**Spring cloud Security With Oauth2**

Spring Security is a framework which provides various security features like: authentication, authorization to create secure Java Enterprise Applications.

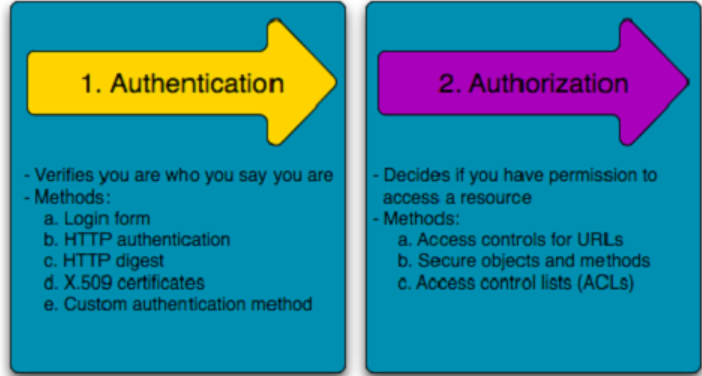
### What is Authentication?

Authentication refers to the validation of credentials such as user ID/ user name / password, etc. in order to verify the identity of any given user.

### What is Authorization?

Authorization takes place once the user identity has been validated and successfully authenticated by any given system. Authorization allows a user to use system resources of the likes of information, databases, funds, locations, permission.

We can apply authorization to authorize web request, methods and access to individual domain.



**Default Security Setup**

Spring Boot application, we need to add the security starter dependency:

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-security</artifactId>  </dependency> |

Step 1: Create class as SecurityConfig class add annotation EnableWebSecurity and extends abstract class of WebSecurityConfigurerAdapter and overload configures methods.

In Memory configuration

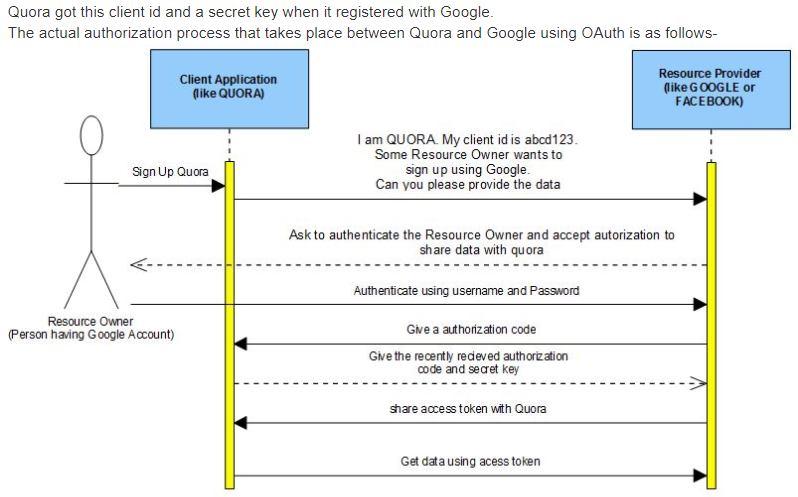
|  |
| --- |
| **import** org.springframework.context.annotation.Configuration;  **import** org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  **import** org.springframework.security.config.annotation.web.builders.HttpSecurity;  **import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  **import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  **import** org.springframework.security.crypto.factory.PasswordEncoderFactories;  **import** org.springframework.security.crypto.password.PasswordEncoder;  @Configuration  @EnableWebSecurity  **public** **class** SecurityConfig **extends** WebSecurityConfigurerAdapter {  @Override  **protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {  PasswordEncoder encoder = PasswordEncoderFactories.*createDelegatingPasswordEncoder*();  auth.inMemoryAuthentication().withUser("nag").password(encoder.encode("nag")).roles("user");  auth.inMemoryAuthentication().withUser("admin").password(encoder.encode("admin")).roles("user","admin");  }  @Override  **protected** **void** configure(HttpSecurity http) **throws** Exception {  http.csrf().disable().authorizeRequests().antMatchers("/product/user/\*\*").hasAnyRole("user", "admin").and().formLogin();  http.csrf().disable().authorizeRequests().antMatchers("/product/admin/\*\*").hasAnyRole("admin").and().formLogin();  }  }  With chaining:  PasswordEncoder encoder = PasswordEncoderFactories.*createDelegatingPasswordEncoder*();  auth.inMemoryAuthentication().withUser("nag").password(encoder.encode("nag")).roles("user") .and().withUser("admin").password(encoder.encode("admin")).roles("user", "admin");  http.csrf().disable().authorizeRequests().antMatchers("/product/user/\*\*").hasAnyRole("user", "admin").and().authorizeRequests().antMatchers("/product/admin/\*\*").hasAnyRole("admin").and().formLogin();  To discard the security auto-configuration and add our own configuration, we need to exclude the SecurityAutoConfiguration class.  @SpringBootApplication(exclude = { SecurityAutoConfiguration.class })  public class SpringBootSecurityApplication {    public static void main(String[] args) {  SpringApplication.run(SpringBootSecurityApplication.class, args);  }  } |

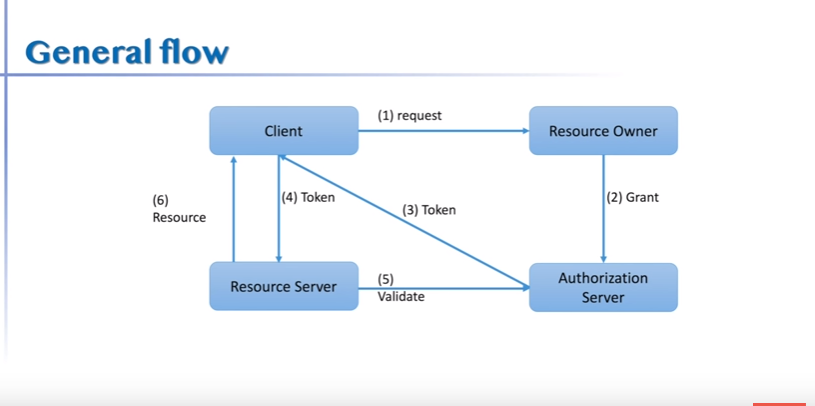
OAuth stands for Open Authorization. It’s a free and open protocol and OAuth2 is an authorization framework, it enable the application to get the limited access to user accounts on an http service such ad facebook, github, google, linkedlin.

It authorize the third party applications to access the user account.

In the above example of Quora, we have 3 actors-

* **Resource Owner** - This is the user who wants to sign up using Quora.
* **Client Application** - This will be Quora
* **Resource Server** - This will be Gmail or Facebook.
* **Authorization Server** - The resource server hosts the protected user accounts, and the authorization server verifies the identity of the user then issues access tokens to the application.





. 