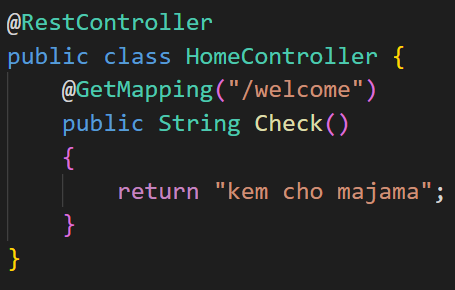
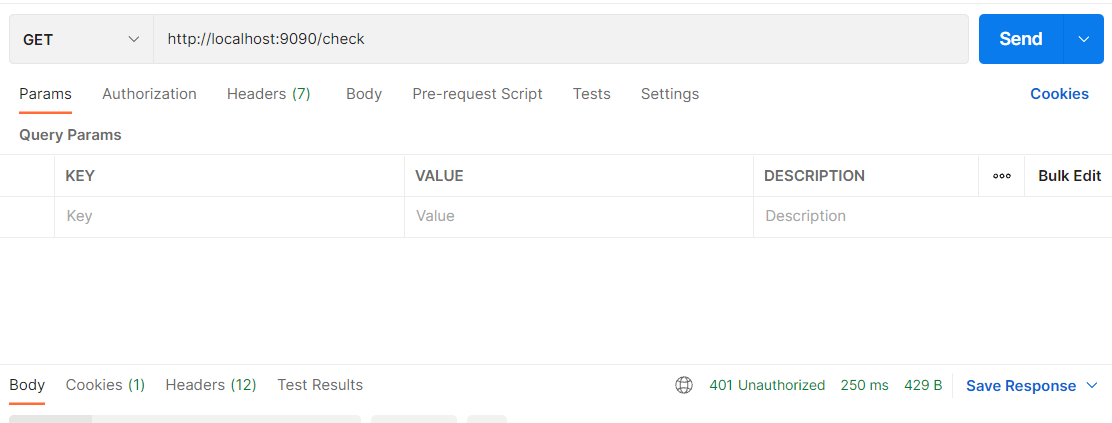
Spring Security Implementation of JWT(Json Web Token) In Spring Boot

1. For the implementation we will make an spring boot project having the following dependencies in it
   1. Spring web
   2. Spring security
   3. Mysql (If want to work with database)
   4. JPA (If want to work with database)
2. And than we will make the homeController in an package named the controllers in the existing main package of the project and will make it as an restController, Example:



1. And than by running this url the error like un-authorize comes, Example:



🡪And this error comes as we have make the spring security enabled in the project

1. Than we will make a new folder named config and in which we will make a file named MySecurityConfig and in which we will do the configuration regarding the spring security

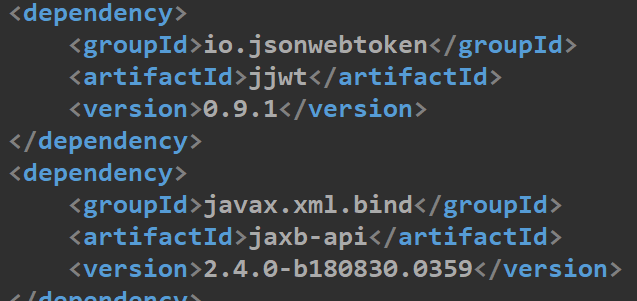
🡪And than in that file go to source and click on Override/Implement methods and implement two configure methods and make some changes as shown below:



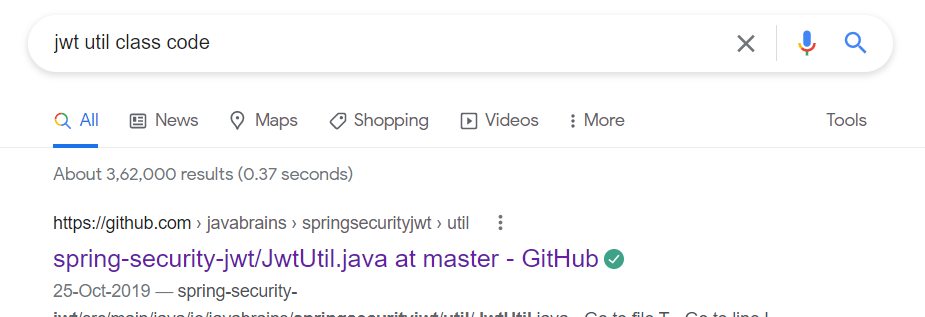
1. And than we will make a new service package and in that we will make a file named customUserDetailsService and will also make it autowired in the configure class, And we will also override the method in this customUserDetailsService from the UserDetailsService class that we have extended, As shown below:

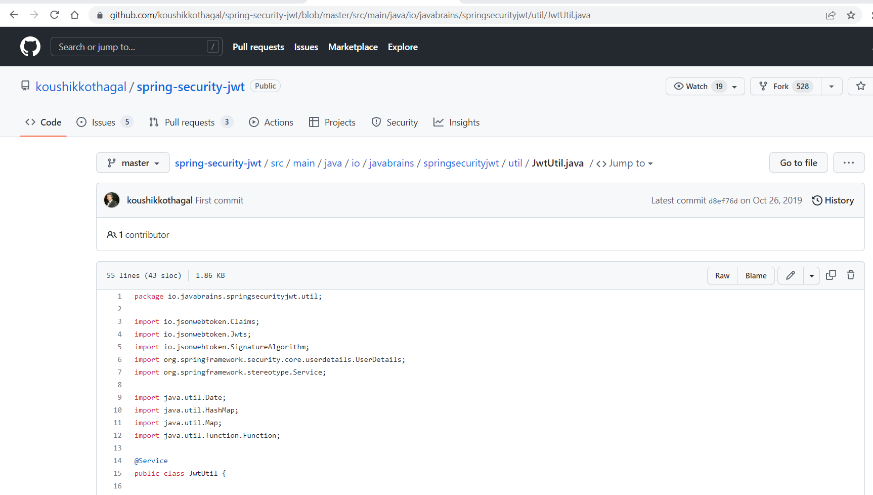


1. And than we will put the two required dependency of the jwt in the pom.xml file as shown in the figure bellow:

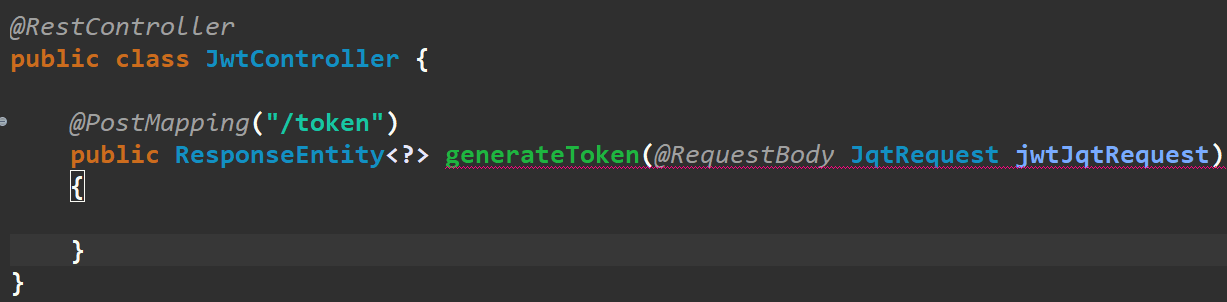


1. Than after updating the maven project, Now we need the util class of the json for that we will write the jwt util class code and we will get the code of the class from an github link and we will make it paste in our new file which is in new folder named helper and the name of new file will be JwtUtil and make sure you make the correction in the packages whenever required, The link from where we copied code is as shown below:

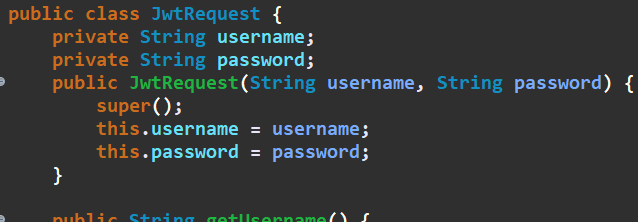




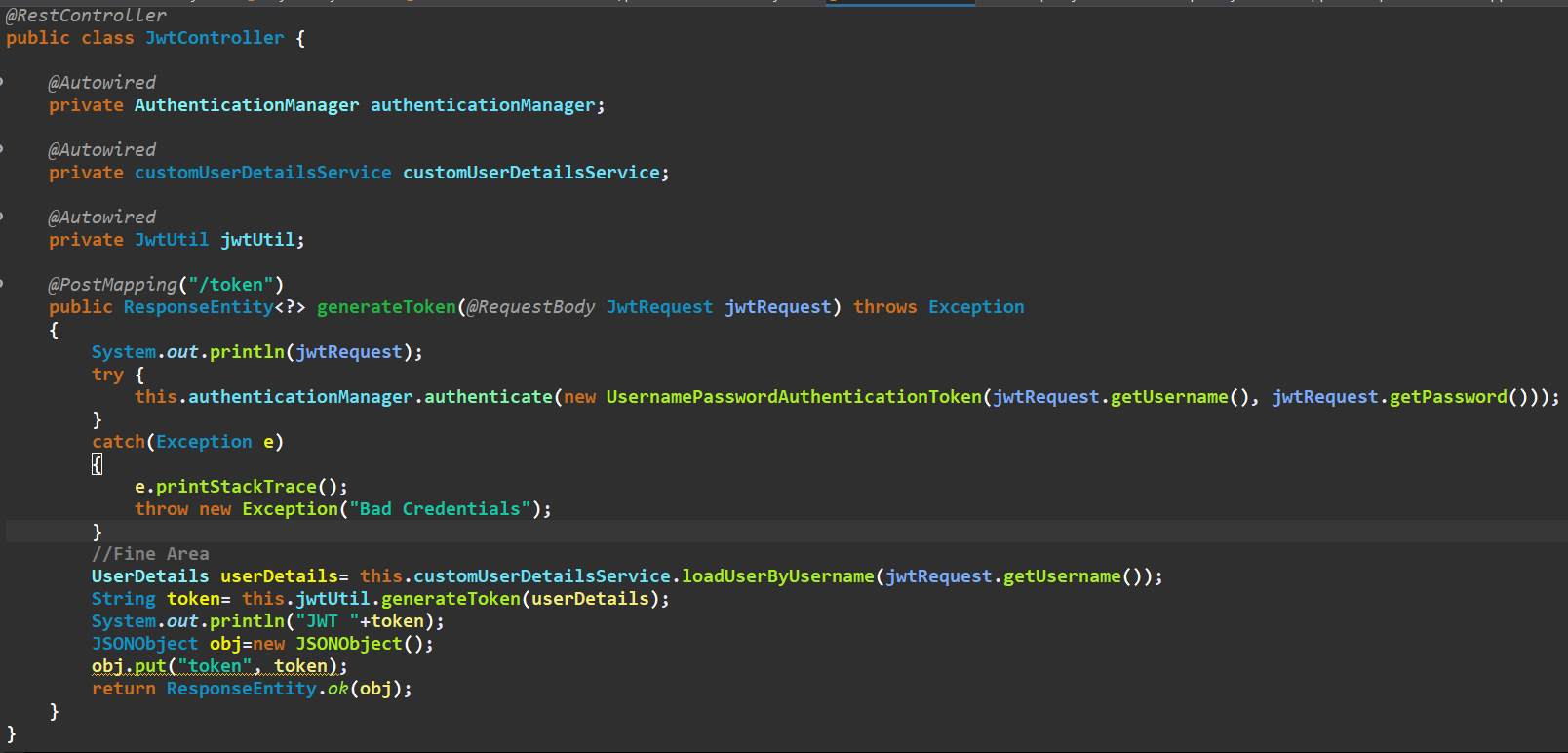
1. And than we will make our new controller named jwtController, And init the things that are to be written are as shown below, And in it there is method for creating the token:



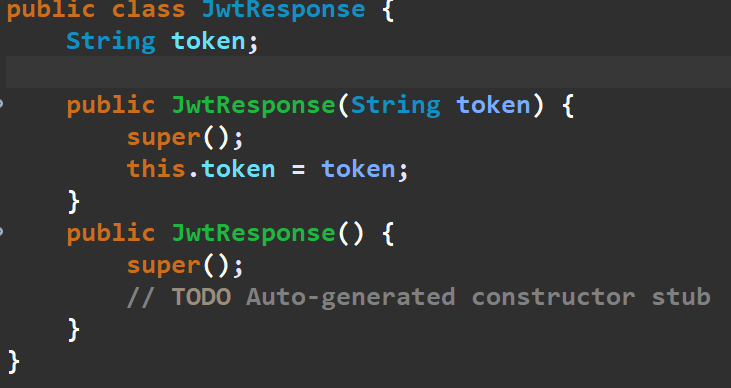
1. And when we want to generate the token than we have to give the username and password, So that we require an model of the user by name JqtRequest for now, But it can have any name as per our requirement, The example is as shown below:



1. Now we have to add the method in the JwtController class for generating the jwt which is as shown below:



1. We will also make a new class in the model package as the JwtResponse, which have the only one variable that is string named token, Example:



1. For converting the data into the json format we can use the dependency as shown below:

<dependency>

<groupId>com.googlecode.json-simple</groupId>

<artifactId>json-simple</artifactId>

<version>1.1.1</version>

</dependency>

🡪And than we can make the json from java file in many ways some of them are as shown below:

🡪The example or the usage of this method is been done in the JwtController, See its code for refrence:

**🡺Now we will work on checking the auth-token**

1. And for it we will make an filter in the config package, By name of the **JwtAuthentificationFilter** and in it, we will extend the class named OncePerRequestFilter and make it mark as the **@Component** annotation

🡪And than we will implement/override the doFilterInternal method in the class

🡪And init write the code as shown below:

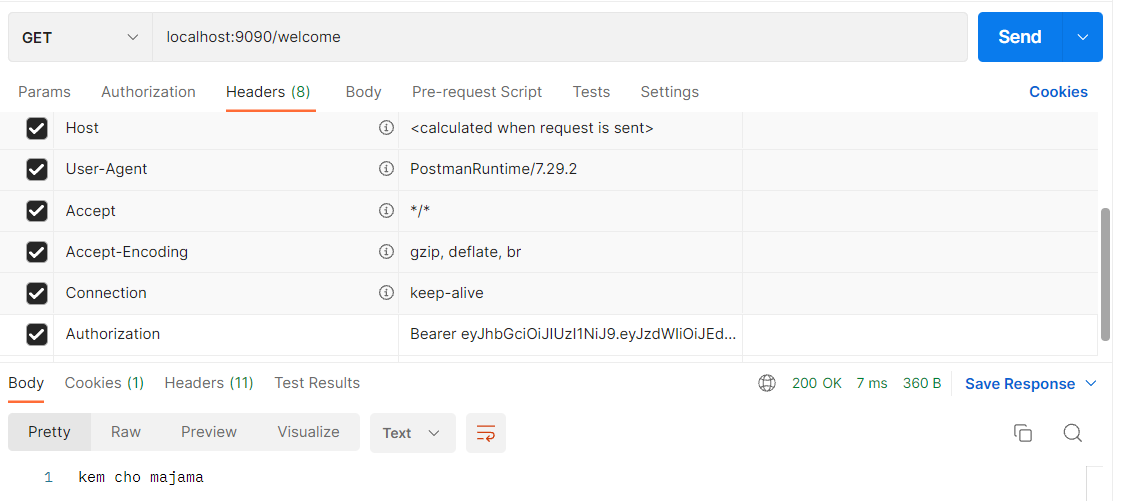
🡪In the above filter method we had first take the jwt(Json Web Token), Than we had checked if it is starting of it is from bearer and than at last we will validate it

1. And than in the **MySecurityConfig** we will add the some code in the configure method, As shown below:



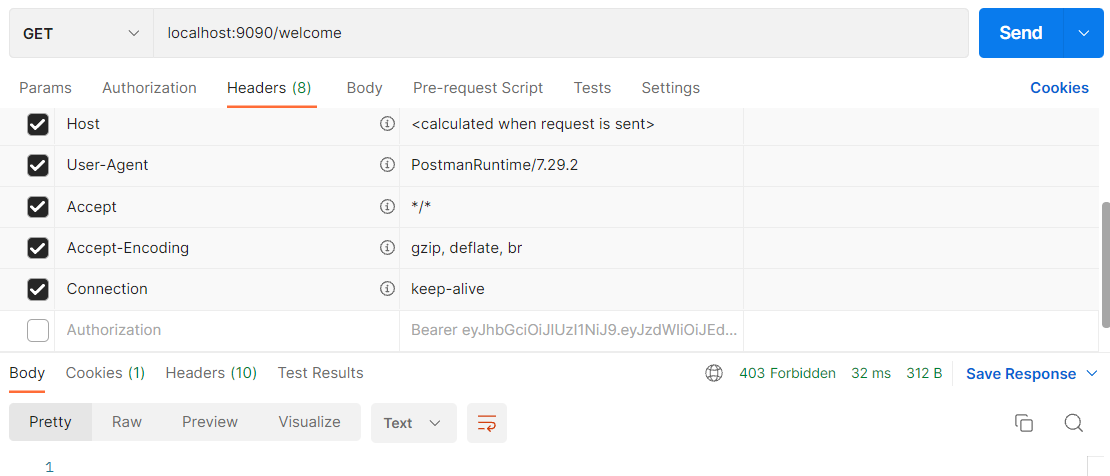
🡪In this block of code we have auto-wired the JwtAuthentificationFilter and then in the configure method, We have added the http’s method named addFilterBefore, As shown in the image

