Crash Course By Learn Code With Durgesh.

1.Refer the diagram show in Image How security is taken care.

Firstly there is a Filter Interface which create the Object of Authentication , then this object is passed to AuthenticationManager Interface to check its validity, but its inteface so need the other helps to check authentication, it uses the Provider Manager to check all the Authentication\_Provider implementation ,and find whether obj is supported or not.If good then authentication is validated and its property is set to true, then this is returned to Provider\_Manager .... and finally returned to Filter , which then sets the Securitycontext of object.

Now to We are setting our own security .

Time 1:00 --------------

Create the MySecurityConfig class and extends the WebSecurityConfigurerAdapter. and over ride its configure(HttpSecurity) methods.

http

.authorizeRequests()

.anyRequest()

.authenticated()

.and()

.httpBasic(); //pop-up login mechanism will be created , no logout features in this.

//username will be "user" and password generated on console.

To run it using POSTMAN : set the Authorization tab -> choose the Type: Basic Auth , and put user and password.

--------------------------------------------------

override the another configure (AuthenticationManagerBuilder auth)

auth.inMemoryAuthentication().withUser("shivam").password(this.passwordEncoder().encode("shivam")).roles("ADMIN");

auth.inMemoryAuthentication().withUser("john").password(this.passwordEncoder().encode("john")).roles("NORMAL");

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

// now since we are using the password encoder http .formbasic() , we need to give Bean of PasswordEncoder

@Bean

public PasswordEncoder passwordEncoder()

{

//return NoPasswordEncoder.getInstance(); //simple plain text as password

}

// If not work , just restart the project and open it in incognito mode.

---------------------------------------------------

AntMatchers - are used to allow the access directly skipping the spring security.

Make Modification in configure (httpSecurity)

http .authorizeRequests()

.antMatchers(HttpMethod.GET, "/home","/public/\*\* ").permitAll()

// we can also define here which Role user can access the which url

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ROLE -- high level overview (NORMAL user)

Authority -- read,write update permissions

Now if we want to allow only the users/ url access to ADMIN

and /public/ url access to NORMAL then we can use the ant Matchers like this.

.antMatchers("/public/\*\*").hasRole("NORMAL");

.antMatchers("/users/").hasRole("ADMIN");

// That means we can use the POSTMAN to check API , with users of different roles.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If we dont want to use the antMatcher , we can directly set the method with accessfier

@PreAuthorize("hasRole('ADMIN')") before particular method

, also need to enable the @EnableGlobalMethodSecurity(prePostEnabled=true) in SpringSecurity class.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CSRF token (Cross site request forgery) :

its type of attack, spring saves us from this and does not allow to add new user/update them ,

so to skip this functionality we need to make csrf().disable() in the SpringSecurity class.

We need to disable it , if using the in non\_browser client eg . POSTMAN .

In place of disabling the csrf we can also pass the Header of key, value pair (copy from cookies section.) get the csrf token need to use csrf.TokenRepository( .withHttpRequestFalse())

and we can go to Cookies seciton of PostMan and can copy the values and put them in Header tab with key ="X-XSRF-TOKEN" value="...."

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

FORM based Authentication.

(refer the Diagram / screenshot )

/logout available.

/username,password are sent,

Now in SpringSecurity //.httpBasic() , use

.formLogin()

.loginPage("/signin") //self made signup , also use antMatcher to make the sign as public , .permitAll()

.loginProcessingUrl("/dologin")

.defaultSuccessUrl("/users/")

Now make the signin page in templates folder , and also inject the spring-boot-stater-thymeleaf dependency in the pom.xml

Desing your login page.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Connecting it with DATABASE (spring\_security)

just replace the inMemoryAuthentication in Configure Method()

Need to setup the configuration in application.properties file

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/spring\_security

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.dialect=org.hibernate.dialect.MYSQL8Dialect

spring.jpa.show-sql=true

\*\* Attach the spring-boot-starter-jpa (remove the version) and my-sql-connector depedencies.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

To make table of UserName in database

@Entity

class User

{

@Id

String username;

}

Now work with general Structure :

1.Make the UserRepository Interface extends jpaRepository in Repo package

2. Now in main SpringBootSecurityLearnApplication implements the CommandLineRunner, it has the fun run, which run as soon as working starts.

3. Now we will create the users in the run method ()

using the setter function

-Inject the field ROLE in User , set construcctors , setter, getters etc

\*\* Imp : while setting role , put ROLE\_USER in setter injection , other times "ROLE\_" is attacjed internally by spring.

4.Autowire the UserRepository and BCryptPassword encoder to save the values

Make the another user like this process only.

The Bean type should be of BCryptPassword encoder

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now next Thing : Make the new class

CustomUserDetail and it implemtents the UserDetails interface , which Spring security uses to maintain the configuration of The Model.

Now override all the methods in CustomUserDetail :

We need User obj to getPassword()....etc

- make constructor of user. and set the password, username using this obj, and set all other properties as true;

-Also in GrantedAuthority Method just : make HashSet

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

HashSet<SimpleGrantedAuthority> set= new HashSet<>();

set.add(new SimpleGrantedAuthority(this.user.getRole()));

return set;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now make new class @Service CustomUserDetailsService it implements UserDetailsService ,

and override the loadUserByUserName()

declare the obj of UserRepository

.

. public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

//User user=this.userRepository.getById(username);

User user=this.userRepository.findByUsername(username);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now at last we need only to use the @Autowired CustomUserDetailService obj in main class.

and just modify the configure()

{

auth.userDetailsService(customUserDetailService).passwordEncoder(passwordEncoder());

}

Refer video of last 20 min

Now just re run your project with the spring-boot-app and try to access the localhost:8080/users/

it will redirect to the signin page, now when admin signin then successfully page is show, other case Forbidden Error.

if we try to access localhost:8080/public/login/ ->

it will redirect to signin page and when normal user credintials are given , its successfullu open the login page.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-------

Error comed in video , because while setting the role, we need to pass the setRole("ROLE\_USER");

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_--