```
                                  Date : 07-Jun-21
                                  Spring Boot 9AM
                                    Mr. RAGHU
                  ----------------------------------------------
                              Security using Spring Boot

a. Form based Login
    | - stateful   (HttpSession) -- [Recomanded for monolithic apps]
    | - stateless  (JWT / Token based) -- [distributed applictions]

b. Open Authorization (OAuth 2.x | SSO)
 ======================================================================
======
        JAAS : Java Authentication and Authorization Service

Authentication : Verify username and password ( Login check)
Authorization  : Verify Role (Authority checking)

--Bank Application---
SAM User --> 'XYZ Bank' (new account)--> internet backing uid:111,
pwd:222

> Can SAM trasfer his money to his friend account? YES
        Req: /acc/send?from=__&to=__

> Can SAM apply for loan in XYZ Bank? YES
    Req: /acc/applyLoan

> Can SAM Approve his loan ? NO  Only Manager (SYED)

> Can SYED Approve loan ? YES

*) User, Manager, Cashier, Accountent...etc --- Authorities (Roles)
----------------------------------------------------------------
Authorization : Which operation can be done by what ROLE .


*) DelegatingFilterProxy, is a pre-defined filter given by Spring F/w
   [org.springframework.web.filter], this filter support Security
Concept
    ' A & A '.

*) This is pre-defined, auto-configured filter, it takes one class
input
   havig 'A&A' Configuration.

*) We need to define one impl class for 'WebSecurityConfigurerAdapter'
defined in package:
org.springframework.security.config.annotation.web.configuration
and ovrride methods:
 1. configure(AuthenticationManagerBuilder) : void
        For Authentication (un/pwd)
 2. configure(HttpSecurity http) : void
        For Authorization (Role, additional config)

*) Abve logic is given as input to filter for execution.
```

```
----------------------------------------------------------------
*) Do not store plain text password in Database, use PasswordEncoder
   ex: BCryptPasswordEncoder
[org.springframework.security.crypto.bcrypt]

===Overview code===========================================
pom.xml
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
</dependency>


a. AppConfig
package in.nareshit.raghu;
//ctrl+shift+O
@Configuration
public class AppConfig {

        @Bean
        public BCryptPasswordEncoder pwdEncoder() {
                return new BCryptPasswordEncoder();
        }

}

c. Security Config
package in.nareshit.raghu;
//ctrl+shift+O
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

        @Autowired
        private BCryptPasswordEncoder encoder;

        protected void configure(AuthenticationManagerBuilder auth)
                        throws Exception
        {
                //authentication (InMemory,JDBC,ORM)
        }

        protected void configure(HttpSecurity http)
                        throws Exception
        {
                // Authorization(permitAll, hasAuthority,
authenticated)

                // Form Configuration

                // Logout Configuration

                // exception configuration


        }
}
```

```
===========================================================
Authentication : (3) types
a. InMemoryAuthentication
     --> RAM (To store un/pwd | Random Access Memory )

b. JdbcAuthentication
       Store un/pwd... inside DB table,
       store/validate using JDBC - API(SQLs)

c. **** UserDetailsservice | ORM Authentication ***
       Store un/pwd... inside DB table,
       store/validate using ORM (Spring Data JPA)
---------------------------------------------------
Authorization : 3 Levels (Which URL/PATH can be accessed by whome)

a. PermitAll   : Every one can access this URL without any login

ex:  /register, /welcome, /contactUs, /aboutUs ...etc

b. hasAuthority : Must login and have role ( Login + Matching Role )

ex:  /approveLoan  ( Login + Manager), /applyLoan ( login + User)
     /viewBalance ( login + Cashier/User)

c. authenticated : only Login (no role is required/any role is valid)

ex: /sendMail (only Login) , /logout (only Login), /viewProfile (only
login)
        /updatepwd(only login)

============================================================
.antMatchers() : This is used to link Controller method with Access
Level
                      (who can access what URL PATH)

1. /register URL can be accessed by everyone

  .antMatchers("/register").permitAll()

2. /home URL can be accessed by every user after login (any role is
valid)

   .antMatchers("/home").authenticated()

3. /activateUser URL can be accessed by ADMIN only after login

  .antMatchers("/activateUser").hasAuthority("ADMIN")

4. /exportData  URL can be accessed by ADMIN (or) CLERK (or) ACNT

 .antMatchers("/exportData").hasAnyAuthority("ADMIN","CLERK","ACNT")

 Roles can be given any words by programmer (hard coded, Enum, DB
based,
     properties).
```

*) antMatchers(Path1,Path2,...)
*) antMatchers supports symbols *(any char) and **(multi level path)

ex: /stdSave, /stdone, /stdRemove

antMatchers("/std*").___________________________

Ex: /emp/save, /emp/export, /emp/find/{id}, /emp/data/delete/{id}

antMatchers("/emp**").______________________________