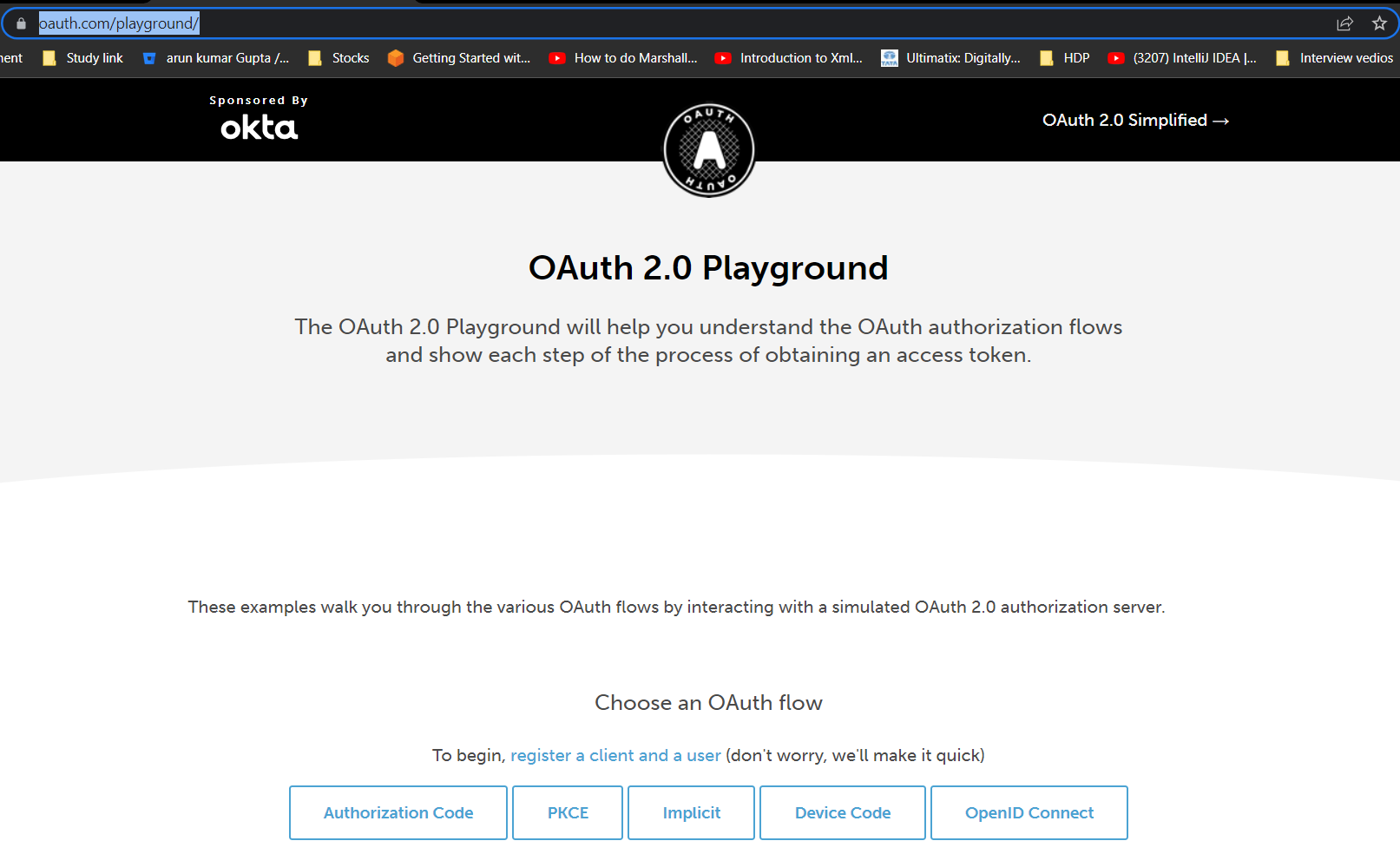
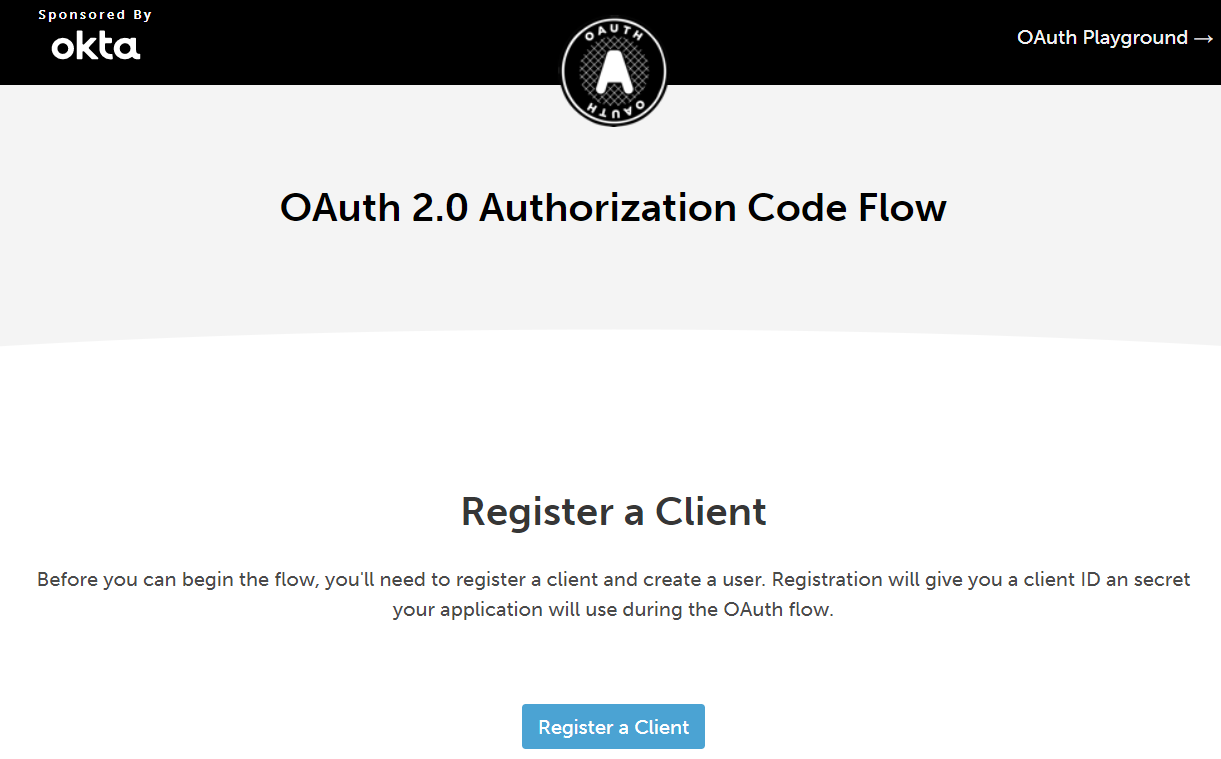


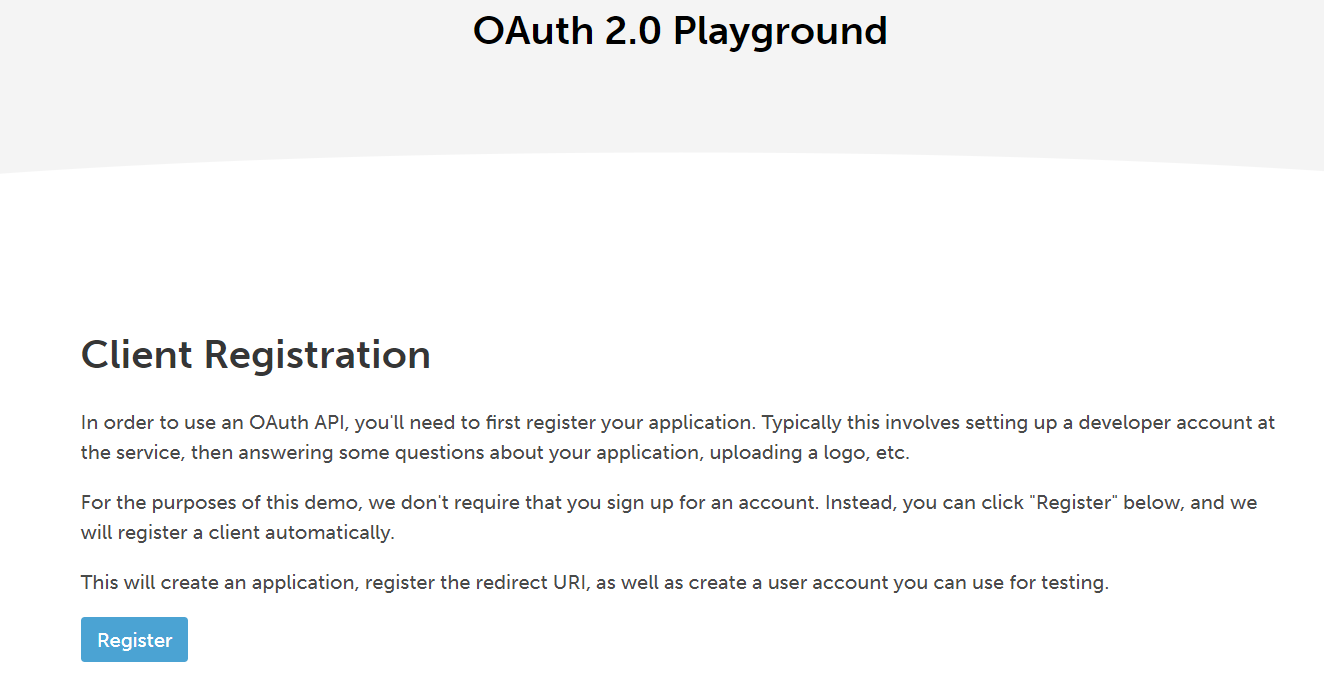
For more detail: <https://www.oauth.com/playground/>

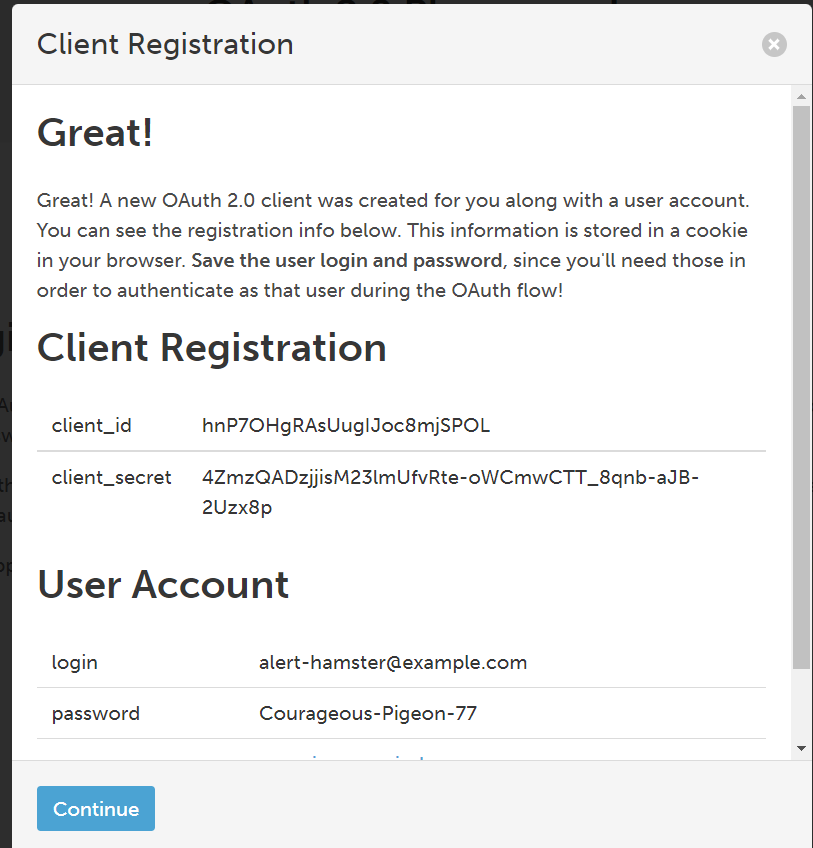


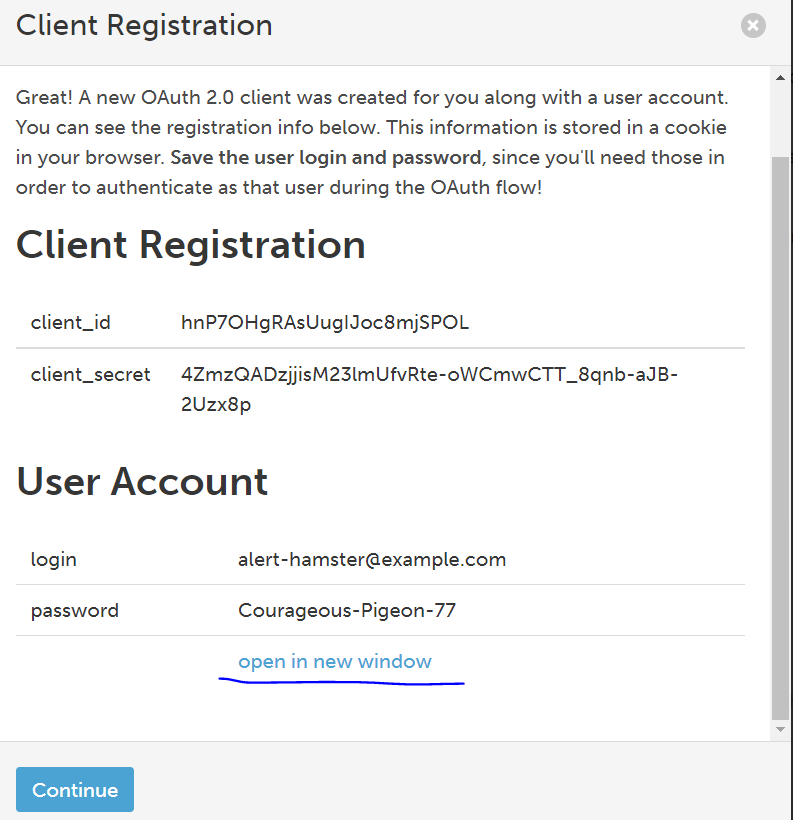
Main flow is : Authorization Code & OpenID Connect

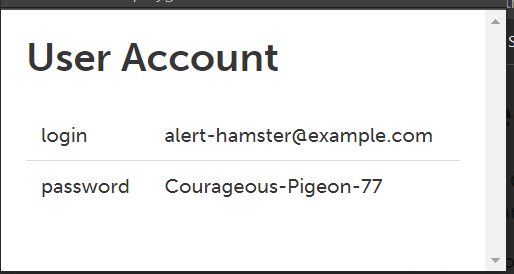
Click on: Authorization Code:

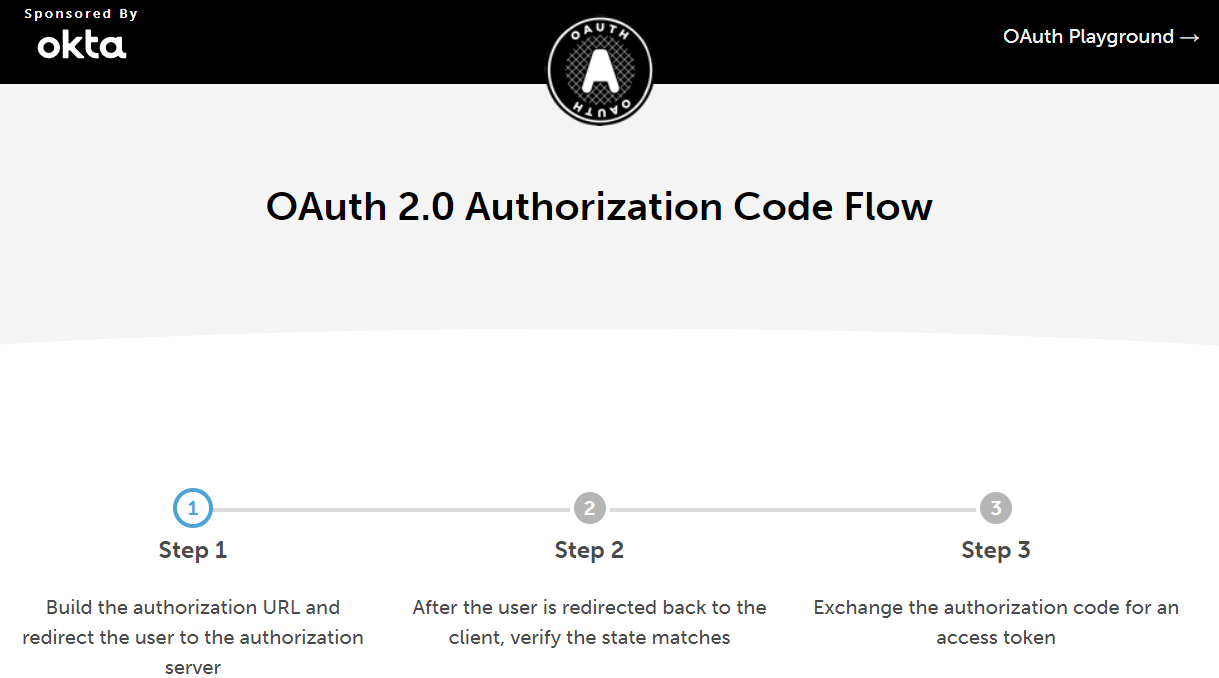


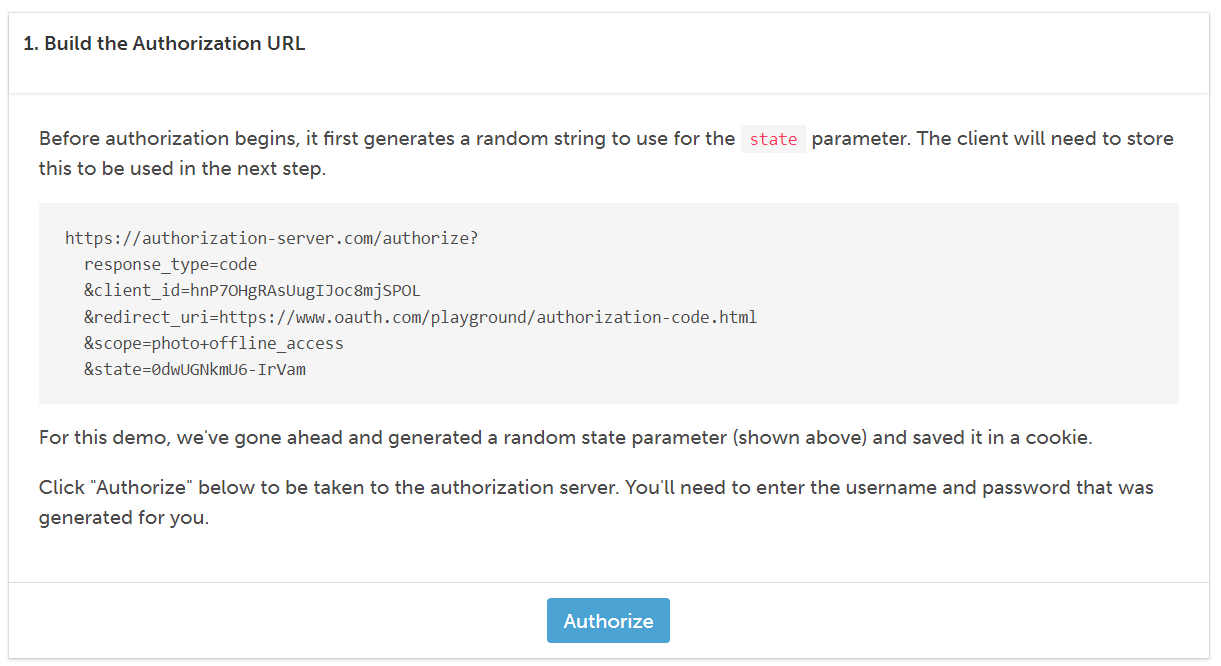




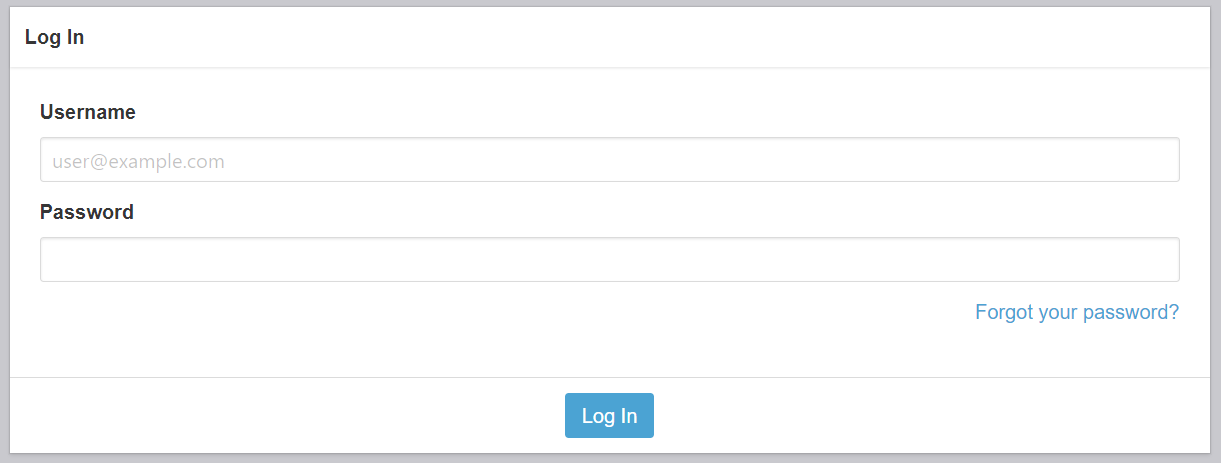


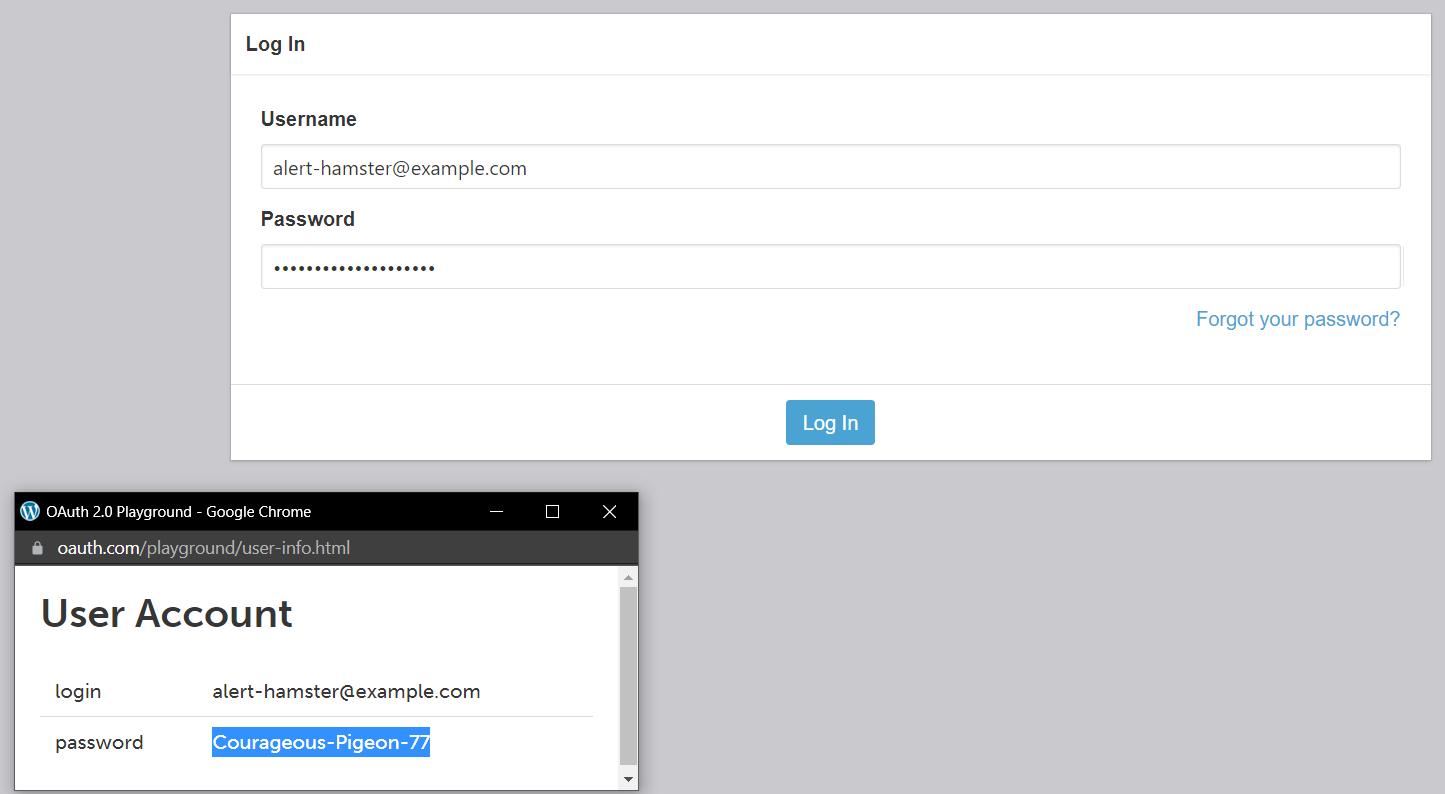


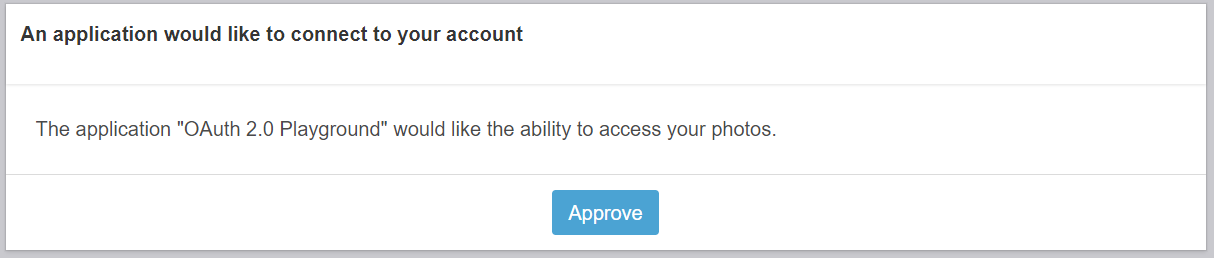


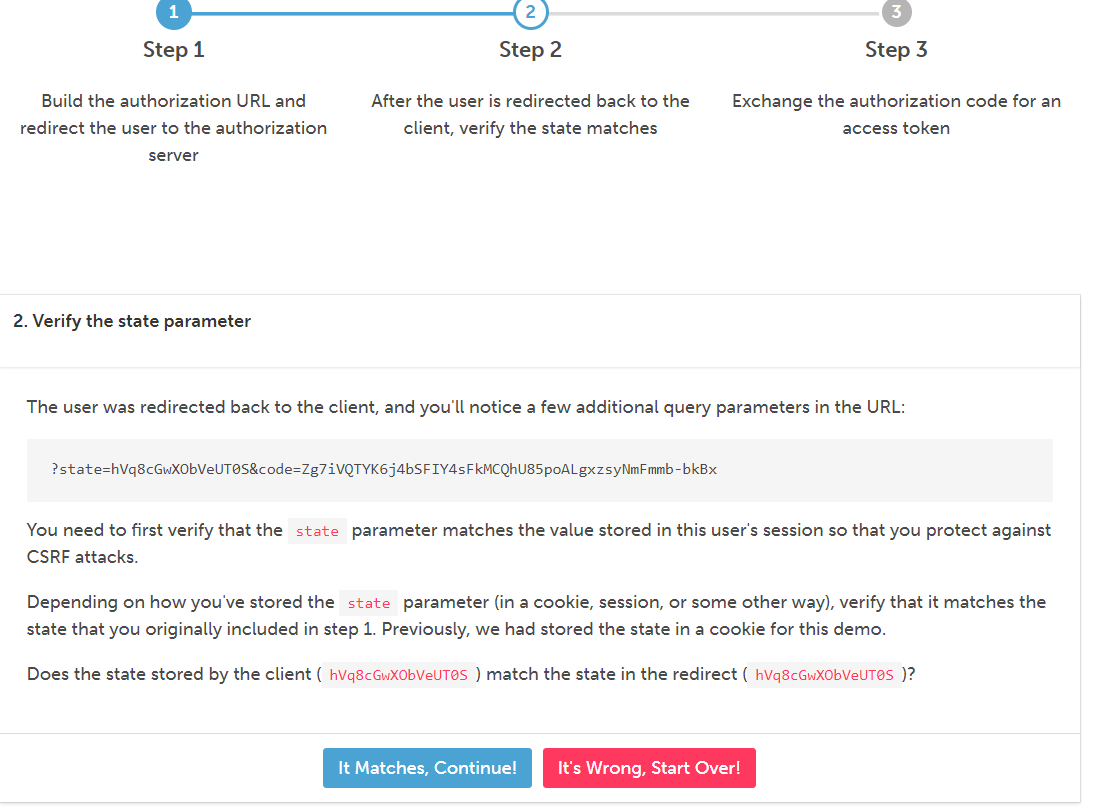


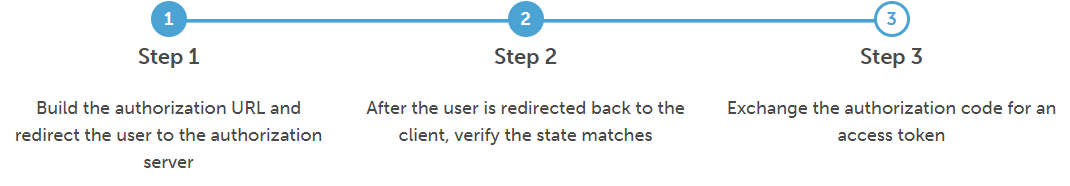
|  |
| --- |
| https://authorization-server.com/authorize?  response\_type=code  &client\_id=hnP7OHgRAsUugIJoc8mjSPOL  &redirect\_uri=https://www.oauth.com/playground/authorization-code.html  &scope=photo+offline\_access  &state=0dwUGNkmU6-IrVam |

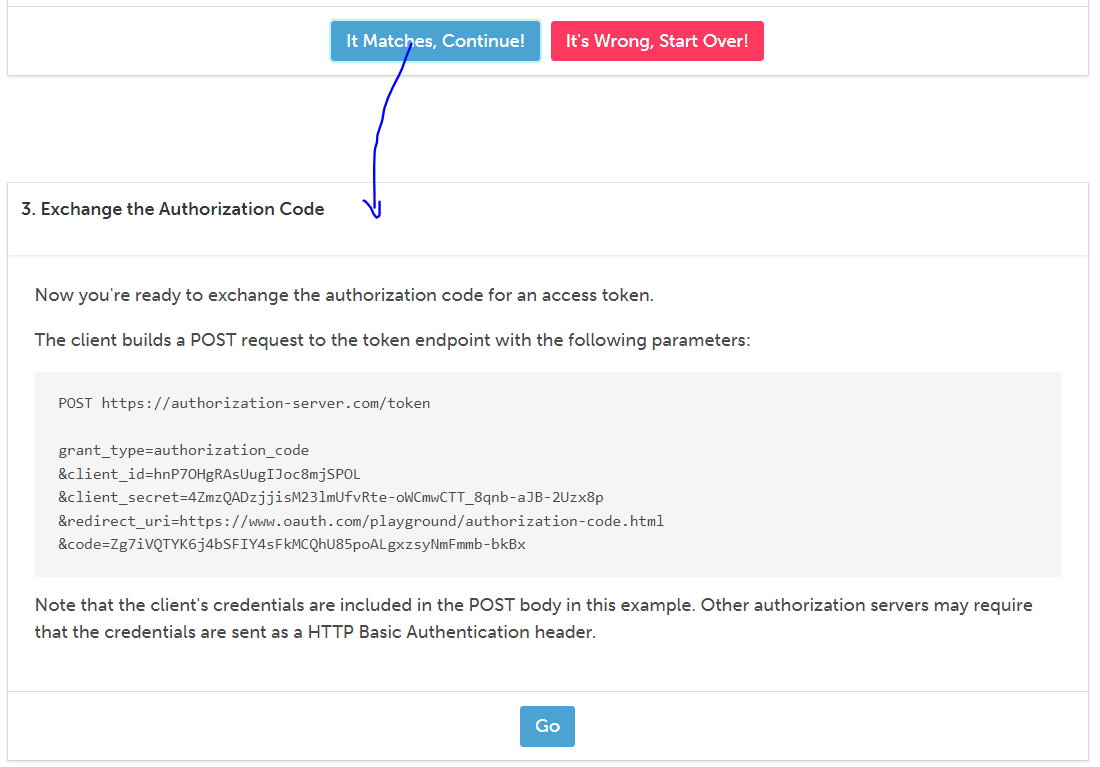


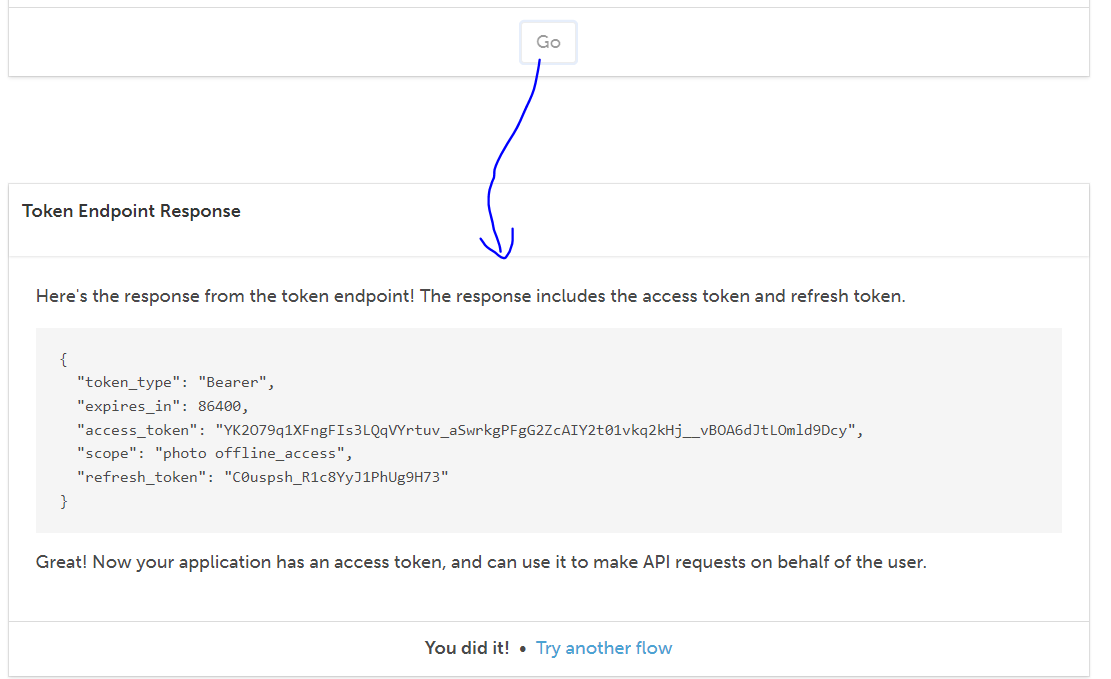


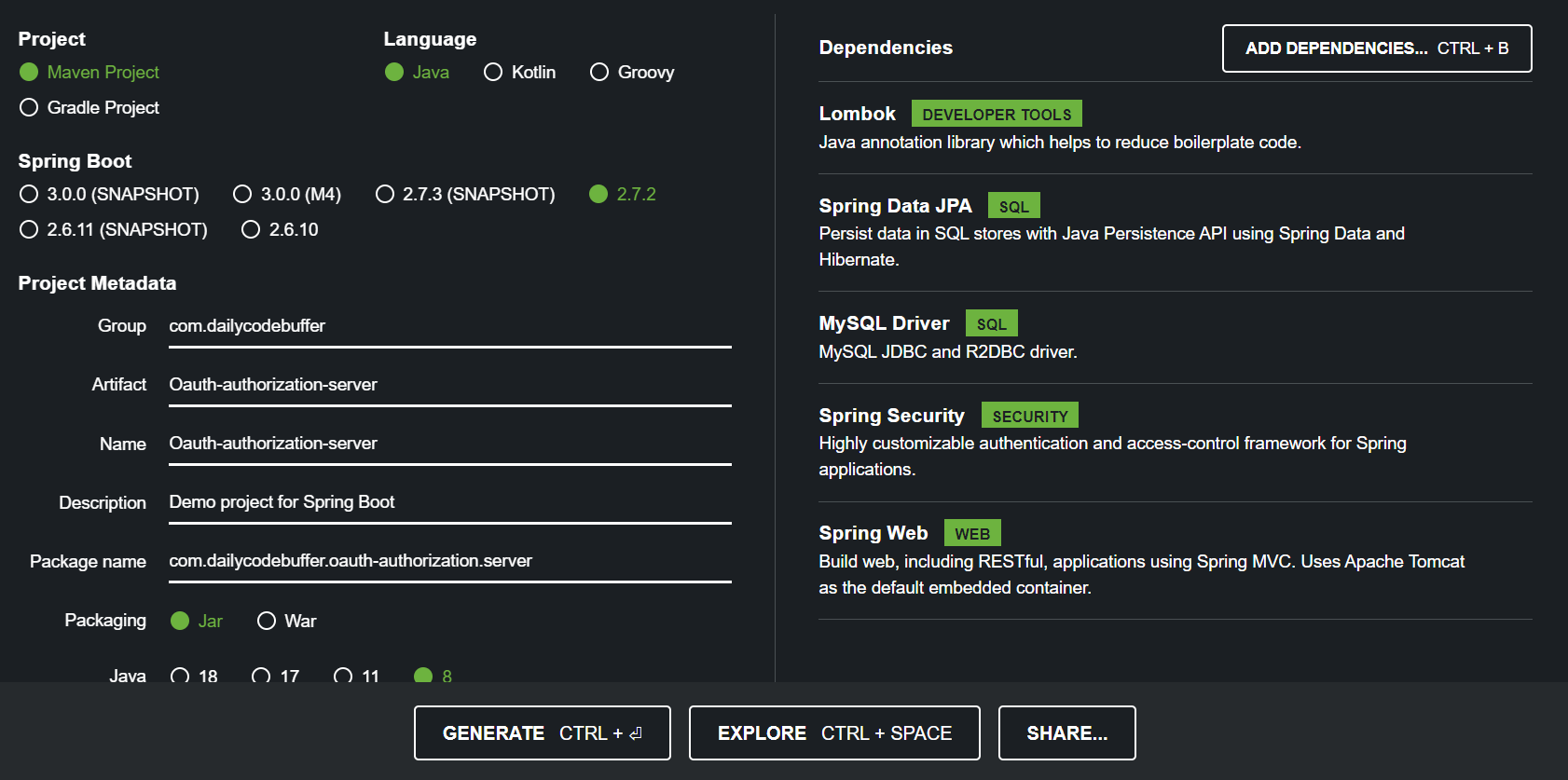












Added one more dependency:

|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-oauth2-authorization-server</**artifactId**>  <**version**>0.3.1</**version**> </**dependency**> |

Step 1: Load the user from the DB and redirect to spring security

/\*

**Step2-**

Now needs to implement the configuration i.e. we need to add configuration for our authorization server. Note this implementation would be standard for all the

Authorization servers so here we just need to use the bunch of code which already written,

We just need to use here. Here we will create public key, private key and others to handles authorization server so that our clients can connect to it and exchange between the authorization codes token can be happen. Now let's create the 'AuthorizationServerConfig' in the config package.

\*/

**Step3-** Register the client to Authorization server using standard code

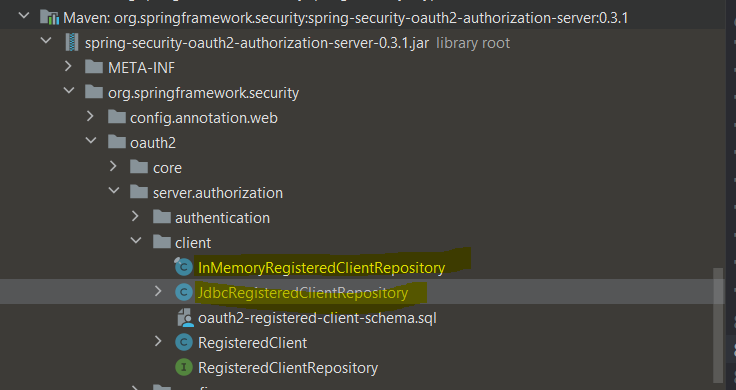
**Step4-** Add all the detail of Authorization server in the client’s application.yml to connect to Authorization server.

Currently here we have only one client i.e. 'SpringSecurityClient' which i want to register

In the Authorization Server so the above information will be added in the client's application.yml

So that client could connect to authorization server.So here as we can see in the above method unique client id (api-client), clientSecret, AuthenticationMethod, GrantType (to be accessed), redirectUrl, scope etc. Same property will be added in yml file

Note: Since we have only one client to be registered that is why we have added only one static method RegisteredClientRepository. But if we want to make it dynamically then we will use JDBC Registration client (JdbcRegisteredClientRepository) here



Step4- Next step is to configure public and private key, which is nothing but the standard configuration for public and private key.

// so this all about authorization server configuration

Step-5:-Now let's do the configuration for Basic spring security, so let's add the default basic security configuration

// let’s create on class 'DefaultSecurityConfig'

Here in the above method we are change the authorization request .i.e all the request must be authenticated with formLogin.

// Now one thing is left that is 'authentication provider' like authentication manager, how you should be managing your authentication and for that we will create the CustomAuthenticationProvider for userid and password i.e.(email and password)

Step6- Create the CustomAuthenticationProvider in the service package for authenticating User credential

//Step-7.1

//Step-7.2

After step 7, now Authorization server is ready, all the configurations are done, handeled all the request. Now we need to register our client to talk to this authorization server. Every Authorization server will give the detail to handle everything in your system. So let's go to AuthorizationServerConfig ==> RegisteredClientRepository and take the detail from there.

Now let's go to our client spring-security-client module and add configuration in application.yml

// Step-8

|  |
| --- |
| **spring**:  **security**:  **oauth2**:  **client**:  **registration**:  **api-client-oidc**:  **provider**: spring  **client-id**: api-client  **client-secret**: secret  **authorization-grant-type**: authorization\_code  **redirect-uri**: **"http://127.0.0.1:8080/login/oauth2/code/{registrationId}"**  **scope**: openid  **client-name**: api-client-oidc  **articles-client-authorization-code**:  **provider**: spring  **client-id**: api-client  **client-secret**: secret  **authorization-grant-type**: authorization\_code  **redirect-uri**: "http://127.0.0.1:8080/authorized"  **scope**: api.read  **client-name**: api-client-authorization-code  **provider**:  **spring**:  **issuer-uri**: http://DESKTOP-B1DPRDQ:9090 |

Step-9: Add the dependency in the Clint’s application pom file

|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-oauth2-client</**artifactId**> </**dependency**> |

Step-10 : Add antMatchers and authorised all the resourses via loginForm.

