Text

Description automatically generated

Rightclick - > run as – Spring boot ->Now Proj will start and looks like below:

Graphical user interface, text

Description automatically generated

Here we will see password gerenared by spring security for authentication:

Open Postman:

Graphical user interface, text, website

Description automatically generated

This will display the result after providing password generated by Spring application. For each and every try , password will change.

IMPORTANT:

What is @RestController in spring boot?

**Spring** 4.0 introduced the @**RestController** annotation in order to simplify the creation of RESTful web services. It's a convenient annotation that combines @Controller and @ResponseBody, which eliminates the need to annotate every request handling method of the controller class with the @ResponseBody annotation

**Spring** @**ResponseBody**

@**ResponseBody** is a **Spring annotation** which binds a method return value to the web **response body**. It is not interpreted as a view name. It uses HTTP Message converters to convert the return value to HTTP **response body**, based on the content-type in the request HTTP header

@Controller -> Used to recognize the class

**@GetMapping**

@GetMapping annotation maps HTTP GET requests onto specific handler methods. It is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET).



The username/password we enter will passs through Authentication Filter then Authentication Manager and Authentication Provider.

Provider and Manager are real Authentication Logic.

After successful Login , under security context our Jsession ID will get stored. This is generated after password/User Authentication and sends back to Authentication Fileter by Provider for Verification purpose.

Until JSession ID is deleteded , this will never prompt for password again. Because, when ever we relogin , it will Check particular jsession and authenticate it.

Lets see the above concept in POSTMAN:

Postman -> header->cookies

Graphical user interface, application, Teams

Description automatically generated

This is jession ID.

After removeal of password also it will work, this is because it verifies only Jession ID.

Graphical user interface, text, application, email

Description automatically generated

Try removing jession id and don’t provide password. It wont work.

Graphical user interface, text, application, email, website

Description automatically generated

Now we need to enter password again to get output.

This is because, after first login jsession id is stored. When next time client tries to login it will check jession id is created or not for particular request. If already authenticated, it will display OP irrespective of password.

Login first time:

JSESSIONID=69EC69B772F32C5CE256D9FA0683A07A; Path=/; Domain=localhost; HttpOnly;

After deleting and Logged In again:

JSESSIONID=5D039287FDD0B7869421DD23DA877CDB; Path=/; Domain=localhost; HttpOnly;

**Custom Implementaion For User Details:**

Create a class that extends WebSecurityConfigurerAdapter

It has 3 Methods

We are using Configure() with different style of parameter.

Authentication Manager Builder allows us to Manage.

**TO CREATE CUSTOMIZED USERNAME/PASSWORD:**

package com.divya.spring.security;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

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

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

*@Configuration* //used as configuration by Spring boot to fetch username and password automatically

public class MySecurityConfig extends WebSecurityConfigurerAdapter{

*@Override*

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

//For encoding Password

BCryptPasswordEncoder passwordEncoder=new BCryptPasswordEncoder();

//Create and store user on flyway

InMemoryUserDetailsManager userDetailsService=new InMemoryUserDetailsManager();

//user is a class from spring to create user

UserDetails user =User.*withUsername*("divya").password(passwordEncoder.encode("krishnan")).authorities("read").build();

userDetailsService.createUser(user);

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

}

//Customize Security

*@Override*

protected void configure(HttpSecurity http) throws Exception {

http.httpBasic();

//Means all request that are received will be authenticated

http.authorizeRequests().anyRequest().authenticated();

}

}

OP:

Graphical user interface, text, application, email

Description automatically generated

Password Encoder is mandatory.

package com.divya.spring.security;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

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

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

*@Configuration* //used as configuration by Spring boot to fetch username and password automatically

public class MySecurityConfig extends WebSecurityConfigurerAdapter{

//@SuppressWarnings("unused")

*@Autowired*

private PasswordEncoder passwordEncoder;

*@Override*

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

//For encoding Password

BCryptPasswordEncoder passwordEncoder=new BCryptPasswordEncoder();

//Create and store user on flyway

InMemoryUserDetailsManager userDetailsService=new InMemoryUserDetailsManager();

//user is a class from spring to create user

UserDetails user =User.*withUsername*("divya").password(passwordEncoder.encode("krishnan")).authorities("read").build();

userDetailsService.createUser(user);

//This is used to encode password.

