**SPRING SECURITY**

Four Important features in Spring Security:

* Authentication and Authorization
* Confidentiality
* Integrity
* CORS and CSRF

Authentication and Authorization

Authentication is telling application who we are, and authorization is all about how much access we have in the application.

Different components in spring security:

When user first time logs in Authentication filter checks whether he is authenticated or not(by token), if he is not authenticated(for first time logins) Authentication filter sends Authentication Manager and it sends to Authentication provider and Authentication provider uses User Details service and password encoder to check the user details are correct or not, if correct it stores the details in security context, and when again the user sends any request the Authentication filter uses Security context to get the details.

Graphical user interface, application, PowerPoint

Description automatically generated

**Lesson 1:**

* Add Spring security dependency to project and write an API in controller.
* Hit the URL in the postman we get 401 because by adding security dependency itself our application will be configured to default security with a user and password.
* Now in postman in basic authentication section add username and password then you will get the response.
* Important to note, we get JSESSIONID with the response when we are first time hitting the URL.
* So AuthenticationFilter for first time travel along all components and generated the JSONSESSIONID and store it in security context.
* Next time when we hit the URL, authentication filter checks weather the request has any JSESSIONID, if have it will check in security context and if it matches then it will allow the request to proceed without any travelling to AuthenticationManager and to other components.
* Even if you provide wrong username and password with correct JSESSIONID the request will get the response because AuthenticationFilter checks in SecurityContext and says its fine.

**Lesson 2** :

We can create our own UserDetailsService and we can create a user with username and password(encrypting using passwordEncoder) and add that user to UserDetailsService.

Graphical user interface, text, application, email

Description automatically generated

Point to note we can create passwordEncoder object in the configure method or we can declare bean method. When we declare a bean method the AuthenticationManger checks for PasswordEncoder bean in spring context and uses it automatically.

Graphical user interface, text, application

Description automatically generated

**Lesson 3** :

* We can create a custom AuthenticationProvider class which implements AuthenticationProvider class of Spring security. Then we must override two methods.
* First method has a parameter from which we can get name and password and compare the details from database.
* In second method we need to tell AuthenticationManager what type of class this custom auth provider returns.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**OAUTH:**

OAuth is open authentication and authorization framework.

It is federated Authentication(authentication can be happen in single place) and delegated authorization(only some part of authorization like facebook we have access for profile only).

