①

Spring-security flow

we send the request /account.htm → who will receive request → spring securityFilterchain ~~[struck]~~ of → Delegating filter proxy

→ Delegating filter proxy will dispatch request to whoom

→ actual securityFilterchain i.e. configured as a bean definition in the ioc container.

→ once the spring security filter chain has receive the request will checks for what intercept url what is it /account.htm, which user can access i.e. full authenticated.

then checks for whether current user is logged in or not.

→ if not logged → forward the request to whoom login page based on basic authentication (<security: basic/>).

The user will enter username and password (we are not writting login page also i.e. loggin handler also we dont need write).

Now the user will enter username and password and click on submit where does the request will comes to login handler, login handler will call whoom → authentication-manager → authentication-manager will call whoom → authentication-provider.

→ authentication-provider will call whoom → authentication-service.

who is that user-service. The user service has been already populated with usernames and passwords of the roles of the user. It returns what appropriate user details of object to whoom i.e. authentication-provider. Here authentication-provider will validated

if the username and password is matching then it returns authentication object → User Principle object will be wrapped into authentication object and returns to authentication manager. authentication manager will wraps that in to security context places in to threadlocal of the application and returns the control to whoom login handler. Login handler will forwards the user to whoom → whatever the resource is accessing. i.e. how we need to enable spring security for our application.

security-beans.xml

→ spring-security configuration file is part of context loader listner (IoC container) not dispatcher servlet ioc because to decouple from mvc components. so that it will works irrespective of spring mvc components

→ Dont directly configure these classes as bean definitions in applicationContext.xml. Let them we configured in seperate configuration file (i.e. security-beans.xml).

→ If security-beans.xml how it will be configured sure IoC container in contextlistner IoC.

```
<context-param>
    <param-name> contextConfigLocation </param-name>
    <param-value> /WEB-INF/applicationContext.xml, /WEB-INF/security-beans.xml </param-value>
</context-param>
```

In application-context.xml all the application components are there like service-bean, persistance-beans, transactionality and all.

→ security configuration is not related our application classes. it is related spring security classes.

→ It is not our application components that is the reason these are not our application components but we are using spring security (using ;) also here. because context listner will store all configuration our application beans ⟷

so quickly disable spring security just remove in the context-param of /WEB-INF/security-beans-xml.

security-beans-xml

① First and we need security filter.

So we dont know securityFilter class name to configure as a bean definition in the security-beans.xml. Instead of it spring security has been provided security-namespaces convient tags.

① <security:HTTP/> } this will enable security HTTP filter.

like | aop <aop:aspectj-autoproxy/>
<tx:advice>

→ No need to remember class name configure as bean definition just by using namespace tag we can.

→ when I write <security:HTTP/> means a bunch of security filter are enable for our application on the Ioc container. and this filter name what springSecurityFilterChain i.e. reason we dont have flexibility of naming it the name is standard and fixed i.e. reason our delegating filter proxy name also should be springSecurityFilterChain only. so we cannot change.

→ we just write tag based on the tag it automatically creates as bean definitions.

② <security:basic/> (or) form (or) digest we need to tell using that tag.

→ How many resources are there for our application

resources are there
which is public resource or protected resource

home.htm → public resource
account.htm → protected resource.

<security:intercept-url pattern="/account.htm" access="fullAuthenticated()"/>

(Intercept this url)

② "pattern" of these what /account.htm, who can "access" this "hasAuthority" (if user has this authority) (or) "fullAuthenticated" (i.e. if user is logged in only user logged-in user can only access), these are the predefined tokens i.e. provided by whoom spring_security who tell us the access rights we are telling.

what are the tokens Allowed

① permit-All → Any one can access.

② FullyAuthenticated() → only logged in user only access.

→ ③ (hasAuthority OF roles → i.e.)

③ hasAuthority ("accountHolder")
↓
any existing role of our application, role-based access ~~has any authority~~.

④ hasAnyAuthority("accountHolder", "staff").
i.e. multiple people → multiple roles of user can access.

③ these are the named tokens we used for providing the access rights for accessing the resource of our application. Hey security filter intercept every url, if the url pattern is /account.htm allow the people access which people fully Authenticated user can only access this resource.

user must be logged in.

Access tokens are through which we restrict the resources based on roles.

Resource url which user can access.

---

(3) we are providing this information security filter next resources we have to intercept

```
<security: intercept-url pattern="/home.htm" (who can access) access="permitAll" />
```

with we compile now (securityfilterchain(), loginhandler, httpconfig) over } All component complete. just by writing 3 tags.

---

(4) what else we need (Identity store, authentication provider, authentication-manager).

let start configuring these classes as bean definitions.

→ authentication-Manager class name we dont know

<security: authentication-manager /> → } this tag

Note Internally it will configure the underlying class as the bean definition.

→ Now authentication-Provider what we need to inject authentication-provider.

<security: authentication-provider /> →

TO authentication-provider what we need to inject authentication-service i.e. depends on identity-store we are using there are different types of authentication services are there.

→ Depends on the identity store we are using different types of authentication-services.

right know we are using identy-store as that inmemory.

i.e. reason there is authentication-service called user-service that means user provided values.

<security: user-service/>

→ TO this user service we are passing what inmemory right so giving the ~~in~~memory user inputs username and password to user service.

<security: user username="joe" password="welcome1" authorities="accountholder" />
<security: user username="bob" password="welcome1" authorities="manager" />

← inmemory database

→ authorites of the user the user can be which role.

the user can be multiple roles.

(identity store what will configure user name, password, roles.)

→ How many no of user are there that many no of user we are entering.

this called inmory -database.

→ that means we are populating usernames and passwords in the components of our class which class (user-service) class itself we are storing. we are not storing some where else.

→ during startup of our application user-service class will be created with loading ther username, password, roles which we configured. and that user-service will be injected into whoom authentication-provider.

→ authentication provida will be injected whom authentication-manager.

→ password encoder not required we will configured we are working.