**Create authentication service that returns JWT**

1. **Create authentication controller and configure it in SecurityConfig**

**/spring-learn/src/main/java/com/cognizant/spring\_learn/controller/AuthenticationController.java**

**package** com.cognizant.spring\_learn.controller;

**import** java.util.HashMap;

**import** java.util.Map;

**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** com.cognizant.spring\_learn.SpringLearnApplication;

@RestController

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

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

@GetMapping("/authenticate")

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

***LOGGER***.info("START authenticate");

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

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

hashMap.put("token", "");

***LOGGER***.info("END authenticate");

**return** hashMap;

}

}

**/spring-learn/src/main/java/com/cognizant/spring\_learn/security/SecurityConfig.java**

**package** com.cognizant.spring\_learn.security;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**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;

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

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

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

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

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

**import** org.springframework.security.config.Customizer;

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

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig {

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

@Bean

**public** InMemoryUserDetailsManager userDetailsService(PasswordEncoder encoder) {

***LOGGER***.info("Start userDetailsService()");

UserDetails admin = User.*withUsername*("admin")

.password(encoder.encode("pwd"))

.roles("ADMIN")

.build();

UserDetails user = User.*withUsername*("user")

.password(encoder.encode("pwd"))

.roles("USER")

.build();

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

}

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

***LOGGER***.info("Start filterChain()");

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

.authorizeHttpRequests(auth -> auth

.requestMatchers("/countries").hasRole("USER")

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

.anyRequest().authenticated()

)

.httpBasic(Customizer.*withDefaults*());

**return** http.build();

}

@Bean

**public** PasswordEncoder passwordEncoder() {

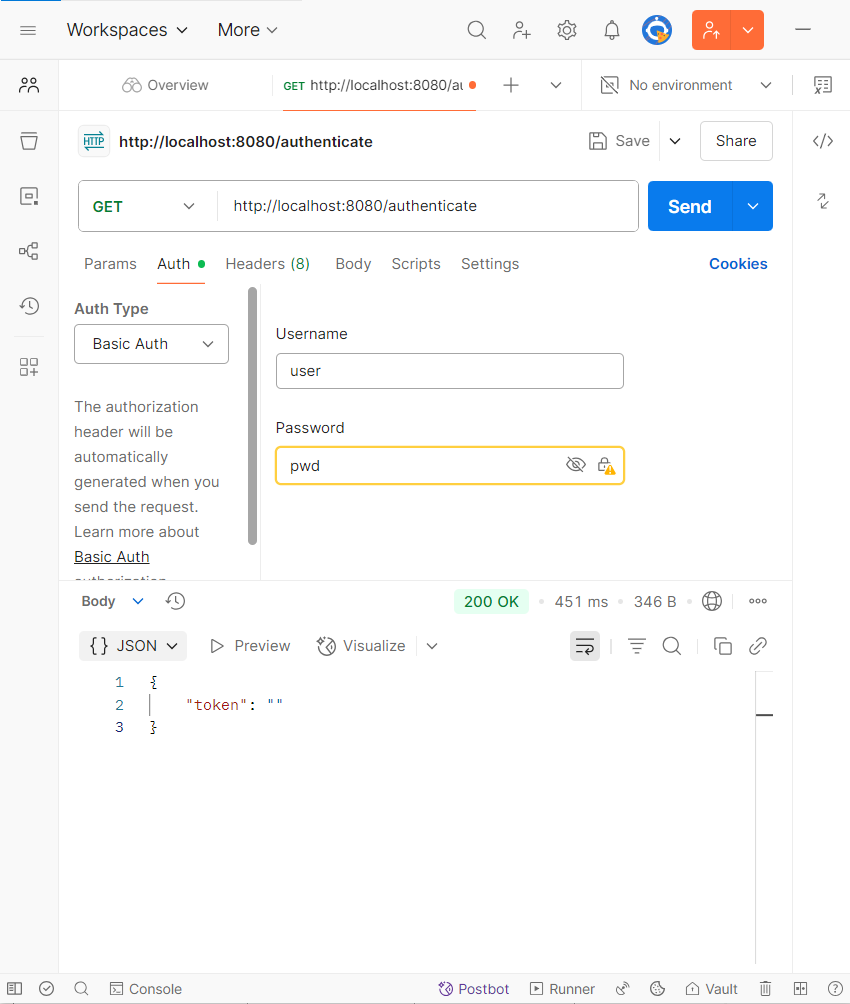
***LOGGER***.info("Start passwordEncoder()");

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

}

}

**Testing** (Postman)



1. **Read Authorization header and decode the username and password**

**/spring-learn/src/main/java/com/cognizant/spring\_learn/controller/AuthenticationController.java**

**package** com.cognizant.spring\_learn.controller;

**import** java.util.Base64;

**import** java.util.HashMap;

**import** java.util.Map;

**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** com.cognizant.spring\_learn.SpringLearnApplication;

@RestController

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

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

@GetMapping("/authenticate")

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

***LOGGER***.info("START authenticate");

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

String username = getUser(authHeader); // Call to the new method

***LOGGER***.debug("Authenticated user: {}", username);

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

hashMap.put("token", "");

***LOGGER***.info("END authenticate");

**return** hashMap;

}

**private** String getUser(String authHeader) {

***LOGGER***.info("Start getUser()");

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

**byte** [] decodedBytes = Base64.*getDecoder*().decode(base64Credentials);

String decodedString = **new** String(decodedBytes);

***LOGGER***.debug("Decoded credentials: {}", decodedString);

String username = decodedString.split(":", 2)[0];

***LOGGER***.debug("Extracted username: {}", username);

***LOGGER***.info("End getUser()");

**return** username;

}

}

**Output** (Logs)

