**Spring Security – OAuth 2.0 Mechanism**

OAuth 2.0, which stands for Open Authorization, is a standard designed to allow a website or application to access resources hosted by other web apps on behalf of a user.

**OAuth 2.0 Roles**

* Resource Owner - The user or system that owns the protected resources and can grant access to them.
* Client - The Client is the system that requires access to the protected resources. To access resources, The Client must hold the appropriate Access Token
* Authorization Server - This server receives requests from the Client for Access Tokens and issues them upon successful authentication and consent by the Resource Owner.
* Resource Server - A server that protects the user's resources and receives access requests from the client. It accepts and validates Access Token from the client and returns the appropriate resources to it.

The application (Client) asks for the resource to the Resource Server and in response to it, it will ask the access token to it. The access token is provided by the Authorization Server only after checking whether the user (Resource Owner) is valid or not. After getting the access token, the resource server will check whether it is a valid token or not. If it is valid, it will grant the resource.

A diagram of a process

Description automatically generated

Demo Project was done to illustrate the OAuth Mechanism

In our project, The Client is our Spring boot Application through which we are requesting the API. The Resource Server is GitHub. Whenever the /secured API is hit, it will be redirected to GitHub. After the authentication of the Resource Owner ie user using Username and Password, the Authorization Server will issue the access token to API call(Spring boot Application). Using that we can access the API.

**Dependencies used for the Project:**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-oauth2-client</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

**Security Configuration Class:**

In Spring Security, the SecurityFilterChain is a fundamental concept used to configure the security filters that handle authentication and authorization in a Spring Boot application. It allows you to define a sequence of filters to be applied to incoming HTTP requests to enforce security policies.

The SecurityFilterChain is typically configured in the SecurityConfigurerAdapter class, and it's part of the overall Spring Security configuration. It's used to set up the security filters that handle various aspects of security, such as authentication, authorization, and CSRF protection.

In this class, we are giving access to the all the API’s which ends with “/” and remaining all the APIs should be authenticated. The .oauth2Login(withDefaults()) configures OAuth 2.0 login support. The withDefaults() method provides default settings for OAuth 2.0 login.

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

return http

.authorizeHttpRequests(auth -> {

auth.requestMatchers("/").permitAll();

auth.anyRequest().authenticated();

})

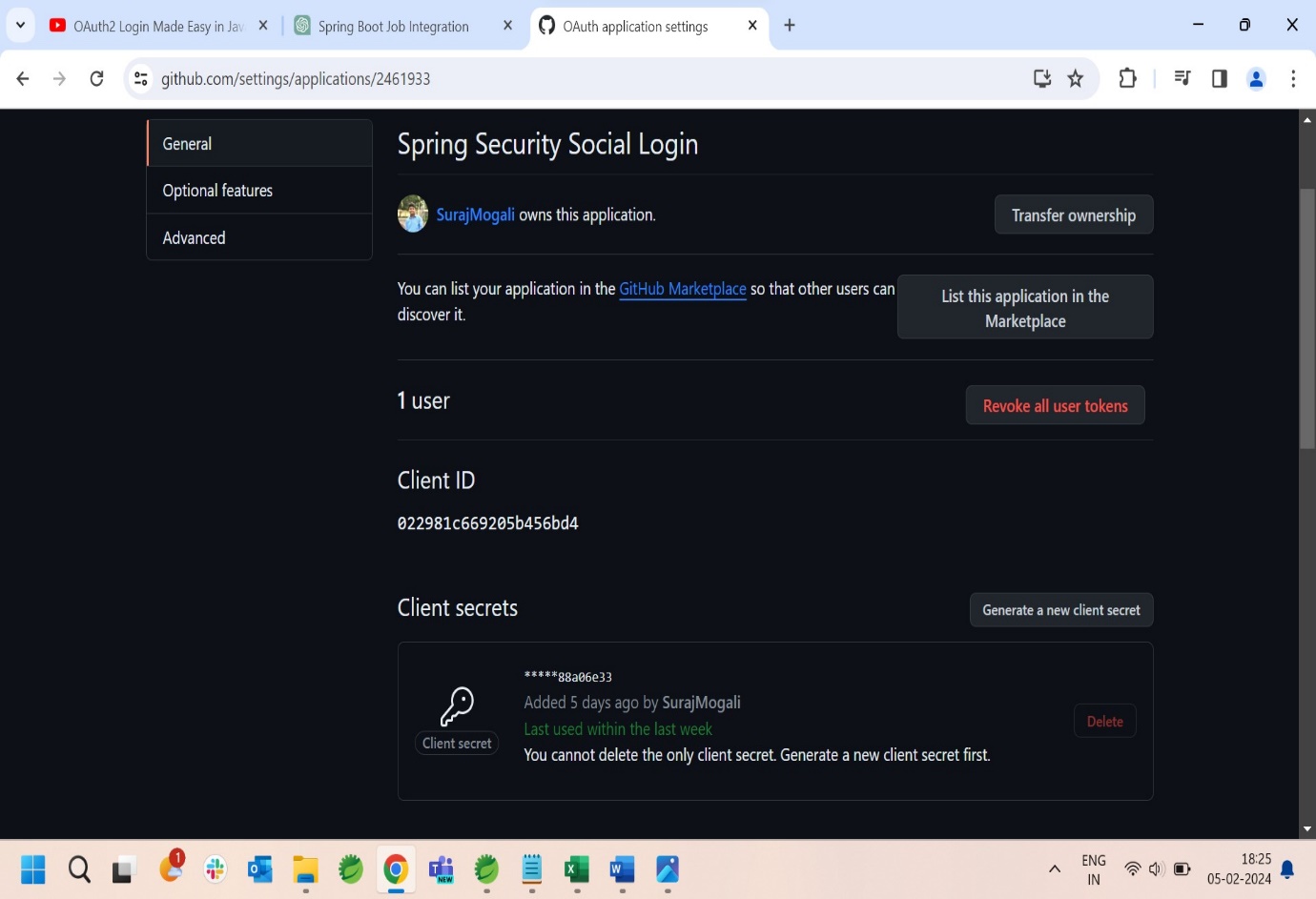
. oauth2Login(withDefaults()) //Configures OAuth 2.0 login support. The withDefaults() method provides default settings for OAuth 2.0 login.

. build ();

}

After this, for GitHub authentication, there are steps where we should follow with which we can redirect our application to GitHub Login

* Go to Settings
* Developer Settings
* OAuthApps
* Give the name for the OAuth application.
* Give the Home Page Url
* Give the callback Url (The api will be redirected to GitHub here)
* Register the application.
* Get the Client ID
* Generate the Client secret.



**Application Properties file:**

Configure the GitHub ClientID and Secret Client in the application properties file of the SpringBoot Application.

spring. security. oauth2.client. registration.github.client-id=022981c669205b456bd4

spring. security. oauth2.client.registration.github.client-secret=29d82ab83ee1de19d3ad1b6f77e6799b88a06e33

logging.level.org.springframework.security=TRACE

Hit the localhost:8080 api, the response will be "Hello, Home!" and when the /secured endpoint is hit,it will be redirected to /login uri and the GitHub Login page will be displayed. Login through the UserName and Password and we will get the desired response.