A picture containing text, businesscard

Description automatically generated

**Grant Types:**

Authorization codes

Password

Client Credentials

Refresh token

Graphical user interface, application, table, PowerPoint

Description automatically generated

Graphical user interface, application, table

Description automatically generated

Graphical user interface, application, table, PowerPoint

Description automatically generated

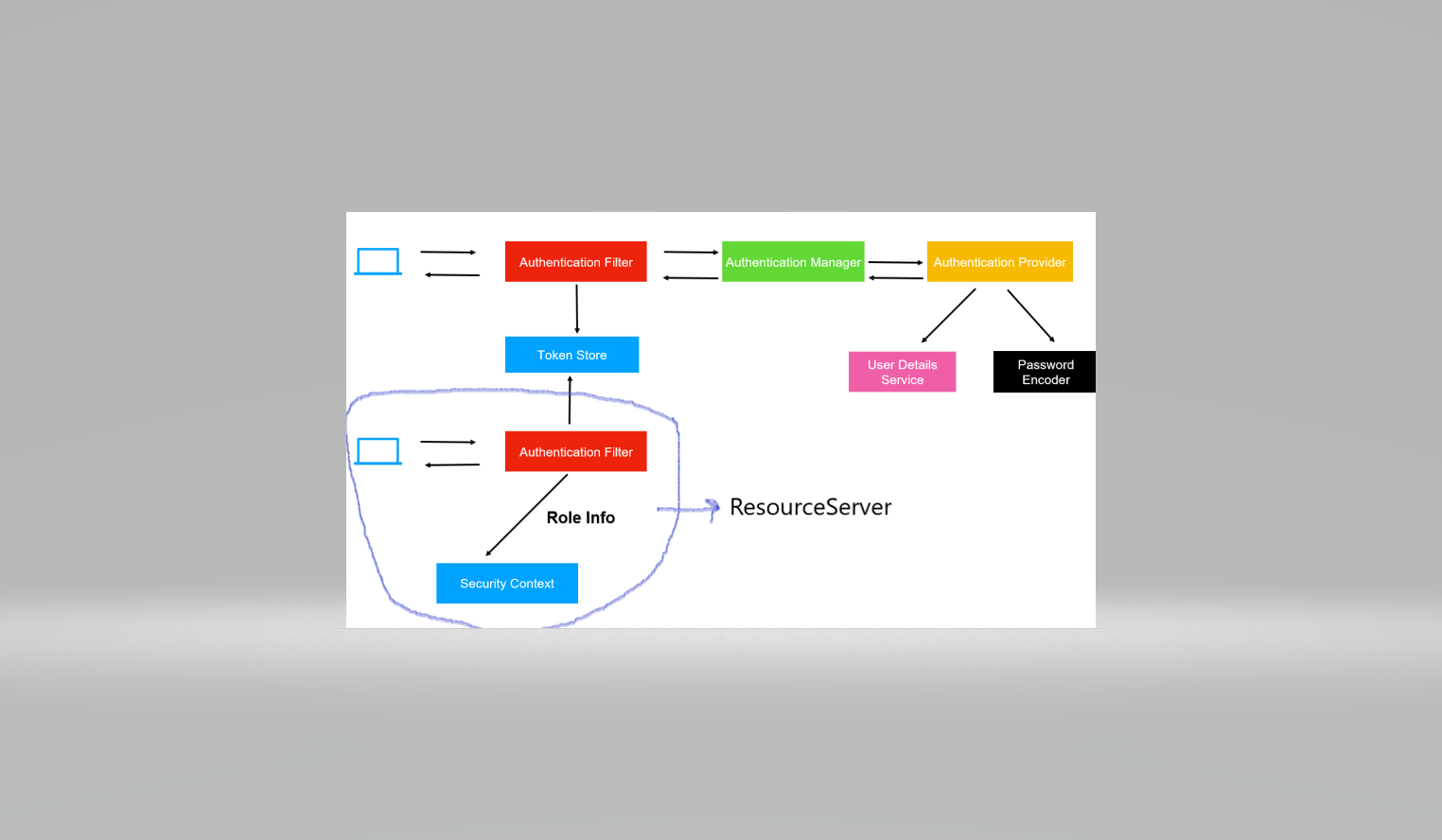
Graphical user interface, application, table, Excel

Description automatically generated

Password grant approach in the course:

First we hit the oauth server <http://127.0.0.1:9091/oauth/token> with requestbody and client details(username and password) and the oauth server gives the token.

With that token we can hit couponservice, then resourceserver/AuthenticationFilter in couponservice checks the token with authserver, authserver checks whether the token is valid, and token is present in token store or not. After that resourceserver get details about user from UserDetailsService checks the roles and stores them in Security context.



**Signing the access Token:**

We must sign an accessToken with some private key.

This can be done in 2 ways:

* Asymmetric
* Symmetric

Asymmetric : While creating the acessToken we will sign it with a private key in auth-server or any where in the code, while reading or decoding the accessToken we use public key to decode.

Symmetric : We sign the accessToken and we read the accessToken with same key.

**OpenId connect** : Open Id or Identity server adds an extra layer on auth server. When we are taking to Auth server, we can request auth server to give information about the user along with access token. So basically, then we get access token and open id information.

**Keycloak** : To start the keycloak server go to the bin and execute below command.

***./standalone.bat***

To change the default port number from 8080 to another port use below command

***./standalone.bat -Djboss.socket.binding.port-offset=100***

Then the port will be 8180

To see the users : <http://localhost:8080/auth/realms/appsdeveloper/users>

[http://localhost:8080/auth/realms/{realm-name}/users](http://localhost:8080/auth/realms/%7brealm-name%7d/users)

admin username:vishal

admin password:vishal

username: [vishalpalla27@gmail.com](mailto:vishalpalla27@gmail.com)

password: vishal123

Scope : Scope is a mechanism in OAuth 2.0 to limit an application’s access to a user’s account. An application can request one or more scopes, this information is then presented to the user in the consent screen, and the access token issued to the application will be limited to the scopes granted.