Now let’s start Authorization server and client application and try to hit

<http://localhost:8080/api/hello>

it will redirect to the login page : Give the username : [Brijesh@gmail.com](mailto:Brijesh@gmail.com) and password 1234

and now it will allow us to access the resources and we will get output:

Hello Brijesh@gmail.comfrom hello Controller

Hence we have created Authorization server and we are authorize the client’s resources to be accessed via Authorization server.

So we have implemented OAuth2 with OpenID Authorization server and client as well so both of them can interact with each other and only the authorised

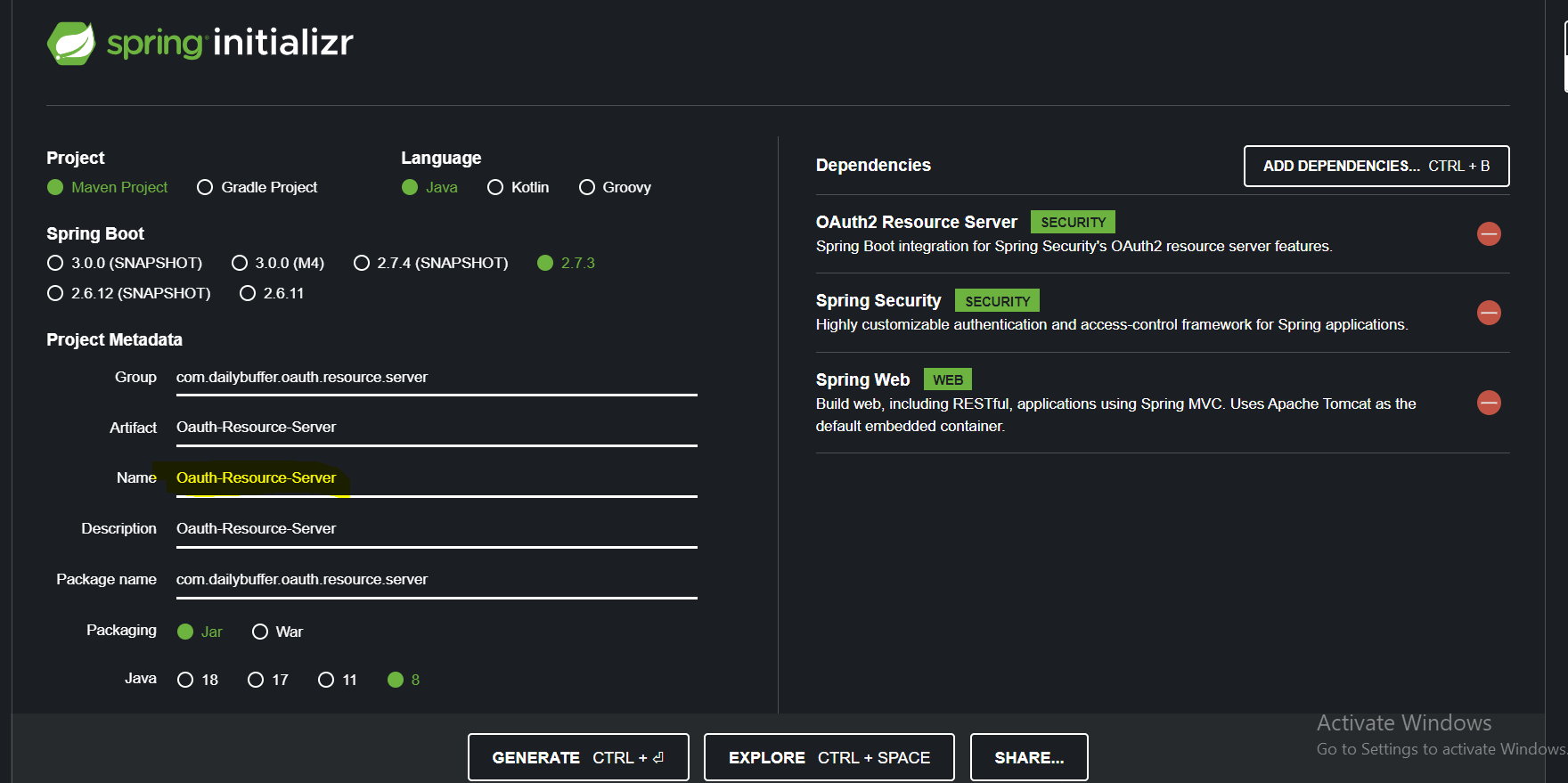
resources are allowed.

Note: Implement Resource Server:

Now what if we have the Resource server as well, where only the authenticated user can have the access to authenticated resources from the client.

Let’s create the Resource Server and see how we can access the resources from the Resource server as well.

Let’s create the spring boot project



Add the module in the paraent pom:

<**modules**>  
 <**module**>spring-security-client</**module**>  
 <**module**>Oauth-authorization-server</**module**>  
 <**module**>Oauth-Resource-Server</**module**>  
 <**module**>StudentServiceSecurityClient</**module**>  
</**modules**>

Let’s configure the Oauth-Resource-Server as well.

1. Add the configuration for spring security in the application.yml of resource server.

|  |
| --- |
| spring: security: oauth2: resourceserver: jwt: issuer-uri: http://auth-server:9000 |

Se we have added the Authorization server configuration

1. Let’s add the Basic spring security configuration as well. Create the config package in Resource server > create the class-> ResourceServerConfig.java
2. Now let’s create the controller api having resources

|  |
| --- |
| @GetMapping("/api/users") **public** String[] getUser(){   **return new** String[]{"Arun", "Tarun","Atharv","Radhika"}; } |

1. Now to access the above Resource server api (resources), we need to call this api in our client application and it has to be authenticated and it has to be with in the scope defined in the ResourceServiceConfig.java i.e. When the scope is matched then only allowed to access the resource. So we need to configure our client to handle the web client . So we will b using web client to call this particular url to fetch the data.

Let’s configure web client :

Go to spring-security-client > Pom.xml and add the below dependencies

|  |
| --- |
| **<dependency>**  **<groupId>org.springframework.web</groupId>**  **<artifactId>spring-webflux</artifactId>**  **<version>5.3.9</version>**  **</dependency>**  **<dependency>**  **<groupId>io.projectreactor.netty</groupId>**  **<artifactId>reactor-netty</artifactId>**  **<version>1.0.9</version>**  **</dependency>** |

1. Web Client Configuration .Create class> WebClientConfiguration inside the config package
2. Now use the web client in the spring-security-cleint app to access the resource