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

}

//Customize Security

*@Override*

protected void configure(HttpSecurity http) throws Exception {

http.httpBasic();

//Means all request that are received will be authenticated

http.authorizeRequests().anyRequest().authenticated();

}

*@Bean*

//Invoke this method where we want or autowire it

public BCryptPasswordEncoder paswordEncoder()

{

return new BCryptPasswordEncoder();

}

}

Password Encoder is mandatory. If we don’t include the highlighted line ,

Include the below:

*@Autowired*

private PasswordEncoder passwordEncoder;

*@Bean*

//Invoke this method where we want or autowire it

public BCryptPasswordEncoder paswordEncoder()

{

return new BCryptPasswordEncoder();

}

This is also a password encoder, which is autowired.

Note:

If we remove password Encoder, Application will start properly, but when we tries in postman, we will not get any response and after postman request we will get huge stack of exception in STS.

CREATE CUSTOM AUTHENTICATION PROVIDER:

Create it inside Authenticatuin Provider.

No need of userdetails./password encoder to do this.

To do the above we need to implements AuthenticationProvider Interface.

It has 2 methods.

1. Authenticate() of return type Authenticate
2. Supports() of return type Boolean.

Authenticate() :

Under this method it has getName() and getCredentials().

getName() – Usef to get username and store it in a Stringvariable

getCredentials() - used to get password and convert to toString() and store it in a String variable.

Now using if-else loop, check the give password is right or not.

If loop returns the below , because expected return type is authenticate.

return new UsernamePasswordAuthenticationToken(userName, password,Arrays.*asList*());

It has 3 Parameters.

Username=authenticate.getName();

Password=authenticate.getCredentials().toString();

Arrays.asList() == Passing empty list because we don’t any authorization/grant type to provide.

In else part we can throw exception.

All the above we are creating in AuthenticationProvider. [Refer the Diagram]

From provider , control will pass to Authentication manager and from Manager control will pass to authenticate filter.

In filter, it will check for authentication is success/not.

If Success , username/password will get stored in security context.

Else it will throw the exception.

Second method is supports.

It just check provider supports or not authentication type returned by Authenticate manager is right / wrong.

For Basic Authentication:

http.httpBasic();

Use this for basic authentication and check in Postman.

For Form based Authentication:

http.formLogin();

Use this for form based authentication and open browser.

Give <http://localhost:8080/hello>

This will take to login screen and provide user name and password configured MyAuthenticationProvider.

Hello is mapped as below:

Text

Description automatically generated

To Add More Endpoints:

Below line used to authenticate any or all request , we are providing.

http.authorizeRequests().anyRequest().authenticated();

To be more specific , we can use

http.authorizeRequests().antMatchers("/hello").authenticated();

This is used to authenticate , whoever using hello endpoint.

**DenyAll()**

Below line indicates that our URL should accept only /hello , for any other request it should not allow.

http.authorizeRequests().antMatchers("/hello").authenticated().anyRequest().denyAll();

**Example:**

This is login screen

Graphical user interface, text, website

Description automatically generated

After Loging to /Hello:

Graphical user interface, text, application

Description automatically generated

For /bye endpoint:

Graphical user interface, text, application, email

Description automatically generated

TO CREATE CUSTOM IDENTITY:

by Implementing Filter Interface:

Create a class like below.

Text

Description automatically generated

doFilter(request,response):

The doFilter method of the Filter is called by the **container** each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain. The FilterChain passed in to this method allows the Filter to pass on the request and response to the next entity in the chain.

Now Go to MySecurityConfig.java class and Add the below Snippet in configure Method :

Graphical user interface, text

Description automatically generated

httpBasic is for basic Authentication purpose.

Authorize Request method is to call “hello” for authenticated user.

addFilterBefore() -- > This is the method of https class, this will register the custom Filter before Spring security filter.

To See The OP:

Open Postman.

Graphical user interface, text, application, email

Description automatically generated

After postman is done. Go to STS Console

Graphical user interface, text

Description automatically generated

Will see the System.out.println Statements.

Instead of implementing Filter Interface, Spring provides some Abstract class and we can extend it for its usage.

GenericFilterBean: It is the implementation of javax.servlet.filter, treats its config parameter.

If we want to use any initial parameters to web.xml by simply having getters and setters method.

Init parameter to web.xml will be automatically injected, through setters method.

OncePerRequestFilter 🡺 Interface.

This can be used if we want to execute a filter only once.

This supports Httpservlet request and response.