🡪And Now the web is been developed to make the auth-token and than we can also make the access to the any page who required the authentication by help of the auth-token, So if we give the following information in header as shown in image than we can take the access to the desired page, Example:

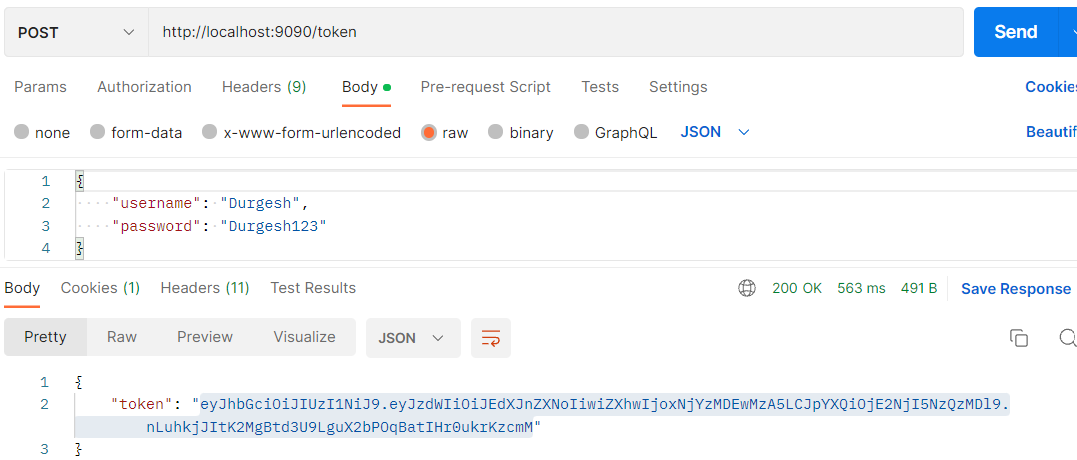


🡪Here we have to compulsorily use the **Bearer** word, and than a space and than we can write the auth-token which is required, And by doing that we will be allowed to view page

🡪And if we remove the auth-token than, there will be 403 forbidden error, As shown below:



1. And we can also make the auth-token by help of the **/token** method that we have made:



1. Now, We can also make the edit in the status code and the statement that is returning, from the server when the authentication does not take place by following the below steps:
   1. We will make a new class named **JwtAuthenticationEntryPoint** and in it we will implement the interface named AuthenticationEntryPoint

🡪Here we have written the implments as we are extending the interface

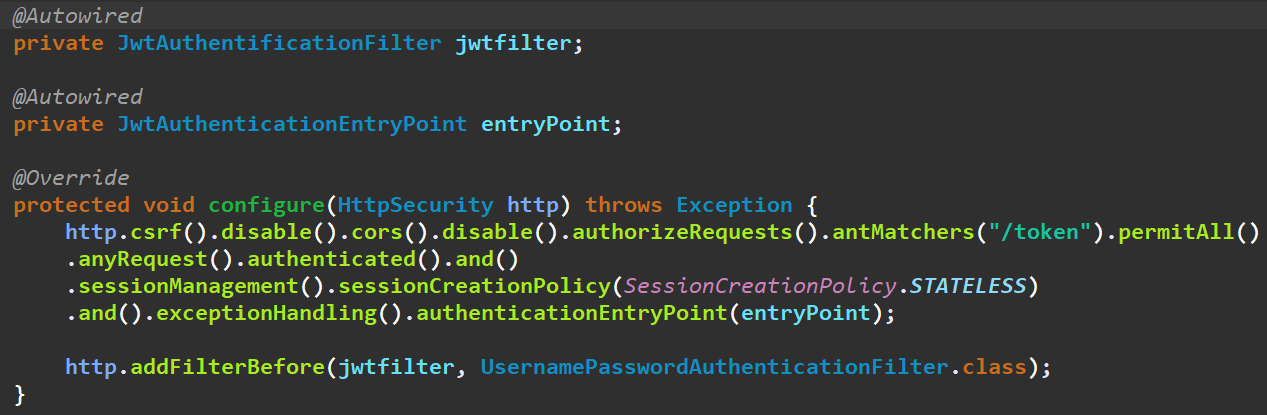
🡪And than in this class we will use the source function and in it we will use the implement/override function and from that we will get the commence method and will add some code inside the method, As shown below:



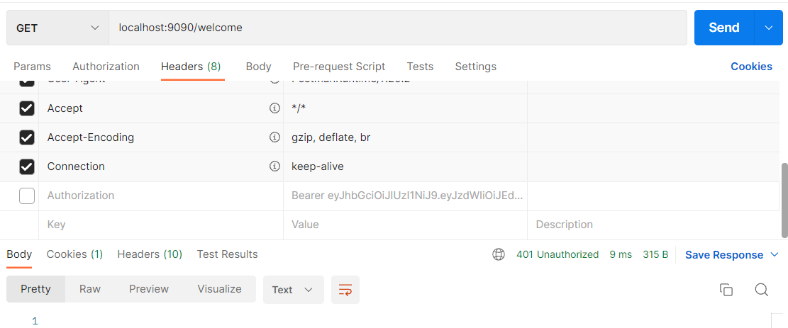
* 1. And than we will make some changes in the **MySecurityConfig** class which is the security configuration class

🡪And in it after the SessionCreationPolicy.STATELESS, we will add the another methods as shown below:

🡪We will also make the autowire of the JwtAuthenticationEntryPoint as shown in the code bellow:



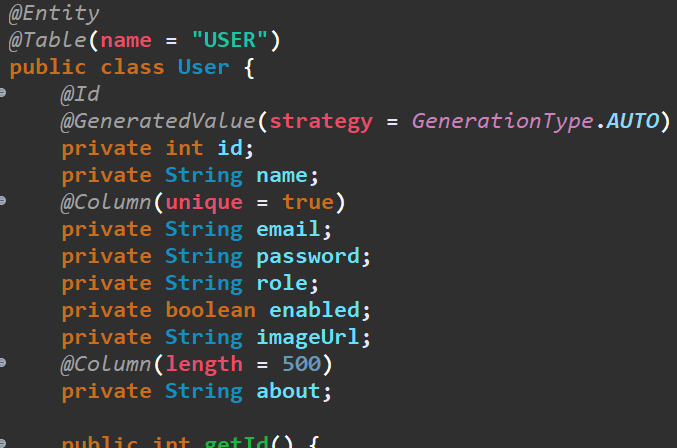
🡪And now we will check, If the our change of the error is working or not, And it is working and the result is as shown below:



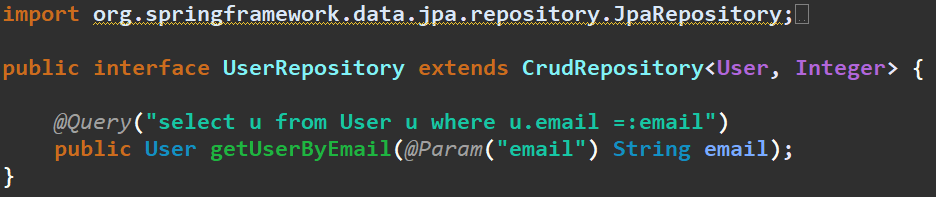
1. For making the checking of the details from the data in the database instead of the service class’s data, We will take the help of the entity class and the repository class and by help of which we can check the if the given email or username exists in the database or not

🡪For checking, We will autowire the userRepository in the service class, Example of all classes is as shown below:

🡺Sample entity class



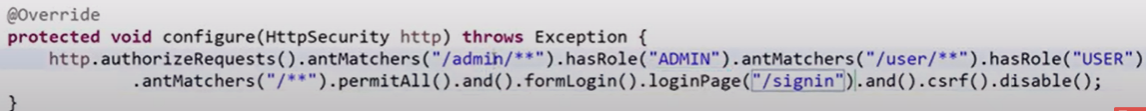
🡺Sample Repository class



🡺Sample Service Class



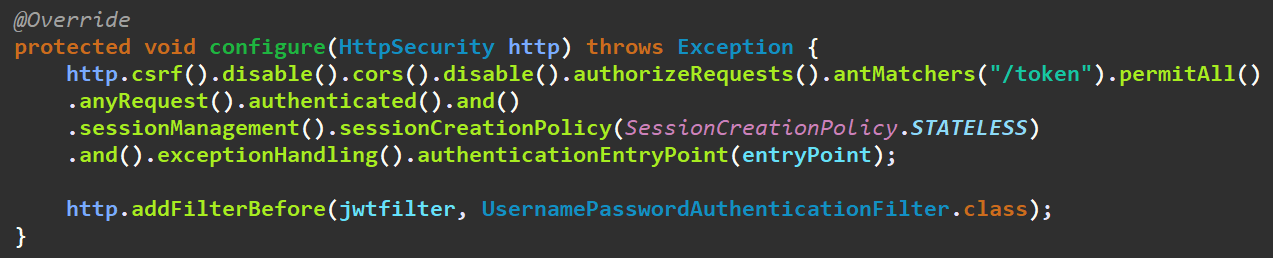
1. We can also make the change in the login page by navigating the link to the new link, By help of the **.loginPage()** method, Example:



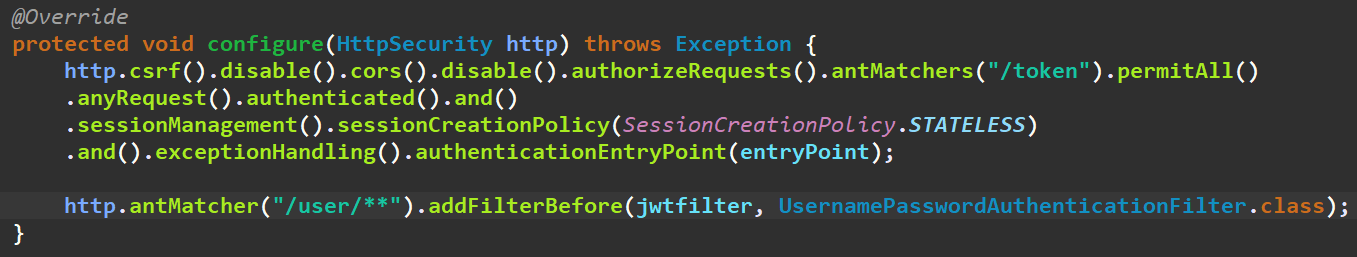
🡪Here we have to give the link of the login page in the string

1. We can also set the filter to the one of the type of the link instead of the all the links by help of the antMatchers Function, As by this method, We can put the different types of the filters on the different links, Example in the MySecurityConfig class and in it configure method:

**🡪Filter for all the links**



**🡪Filter for the link with the /users in it:**

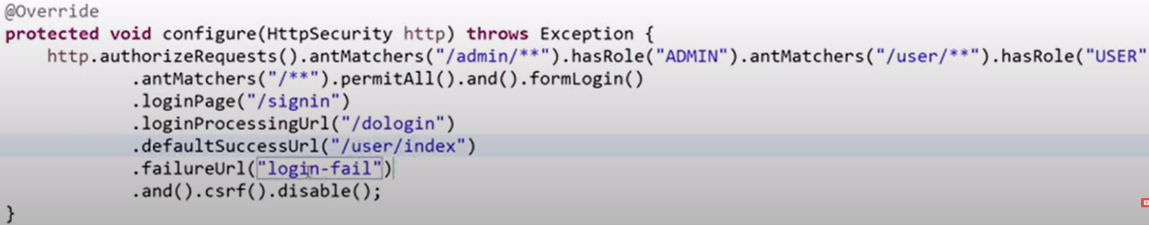


**🡪Different filter for the different links with different urls:**



🡪But make sure you make the different filter like that of the jwtfilter, for the jwtfilter1 and make it to be declared as private and than also make it autowired

1. And if we make the settings of the authenticating via auth-token than the login page will not be showed anymore, And directly it will give un-authorized error if the auth-token is not present in the required page
2. And if the spring boot app was running on the simple spring security instead of the auth-token one than we can redirect the page to different links in situations like un-authenticated, defaultSuccessUrl and etc by help of the following shown methods:



1. We can add the items in the auth-token by editing the JwtUtil class which we have made and by help of which we perform different actions to the jwt



🡪By putting the new items and its value in the claims we add that value in the particular auth-token

1. If there is the cors error while accessing the secured url, Than we have to add the .cors() function to the configuration of that specific url, Example:



1. We can also make the time limit of the jwt token to infinite by making some change in the JwtUtil.js class, And in it we will make change in the createToken method:

