**Create authentication service that returns JWT**   
  
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

**Program**

**pom.xml**

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- JWT -->

<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>

</dependencies>

**JwtAuthApplication.java**

package com.cognizant.jwt;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication {

public static void main(String[] args) {

SpringApplication.run(JwtAuthApplication.class, args);

}

}

**MyUserDetailsService.java**

package com.cognizant.jwt.security;

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

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class MyUserDetailsService implements UserDetailsService {

@Override

public UserDetails loadUserByUsername(String username) {

// Hardcoded user. You can replace this with DB check.

if ("user".equals(username)) {

return new User("user", "{noop}pwd", List.of(new SimpleGrantedAuthority("ROLE\_USER")));

}

throw new UsernameNotFoundException("User not found");

}

}

**JwtUtil.java**

package com.cognizant.jwt.util;

import io.jsonwebtoken.\*;

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)

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

.setExpiration(new Date(System.currentTimeMillis() + 10 \* 60 \* 1000)) // 10 min

.signWith(key)

.compact();

}

public Key getKey() {

return key;

}

}

**AuthController.java**

package com.cognizant.jwt.controller;

import com.cognizant.jwt.util.JwtUtil;

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

import org.springframework.http.ResponseEntity;

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

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

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

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

import javax.servlet.http.HttpServletRequest;

import java.util.\*;

@RestController

public class AuthController {

@Autowired

private AuthenticationManager authManager;

@Autowired

private JwtUtil jwtUtil;

@Autowired

private UserDetailsService userDetailsService;

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

String authHeader = request.getHeader("Authorization");

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

return ResponseEntity.status(401).body("Missing or invalid Authorization header");

}

String base64Credentials = authHeader.substring("Basic ".length()).trim();

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

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

try {

Authentication auth = new UsernamePasswordAuthenticationToken(values[0], values[1]);

authManager.authenticate(auth);

UserDetails userDetails = userDetailsService.loadUserByUsername(values[0]);

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

return ResponseEntity.ok(Map.of("token", token));

} catch (Exception e) {

return ResponseEntity.status(403).body("Authentication failed: " + e.getMessage());

}

}

}

**SecurityConfig.java**

package com.cognizant.jwt.security;

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

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

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

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {

return config.getAuthenticationManager();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

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

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

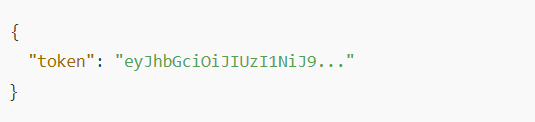
.httpBasic(); // Accepts Basic Auth credentials

return http.build();

}

}

**Output**

****