**Securing RESTful Web Services with Spring Security**

**SecurityConfig.java**

package com.cognizant.spring\_learn.security;

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.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.web.SecurityFilterChain;

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

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig{

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests((authz) -> authz.anyRequest().authenticated())

.httpBasic(*withDefaults*());

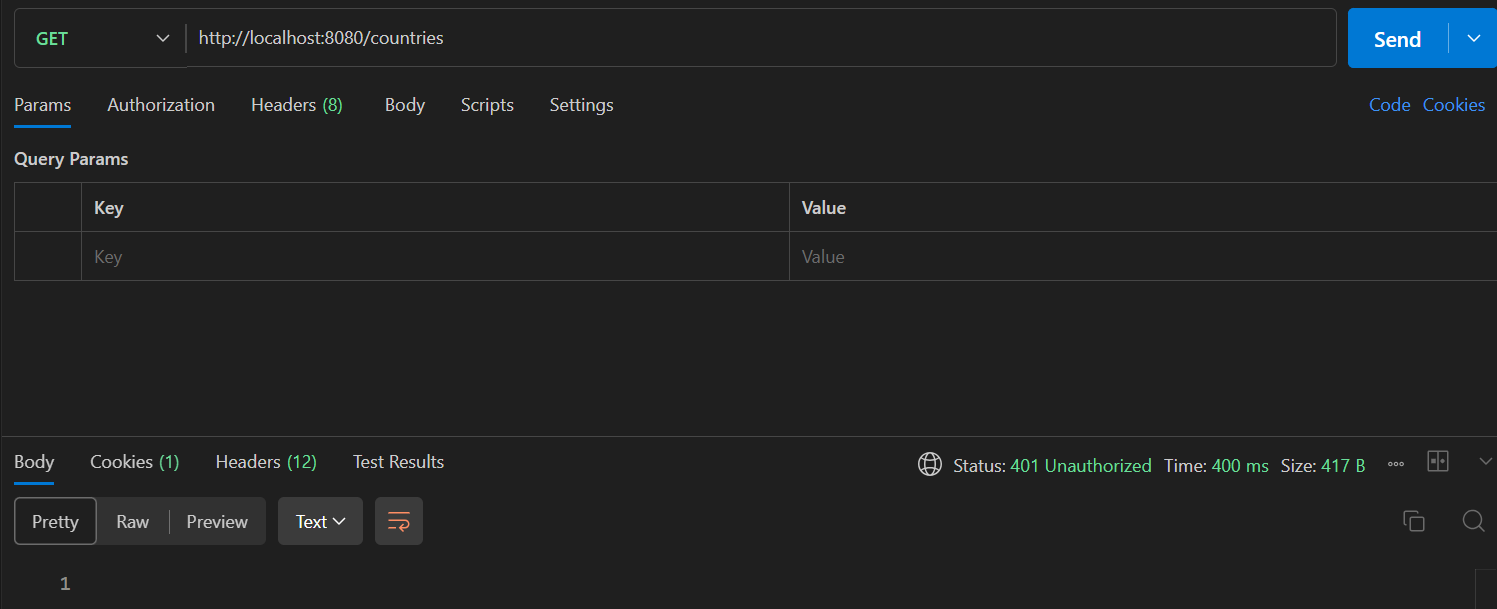
return http.build();

}

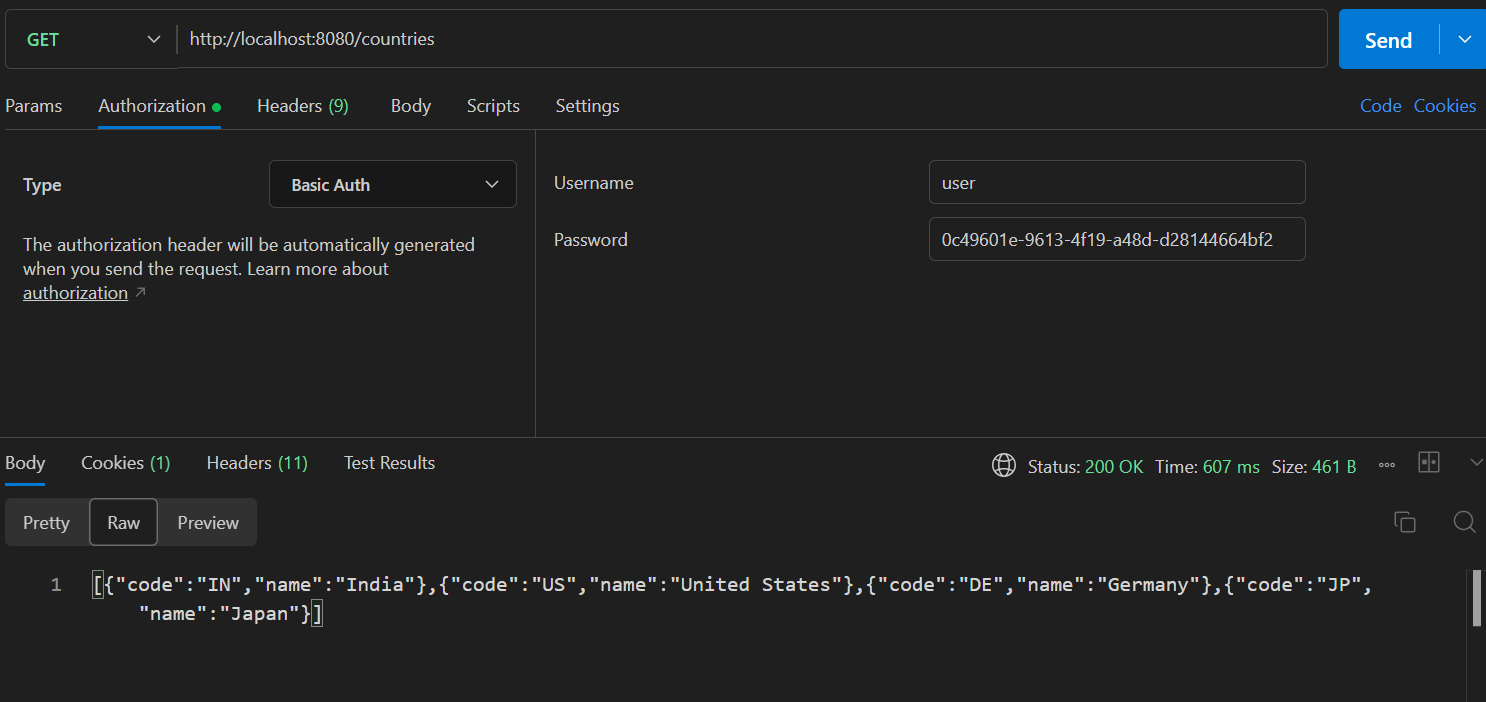
}

**Output:**

Without Authorization:



With Authorization:



**Creating users and roles in Spring 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.core.userdetails.User;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

public class SecurityConfig {

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

*@Bean*

public UserDetailsService userDetailsService(PasswordEncoder encoder) {

InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();

manager.createUser(User.*withUsername*("admin")

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

.roles("ADMIN")

.build());

manager.createUser(User.*withUsername*("user")

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

.roles("USER")

.build());

return manager;

}

*@Bean*

public PasswordEncoder passwordEncoder() {

LOGGER.info("START");

return new BCryptPasswordEncoder();

}

*@Bean*

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

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

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.httpBasic(httpBasic -> {});

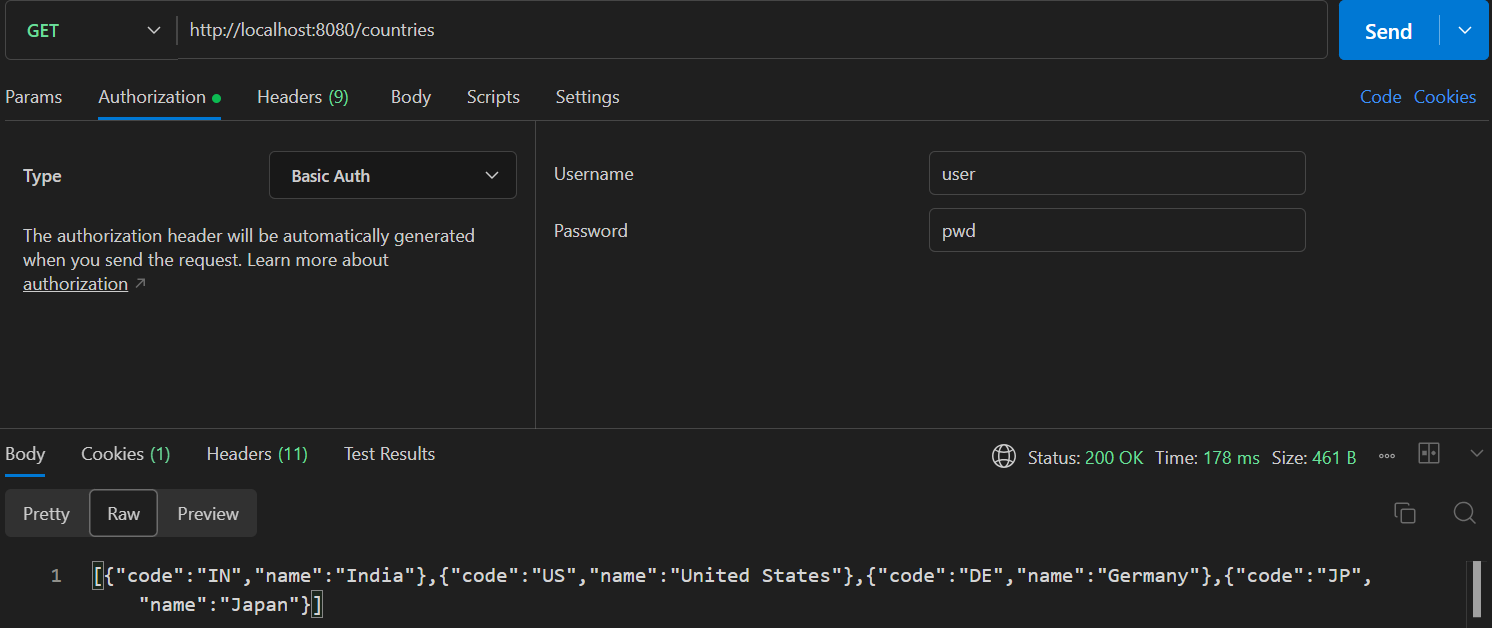
return http.build();

}

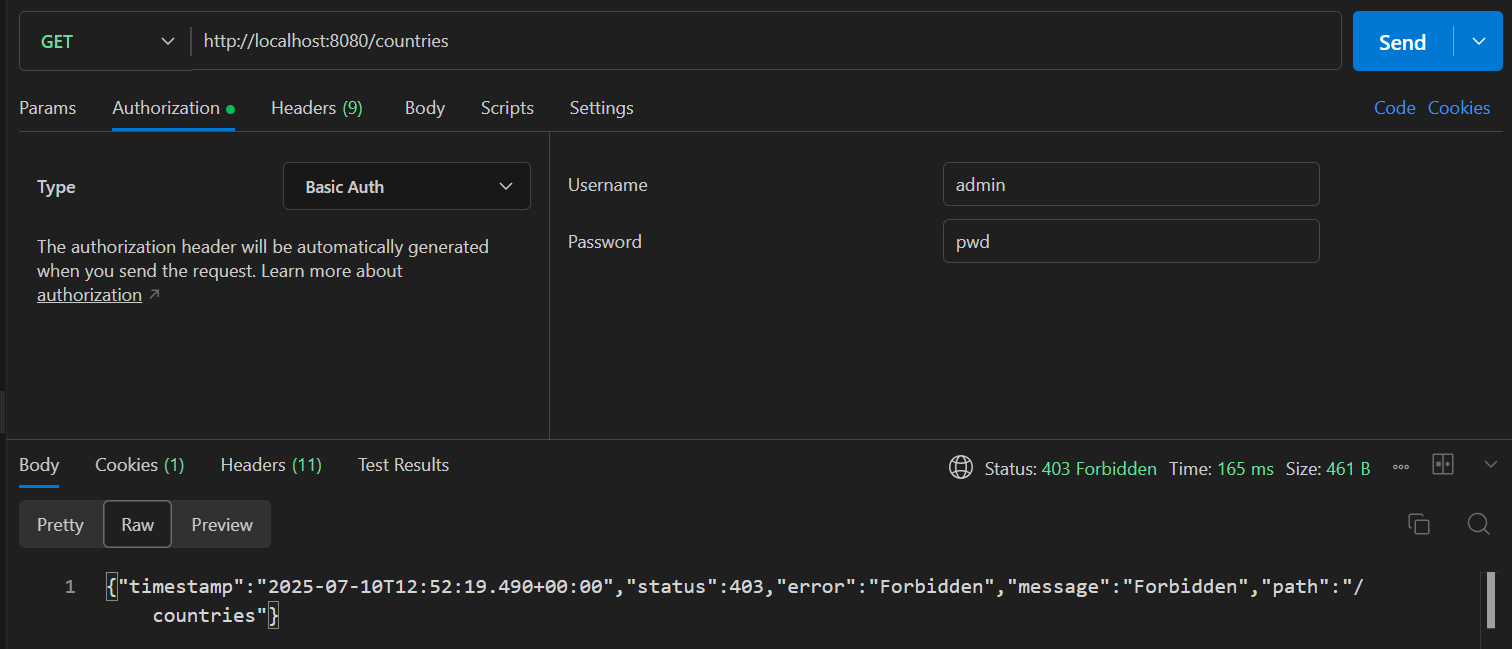
}

**Output:**

User:



Admin:



**Create authentication controller and configure it in SecurityConfig**   
  
**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.RequestMapping;

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

*@RestController*

*@RequestMapping*

public class AuthenticationController {

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

*@GetMapping*("/authenticate")

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

LOGGER.info("START");

LOGGER.debug(authHeader);

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

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

LOGGER.info("END");

return map;

}

}

**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.core.userdetails.User;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

public class SecurityConfig {

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

*@Bean*

public UserDetailsService userDetailsService(PasswordEncoder encoder) {

InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();

manager.createUser(User.*withUsername*("admin")

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

.roles("ADMIN")

.build());

manager.createUser(User.*withUsername*("user")

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

.roles("USER")

.build());

return manager;

}

*@Bean*

public PasswordEncoder passwordEncoder() {

LOGGER.info("START");

return new BCryptPasswordEncoder();

}

*@Bean*

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

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

.authorizeHttpRequests(auth -> auth

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

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

.anyRequest().authenticated()

)

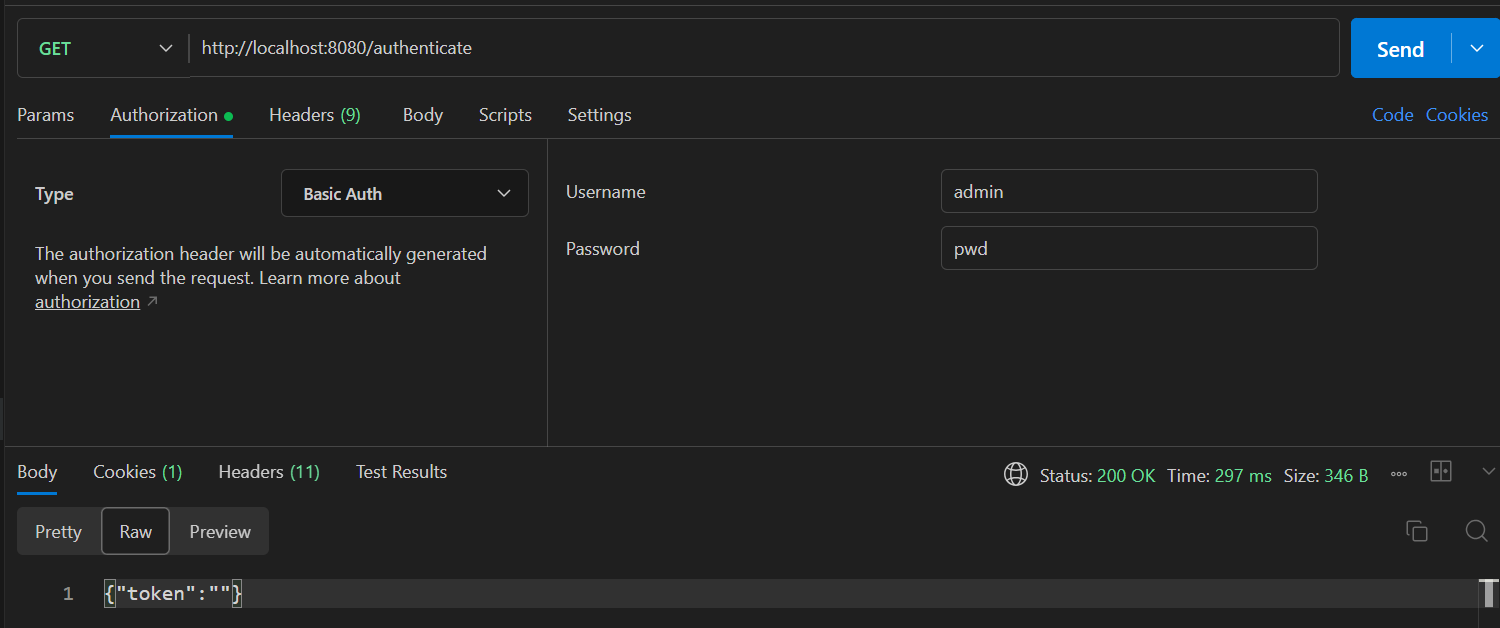
.httpBasic(httpBasic -> {});

return http.build();

}

}

**Output:**



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

**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.RequestMapping;

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

*@RestController*

*@RequestMapping*

public class AuthenticationController {

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

*@GetMapping*("/authenticate")

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

LOGGER.info("START");

LOGGER.debug("Authorization Header: "+authHeader);

String username=getUser(authHeader);

LOGGER.debug("Username: "+username);

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

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

LOGGER.info("END");

return map;

}

private String getUser(String authHeader) {

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

LOGGER.debug("Authorization Header: "+authHeader);

String encodedCredentials=authHeader.substring(6);

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

String decodedCredentials=new String(decodedBytes);

LOGGER.debug("Decoded Credentials: "+decodedCredentials);

int colonIndex=decodedCredentials.indexOf(':');

if (colonIndex != -1) {

String username = decodedCredentials.substring(0, colonIndex);

LOGGER.debug("Extracted Username: "+username);

return username;

} else {

LOGGER.warn("Invalid credentials format: no colon found");

}

}

else {

LOGGER.warn("Authorization header is null or does not start with Basic");

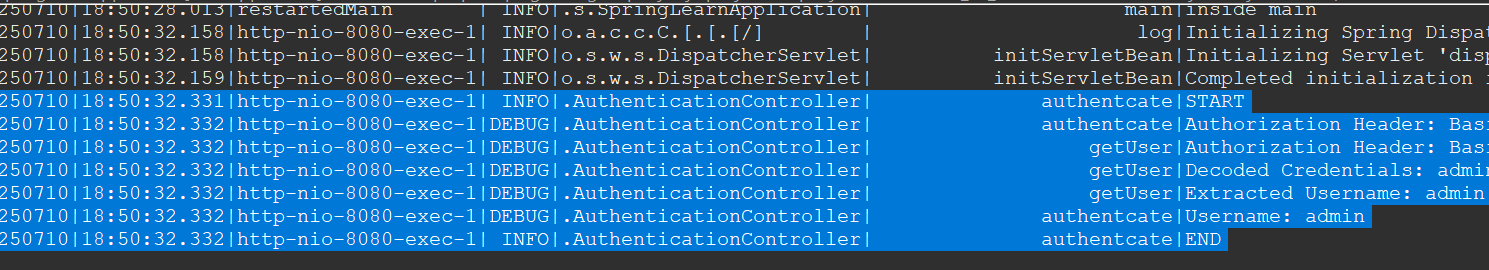
}

return null;

}

}

**Output:**

****

250710|18:50:32.331|http-nio-8080-exec-1| INFO|.AuthenticationController| authentcate|START

250710|18:50:32.332|http-nio-8080-exec-1|DEBUG|.AuthenticationController| authentcate|Authorization Header: Basic YWRtaW46cHdk

250710|18:50:32.332|http-nio-8080-exec-1|DEBUG|.AuthenticationController| getUser|Authorization Header: Basic YWRtaW46cHdk

250710|18:50:32.332|http-nio-8080-exec-1|DEBUG|.AuthenticationController| getUser|Decoded Credentials: admin:pwd

250710|18:50:32.332|http-nio-8080-exec-1|DEBUG|.AuthenticationController| getUser|Extracted Username: admin

250710|18:50:32.332|http-nio-8080-exec-1|DEBUG|.AuthenticationController| authentcate|Username: admin

250710|18:50:32.332|http-nio-8080-exec-1| INFO|.AuthenticationController| authentcate|END

**Generate token based on the user**

**AuthenticationController.java:**

package com.cognizant.spring\_learn.controller;

import java.nio.charset.StandardCharsets;

import java.util.Base64;

import java.util.Date;

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.RequestMapping;

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

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.security.Keys;

import javax.crypto.SecretKey;

*@RestController*

*@RequestMapping*

public class AuthenticationController {

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

private static final SecretKey ***SECRET\_KEY*** = Keys.*hmacShaKeyFor*("myverysecretkeyformyjwtmustbe32bytes".getBytes(StandardCharsets.***UTF\_8***));

*@GetMapping*("/authenticate")

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

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

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

String username = getUser(authHeader);

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

String token = generateJwt(username);

***LOGGER***.debug("Token: {}", token);

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

map.put("token", token);

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

return map;

}

private String getUser(String authHeader) {

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

String encodedCredentials = authHeader.substring(6);

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

String decodedCredentials = new String(decodedBytes, StandardCharsets.***UTF\_8***);

***LOGGER***.debug("Decoded Credentials: {}", decodedCredentials);

int colonIndex = decodedCredentials.indexOf(':');

if (colonIndex != -1) {

String username = decodedCredentials.substring(0, colonIndex);

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

return username;

} else {

***LOGGER***.warn("Invalid credentials format: no colon found");

}

} else {

***LOGGER***.warn("Authorization header is null or does not start with Basic");

}

return null;

}

private String generateJwt(String user) {

return Jwts.*builder*()

.setSubject(user)

.setIssuedAt(new Date())

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

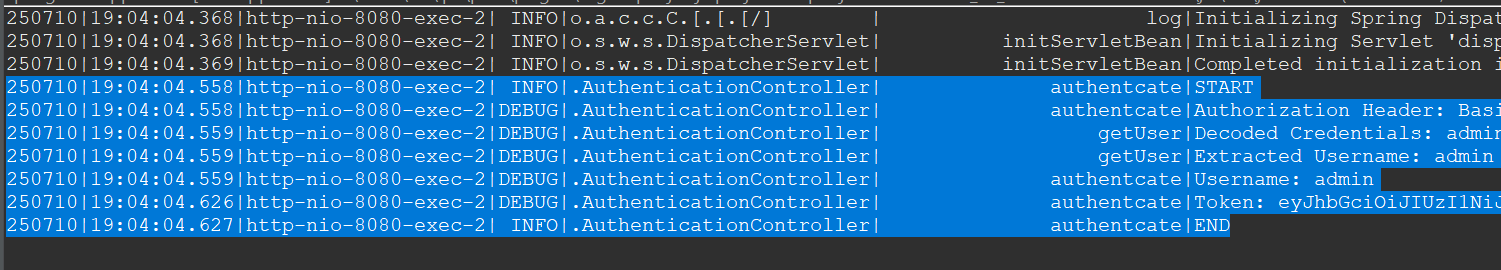
.signWith(***SECRET\_KEY***)

.compact();

}

}

**Logs:**



250710|19:04:04.558|http-nio-8080-exec-2| INFO|.AuthenticationController| authentcate|START

250710|19:04:04.558|http-nio-8080-exec-2|DEBUG|.AuthenticationController| authentcate|Authorization Header: Basic YWRtaW46cHdk

250710|19:04:04.559|http-nio-8080-exec-2|DEBUG|.AuthenticationController| getUser|Decoded Credentials: admin:pwd

250710|19:04:04.559|http-nio-8080-exec-2|DEBUG|.AuthenticationController| getUser|Extracted Username: admin

250710|19:04:04.559|http-nio-8080-exec-2|DEBUG|.AuthenticationController| authentcate|Username: admin

250710|19:04:04.626|http-nio-8080-exec-2|DEBUG|.AuthenticationController| authentcate|Token: eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImlhdCI6MTc1MjE1NDQ0NCwiZXhwIjoxNzUyMTU1NjQ0fQ.7oE\_apSfkK4aXHFf9HzRMLGzzniijLCv1hiSDvHEJWk

250710|19:04:04.627|http-nio-8080-exec-2| INFO|.AuthenticationController| authentcate|END

**Output:**

