Date : 09-Jun-21
Spring Boot 9AM
Mr. RAGHU

Task:
https://www.youtube.com/watch?v=rgG2_T-OB8g

Spring Security -- InMemory Authentication

>> Used in Dev/Test Environments only. Not used in production.
>> It is Storing data in RAM (not using any Database).
>> We have to provide un/pwd, roles as hadrd coded (direct values in code).

>> to create one User data in RAM(InMemory Authentication) code:

```
auth
  .inMemoryAuthentication()
  .withUser("SAM")
  .password("{noop}SAM")
  .authorities("ADMIN");
```

>> We should use any one password encoder, else provide {noop} (No password encoder)

==========

*) Spring Security advantages:-

a. Default Login form :
There is a pre-defined Login HTML Form is given by Spring Security.
We can even customize.

b. Default Session Management

I. create one new HTTP Session on Loing success
[HttpServletRequest request]

```
HttpSession session =  request.getSession(true);
                (or)
HttpSession session =  request.getSession();
```

II. Read existed session

```
HttpSession session =  request.getSession(false);
```

III. set/modify/remove data to Session [manually]

```
session.setAttribute("user",userObj);
session.removeAttribute("user");
```

IV. DELETE/Remove Session on logout

```
HttpSession session =  request.getSession(false);
session.invalidate();
```

c. Provides default for service URLs

>> Default login URL : /login (GET) (we can see this page)

```
  >> Default Login check URL    : /login (POST) [do login validation]
  >> Default Logout URL         : /logout (GET)  [link to do logout]
  >> Default Logout success URL : /login?logout  [if we click on
logout success]
  >> Default Login Failure URL  : /login?error  [if we enter invalid
un/pwd]

================================================================
Stateful : Store user data at server side using HTTPSession
Stateless: Do not store user data at server, communicate using one
common KEY
        (Token/JWT)-- Store require data at client(browser side)

https://www.youtube.com/watch?v=feETfZbvu-k


*) we can use PasswordEncoder that will convert Readable pwd into
unreadable format

package in.nareshit.raghu;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

public class Test {

        public static void main(String[] args) {
                String pwd ="AJAY";
                BCryptPasswordEncoder encoder = new
BCryptPasswordEncoder();
                String encPwd = encoder.encode(pwd);
                System.out.println(encPwd);
        }
}

*** Decode is not possible. You Form Login Pwd also encoded and
compared
        with DB Password, if matching success, else fail.
----------------------------------------------------------------
---------
                        Spring Security using JDBC

>> We are storing data in Database table and do login validate using
same data.


*) This JDBC Security needs input as
a. link with Database Connection

@Autowired
private DataSorce datasource;

b. SQL#1 To fetch user name, pwd,enabled data using user name

select uname, upwd, uenabled from user_tab where uname=?


c. SQL#2 To fetch user name,roles data using user name
```

(if first SQL returns data then only 2nd/this one is executed)

```
select uname, urole from user_tab where uname='SAM'
```

d. provide password encoder

```
        @Autowired
        private BCryptPasswordEncoder passwordEncoder;
```

--Database setup----------

a. create database table

```
> create table user_tab(uid int, uname varchar(20), upwd varchar(70),
                uenabled int, urole varchar(20));

> desc  user_tab;
```

b. Insert data

```
insert into user_tab values(1, 'SAM',
'$2a$10$pjZ/Js6pVe5YZxX0PUQHluTEmEPTfY/J2u6l9iNifPovIQYeS08We',1,'ADMI
N');
insert into user_tab values(2, 'SYED',
'$2a$10$z1XBko5AciezIg9DGiScB./f8k1be4qHUqY.l/CcpHh1HkLa1t5Uq',1,'EMPL
OYEE');
insert into user_tab values(3, 'AJAY',
'$2a$10$RS.vg5fjlSb7bjYGDFXOVObZCdQzX0TpArzJQyW0mqeON5sPtPujO',1,'GUES
T');

insert into user_tab values(4,
'MR',$2a$10$DATEfIQEhhCLbX69kpmGc.0yYbO3uqtlth0OAt4yPSpLci.H7HjFW,1,'A
DMIN');
```

======

(coding)==========

1. Create Starter

```
Name : SpringBoot2SecurityJdbcEx
Dep  : web, devtools, mysql, jdbc api, security, thymeleaf

pom.xml : spring-boot-starter-jdbc
```

2. properties

```
server.port=9696

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/boot9am
spring.datasource.username=root
spring.datasource.password=root
```

3. Controller

```
package in.nareshit.raghu.controller;

import java.security.Principal;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
```

```
@Controller
public class AppTestController {

	@GetMapping("/home")
	public String showHome() {
		return "HomePage";
	}

	@GetMapping("/admin")
	public String showAdmin() {
		return "AdminPage";
	}

	@GetMapping("/emp")
	public String showEmployee() {
		return "EmployeePage";
	}

	@GetMapping("/profile")
	public String showProfile(Principal p,Model model)
	{
		model.addAttribute("userName", p.getName());
		System.out.println(p.getClass().getName());
		return "ProfilePage";
	}

	@GetMapping("/denied")
	public String showAccessDenied() {
		return "AccessDenied";
	}
}
```

4. UI Pages

```
--AdminPage.html--
<html xmlns:th="https://www.thymeleaf.org/">
	<head></head>
	<body>
		<h3>WELCOME TO ADMIN PAGE</h3>
		<a th:href="@{/signout}">LOGOUT</a>
	</body>
</html>
---EmployeePage.html--
<html xmlns:th="https://www.thymeleaf.org/">
	<head></head>
	<body>
		<h3>WELCOME TO EMPLOYEE PAGE</h3>
		<a th:href="@{/signout}">LOGOUT</a>
	</body>
</html>
----ProfilePage.html---
<html xmlns:th="https://www.thymeleaf.org/">
	<head></head>
	<body>
		<h3>WELCOME TO PROFILE PAGE</h3>
		<h4>Hello User <span th:text="${userName}"></span>
```

```
</h4>
                <a th:href="@{/signout}">LOGOUT</a>
        </body>
</html>

--HomePage.html--
<html xmlns:th="https://www.thymeleaf.org/">
        <head></head>
        <body>
                <h3>WELCOME TO HOME PAGE</h3>
        </body>
</html>

5. Security Config
package in.nareshit.raghu.config;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders
.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecuri
ty;
import
org.springframework.security.config.annotation.web.configuration.Enabl
eWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSe
curityConfigurerAdapter;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import
org.springframework.security.web.util.matcher.AntPathRequestMatcher;

@EnableWebSecurity
@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {

        @Autowired
        private BCryptPasswordEncoder passwordEncoder;

        @Autowired
        private DataSource datasource;

        protected void configure(AuthenticationManagerBuilder auth)
throws Exception {
                //a. link with Database Connection
                //b. SQL#1 To fetch user name, pwd data using user
name
                //c. SQL#2 To fetch user name,roles data using user
name
                //d. provide password encoder

                auth.jdbcAuthentication()
```

```
                .dataSource(datasource)
                .usersByUsernameQuery("SELECT UNAME, UPWD, UENABLED
FROM USER_TAB WHERE UNAME=?")
                .authoritiesByUsernameQuery("SELECT UNAME, UROLE FROM
USER_TAB WHERE UNAME=?")
                .passwordEncoder(passwordEncoder)
                ;
        }

        protected void configure(HttpSecurity http) throws Exception {
                http.authorizeRequests()
                .antMatchers("/home").permitAll()
                .antMatchers("/admin").hasAuthority("ADMIN")
                .antMatchers("/emp").hasAuthority("EMPLOYEE")
                .anyRequest().authenticated()

                .and()
                .formLogin()
                .defaultSuccessUrl("/profile", true)

                .and()
                .logout()
                .logoutRequestMatcher(new
AntPathRequestMatcher("/signout"))

                .and()
                .exceptionHandling()
                .accessDeniedPage("/denied")
                ;


        }
}
```

6. AppConfig

```
package in.nareshit.raghu.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
public class AppConfig {

        @Bean
        public BCryptPasswordEncoder encoder() {
                return new BCryptPasswordEncoder();
        }
}
```

==========

7. For password encode manually

```
package in.nareshit.raghu;

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

```
public class Test {

	public static void main(String[] args) {
		String pwd ="AJAY";
		BCryptPasswordEncoder encoder = new
BCryptPasswordEncoder();
		String encPwd = encoder.encode(pwd);
		System.out.println(encPwd);
	}
}
```

*) Spring Security saying : current user data(username) can be found using 'Principal' acces where ever you want inside your code.

```
String name = p.getName(); //current user/login name
```

>> Impl class is : UsernamePasswordAuthenticationToken.
[do not use this class directly]

Servlets HttpServletRequest / RequestFacade?