**Create authentication service that returns JWT  
  
pom.xml**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

**AuthenticationController.java**

package com.cognizant.springlearn.controller;

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

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

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private static final Logger LOGGER = LoggerFactory.getLogger(AuthenticationController.class);

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

LOGGER.info("START");

LOGGER.debug("Authorization Header: {}", authHeader);

String user = getUser(authHeader);

String token = generateJwt(user);

Map<String, String> map = new HashMap<>();

map.put("token", token);

LOGGER.info("END");

return map;

}

private String getUser(String authHeader) {

String encodedCredentials = authHeader.substring("Basic ".length());

byte[] decodedBytes = Base64.getDecoder().decode(encodedCredentials);

String decodedCredentials = new String(decodedBytes);

return decodedCredentials.split(":")[0]; // extract username

}

private String generateJwt(String user) {

JwtBuilder builder = Jwts.builder();

builder.setSubject(user);

builder.setIssuedAt(new Date());

builder.setExpiration(new Date(System.currentTimeMillis() + 20 \* 60 \* 1000)); // 20 mins

builder.signWith(SignatureAlgorithm.HS256, "secretkey");

return builder.compact();

}

}

**SecurityConfig.java**

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable()

.httpBasic().and()

.authorizeRequests()

.antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

.and()

.addFilter(new JwtAuthorizationFilter(authenticationManager()));

}