url/>  
    <licenses>  
       <license/>  
    </licenses>  
    <developers>  
       <developer/>  
    </developers>  
    <scm>  
       <connection/>  
       <developerConnection/>  
       <tag/>  
       <url/>  
    </scm>  
    <properties>  
       <java.version>17</java.version>  
    </properties>  
    <dependencies>  
       <dependency>  
          <groupId>org.springframework.boot</groupId>  
          <artifactId>spring-boot-starter-web</artifactId>  
       </dependency>  
  
       <dependency>  
          <groupId>org.springframework.boot</groupId>  
          <artifactId>spring-boot-devtools</artifactId>  
          <scope>runtime</scope>  
          <optional>true</optional>  
       </dependency>  
       <!-- Required only for Java 11+ with jjwt 0.9.1 -->  
       <dependency>  
          <groupId>javax.xml.bind</groupId>  
          <artifactId>jaxb-api</artifactId>  
          <version>2.3.1</version>  
       </dependency>  
  
       <dependency>  
          <groupId>org.springframework.boot</groupId>  
          <artifactId>spring-boot-starter-test</artifactId>  
          <scope>test</scope>  
       </dependency>  
       <!-- Spring Security -->  
       <dependency>  
          <groupId>org.springframework.boot</groupId>  
          <artifactId>spring-boot-starter-security</artifactId>  
       </dependency>  
       <dependency>  
          <groupId>jakarta.servlet</groupId>  
          <artifactId>jakarta.servlet-api</artifactId>  
          <version>6.0.0</version>  
          <scope>provided</scope>  
       </dependency>  
       <dependency>  
          <groupId>javax.servlet</groupId>  
          <artifactId>javax.servlet-api</artifactId>  
          <version>4.0.1</version>  
          <scope>provided</scope>  
       </dependency>  
  
  
       <!-- JWT Support -->  
       <dependency>  
          <groupId>io.jsonwebtoken</groupId>  
          <artifactId>jjwt</artifactId>  
          <version>0.9.1</version>  
       </dependency>  
  
    </dependencies>  
  
    <build>  
       <plugins>  
          <plugin>  
             <groupId>org.springframework.boot</groupId>  
             <artifactId>spring-boot-maven-plugin</artifactId>  
          </plugin>  
       </plugins>  
    </build>  
  
</project>

Output:

