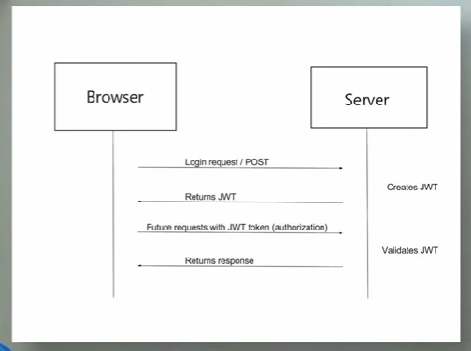
Code Buffer

Implementing JWT (JSON Web Tokens) with Spring Security in Springboot App

* So what basically will be doing here is ,
* we will be creating one API that will authenticate our username and password what we provide to login, based on the credential that we provide to the server then authentication will happen at the server level and JWT token will be created and that JWT token will be passed to the client,



Now , the client is responsible to send that JWT token to every request sent to the server and server will authenticate that JWT token and proceed ahead with the request.

So, this is how JWT Token works.

Q,

ANS.

* Now JWT consist of with 3 parts
* 1. Header
* 2. PayLoad Body
* 3.Signature

Q.

**STEP 1: start.sping.io**

**STEP 2: Generated project with 3 dependencies namely, web, Lombok and Security dependencies**

**STEP 3: Created different packages(config,entity, controller, service )etc**

**First we move to SecurutyConfiguration.java**

**STEP 1 + STEP 2**

**Then move to UserService.java**

**STEP 3 + STEP 4+STEP 5**

**Then again move to SecurutyConfiguration.java**

**STEP 6**

**Then move to HomeController.java**

**STEP 7 (**here whenever we are trying to access this application this message should be display to the user but even before displaying this message to the user our

//request should be authenticated sp spring security will come in picture and it will ask for the credential)

**Then move to HomeController.java**

**STEP 8**

Since we are not encoding or decoding any password in our application lets go to the main application class (JwtCodeBufferApplication.java) and implement this method for no password encoding in our application

**NOTE: upto 12:00 we have just implemented Spring Security.**

Now what we will do is we will be implementing jwt. So wil be adding jwt dependencies and what will be doing is will be creating one more method to authenticate the request and create the JWT Token and from next on will be passing jwt token every time when we have to perform any action to our server

**Then move to pom.xml**

**STEP 9**

And add two dependencies

<!-- jjwt and jaxb-api has been taken for JWT Token -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

</dependency>

**Now, I will create the jwt utility file over herethat will be responsible to have al the behaviour of the the JWT like**

**create a Token ,**

**Validate a Token and all the other methods . So let’s see that**

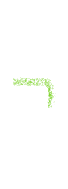
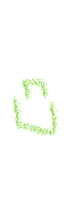
**So 1st created on utility package and inside that package created one class with name JWTUtility**

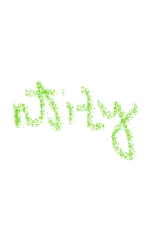
**Then created package with name utility and inside that package created one class with name JWTToken.java**

**So in this file we have created couple of utility methods**

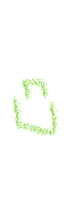
**Now will create one authenticate API which will authenticate the user information username and password which we are sending and on the basis of that it will create the JWT Token for us.**

So now let’s move to **move to HomeController.java again but even before going to the controller we have to create one model( Student) and DTO layer to return the response to the user**

****

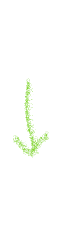
**Student.java**



****



**StudentDTO.java**

****

**HomeController.java**

@PostMapping("/authenticate")

**public** StudentDTO authenticate(@RequestBody Student student)**throws** Exception

{

// Now here i have to authenticate my username and password with the Authentication Manager of Spring Security and if its valid then i have to create jwt token and give return back

// For that will be using jwt utility and Authentication Manageer from the Spring security. So first lets annotate it at the top

**try**

{

authenticationManager.authenticate(

**new** UsernamePasswordAuthenticationToken(

student.getUsername(), student.getPasword()

)

);

}

**catch**(BadCredentialsException e)

{

**throw** **new** Exception("Invalid\_Credential"+e);

}

// Now once the authentication is done let's create the JWT Token

// So first will create the USer Details object from the user name that we have

// So will create the final UserDetails from the spring security

**final** UserDetails userDetails // UserDetails from spring security

=userService.loadUserByUsername(student.getUsername());

**final** String token=

jwtUtility.generateToken(userDetails);

// Once th etokn is created we need to wrap this token into response and send back

**return** **new** StudentDTO(token);

}

Now if you will run and hit the API you will be getting unauthorize request as a response

**{**

**"timestamp": "2022-01-26T11:33:39.961+00:00",**

**"status": 403,**

**"error": "Forbidden",**

**"message": "",**

**"path": "/authenticate"**

**}**

**To make this request as a authorize request for that authentication has to be allowed. I mean this API should be allowed from Spring Security to generate the token without any credentials once we get the JWT token all the other request should be authenticated. So le’s add those information in SecurityConfiguration.java**

// This method has been overriden to authorize the request

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.csrf().disable() // disable the cross reference

.authorizeRequests()

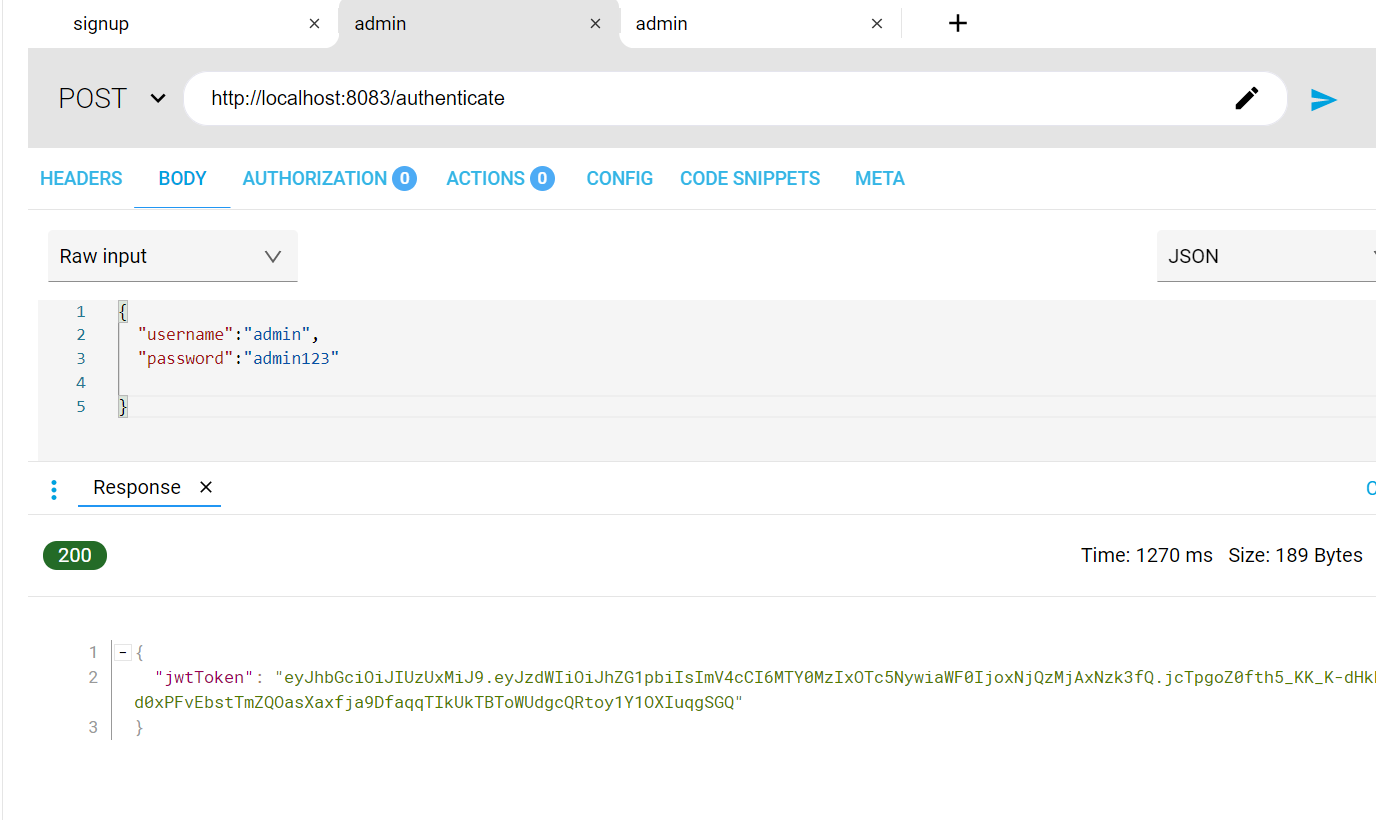
.antMatchers("/authenticate")

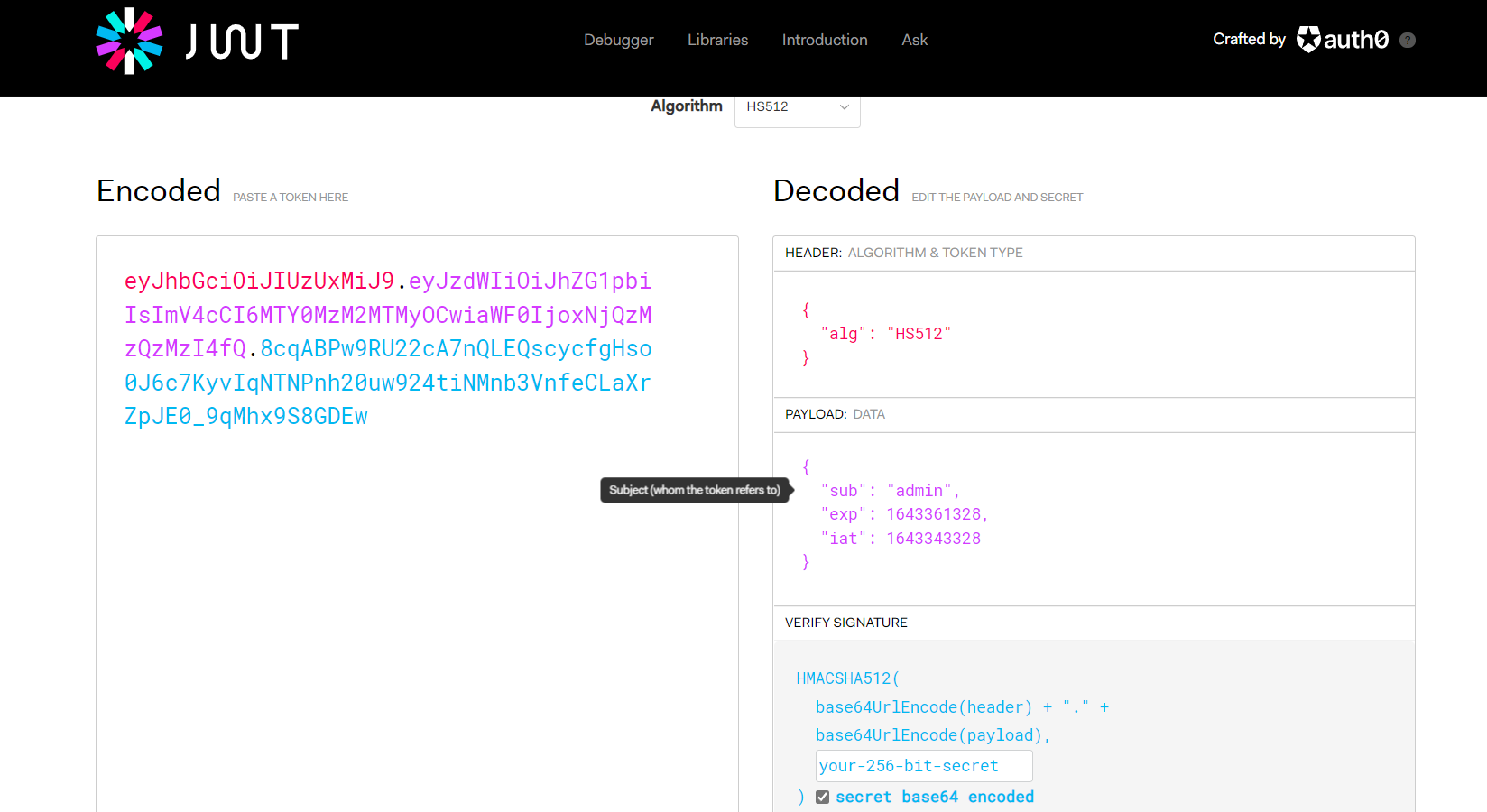
.permitAll()

.anyRequest() // Any Request

.authenticated(); // should be authenticated

}

****

**Now, Once the JWT token is generated to verify the token do switch to the** [**https://jwt.io/**](https://jwt.io/) **and there click on Debugger Menu. After that paste your copird JWT Token**

Now, with every request we have to pass this JWT token over there and our code should be able to understand this token to authenticate the user. So currently if I use directly this then our system won’t be able to understand what we have passed.

So wee need tot add code for that. So what will do will add the **Filter.** So whenever any request is there for each and every request we need to authenticate and will be using **once per request filter from the spring web to add this filter over there.**

**So whenever we try to hit some request that filter will be executed first and then request will be passed to the next filter. So let’s add that**

**So, first will create one package with name filter**



* **An then inside this filter will create one new class with name JwtFilter**  and this filter will extend **OncePerRequestFilter.**
* **After that do override** doFilterInternal() of **OncePerRequestFilter class.**

# [What is OncePerRequestFilter?](https://stackoverflow.com/questions/13152946/what-is-onceperrequestfilter)

**Ans.**

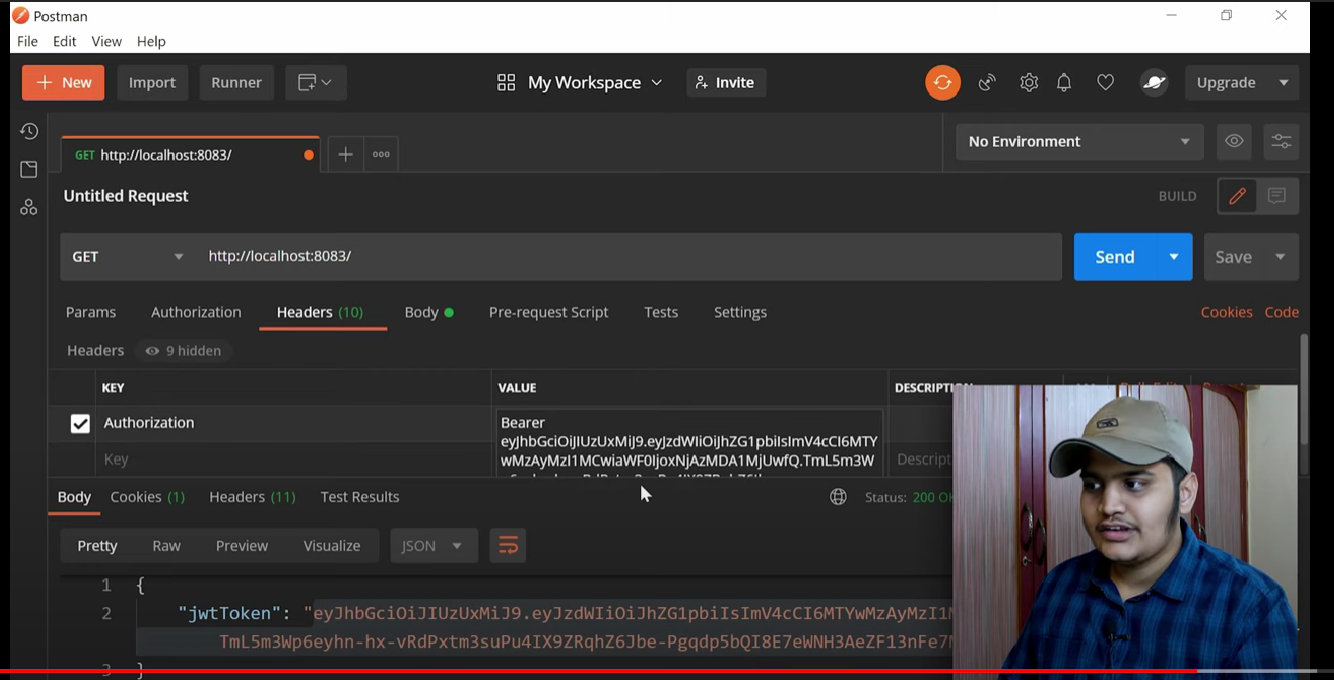
* **Filter base class that aims to guarantee a single execution per request dispatch, on any servlet container.**
* **It provides a**[**doFilterInternal(HttpServletRequest, HttpServletResponse, FilterChain)**](https://docs.spring.io/spring-framework/docs/3.2.2.RELEASE_to_4.0.0.M1/Spring%20Framework%203.2.2.RELEASE/org/springframework/web/filter/OncePerRequestFilter.html#doFilterInternal(HttpServletRequest,%20HttpServletResponse,%20FilterChain))**method with HttpServletRequest and HttpServletResponse arguments.**

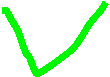
**NOTE:**  will have to autowire JwtUtility and UserService class as we have to use it in our project,

**Q. How will be passing JWT Token with every request ?**

**Ans.**

- So will be passing it in the header section with Bearer namespace

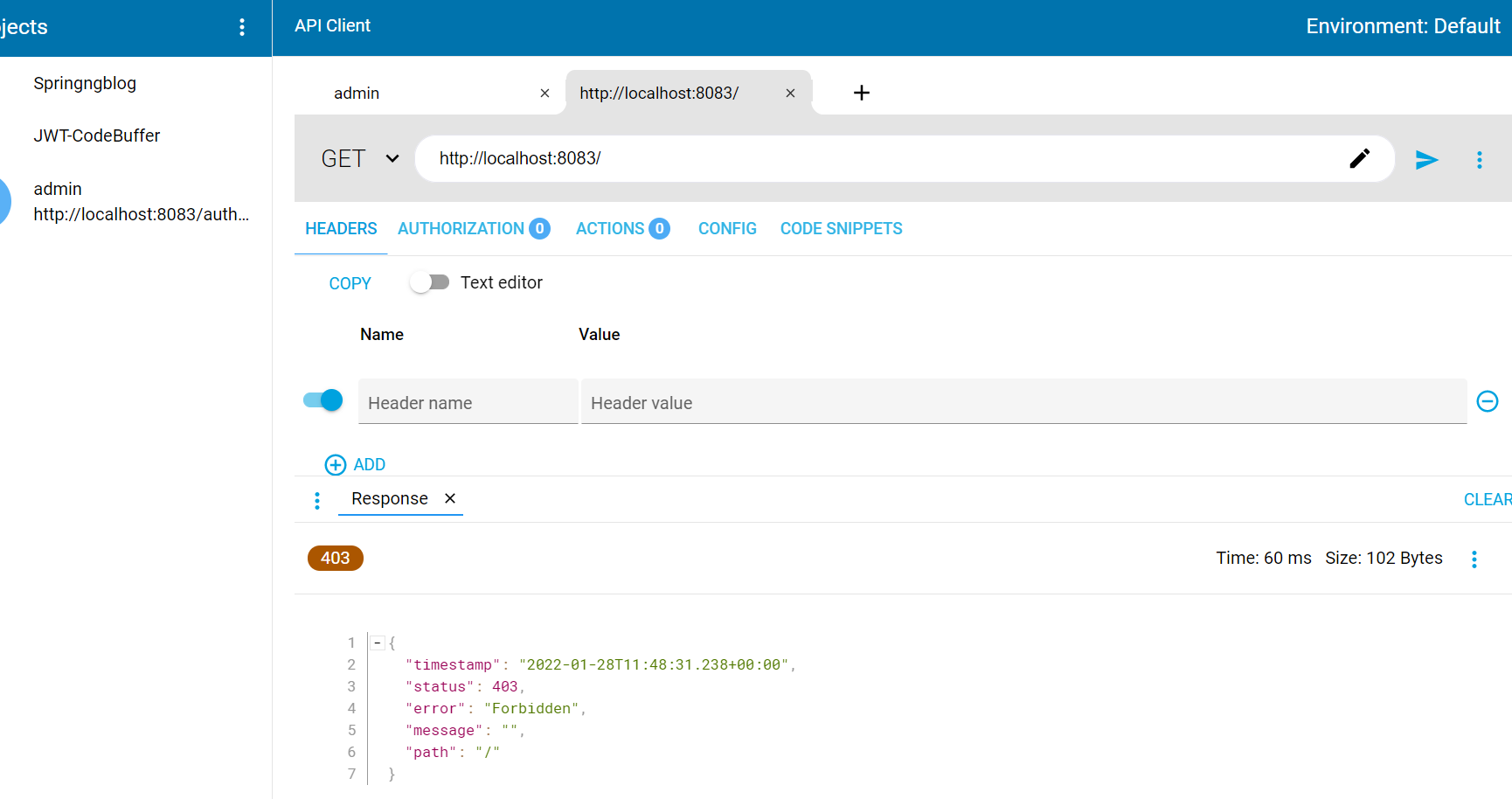
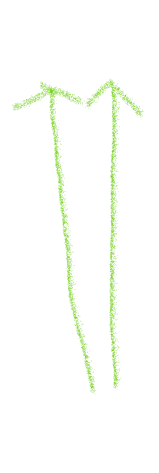




So in the Header section we selected key =**Authorization** and value=” Bearer “then paste your JWT Token” ”

So this is how will be passing token information here. So in short , we need to get the Authorization information from the request header .

So, once the token is created we want to send that token with every request which shows that the request which is coming is authenticated. And suppose we are not sending that token with the request then will get the below error,



When header information is not provided then we get this error.