**5. JWT-HandsOn**

# Create authentication service that returns JWT

# AuthenticationController.java

*package com.cognizant.spring\_learn.controller;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
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 String SECRET\_KEY = "mySecretKey";  
  
 @GetMapping("/authenticate")  
 public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {  
 byte[] decodedAuth = Base64.getDecoder().decode(authHeader.substring("Basic ".length()));  
 String decoded = new String(decodedAuth);  
 String[] credentials = decoded.split(":");  
 String username = credentials[0];  
 String password = credentials[1];  
  
 // Simple check (in real applications, use user service to validate credentials)  
 if ("user".equals(username) && "pwd".equals(password)) {  
 String token = Jwts.builder()  
 .setSubject(username)  
 .setIssuedAt(new Date(System.currentTimeMillis()))  
 .setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 10))  
 .signWith(SignatureAlgorithm.HS256, SECRET\_KEY)  
 .compact();  
  
 Map<String, String> response = new HashMap<>();  
 response.put("token", token);  
 return response;  
 }*

*else {  
 throw new RuntimeException("Invalid Credentials");  
 }  
 }  
}*

# SecurityConfig.java

*package com.cognizant.spring\_learn.config;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
  
@SuppressWarnings("deprecation")  
@Configuration  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http  
 .csrf().disable()  
 .authorizeRequests()  
 .antMatchers("/authenticate").authenticated()  
 .and()  
 .httpBasic();  
 }  
}*

# pom.xml (JWT Dependency)

*<dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt</artifactId>  
 <version>0.9.1</version>  
</dependency>*