

1. Keycloak (serveur OAuth2 et OIDC)

keycloak est un serveur d'autorisation **oauth2** et **oidc** basé sur *jboss wildfly* et *java >=8*
Ce serveur est par défaut basé sur une base *H2* (mais peut être configuré pour utiliser une base *mysql* ou *postgres*) et dispose d'une **ihm** intégrée pour configurer des utilisateurs.

1.1. Installation de keycloak

- Télécharger le serveur via cette url:
<https://github.com/keycloak/keycloak/releases/download/14.0.0/keycloak-14.0.0.zip>
- Extraire le contenu de l'archive dans un répertoire (ex: c:\prog\keycloak-14.0.0)
- Modifier si besoin le numéro de port dans *keycloak-14.0.0\standalone\configuration\standalone.xml* (ex: 8080 --> 8989)

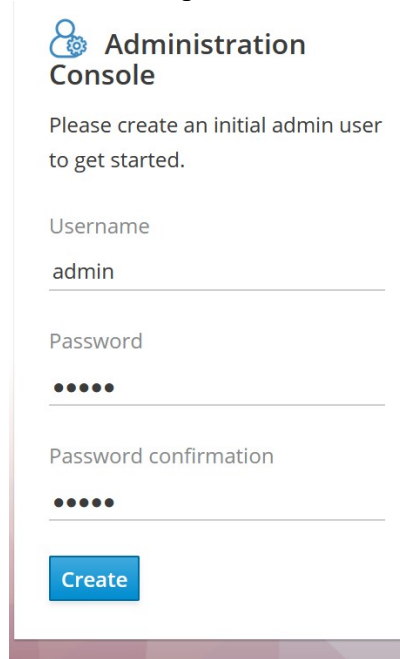
1.2. Démarrage du serveur keycloak :

keycloak-14.0.0/bin/standalone.bat

URL: <http://localhost:8080/auth/> ou bien <http://localhost:8989/auth/>

1.3. Première connexion

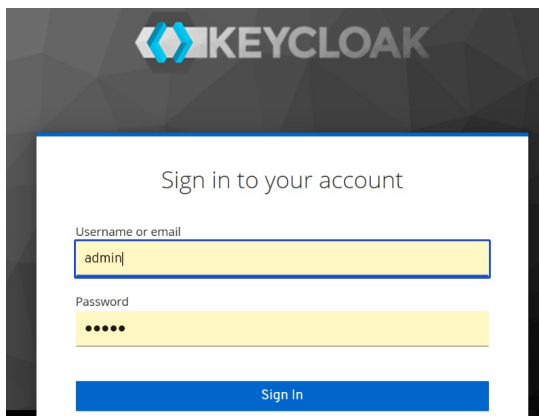
Créer un compte **admin/admin** ou autre lors de la première connexion.



The screenshot shows the 'Administration Console' interface. At the top, it says 'Please create an initial admin user to get started.' Below this, there are three input fields: 'Username' with the value 'admin', 'Password' with masked characters '....', and 'Password confirmation' also with masked characters '....'. At the bottom left of the form is a blue button labeled 'Create'.

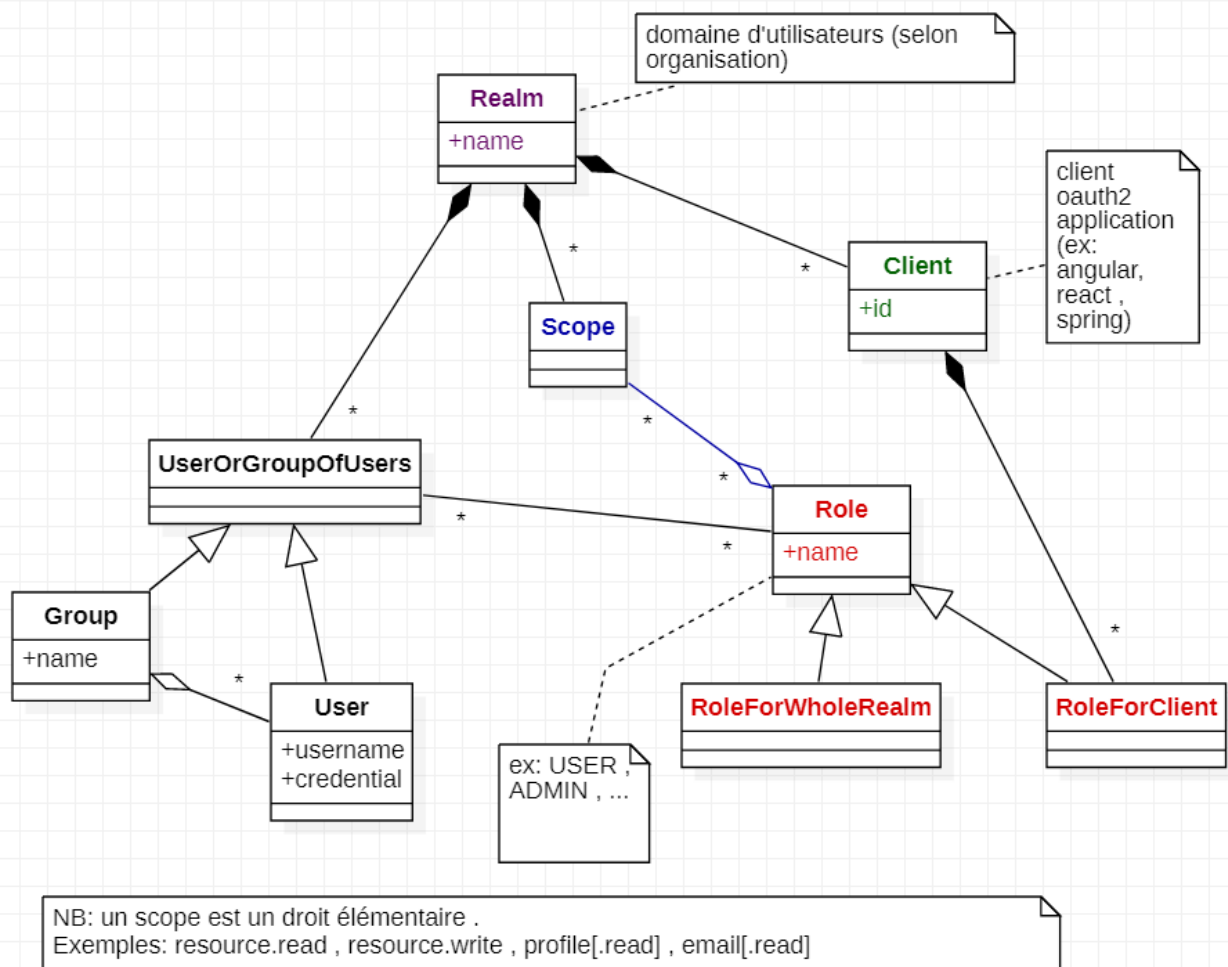
2. Administration de keycloak

Se connecter en tant qu'administrateur (ex : *admin/admin*)



2.1. Structure de la configuration de keycloak

Keycloak configuration (oauth2/oidc server)



2.2. Initialisation d'un nouveau "realm" et "user/password"

S'être préalablement connecté en tant qu'administrateur .

Menu "**master/add realm**"

Add realm

Import	<input data-bbox="606 488 805 544" type="button" value="Select file"/>
Name *	<input data-bbox="603 582 906 649" type="text" value="myrealm"/>
Enabled	<input checked="" data-bbox="603 689 762 757" type="checkbox"/>
	<input data-bbox="603 790 738 857" type="button" value="Create"/> <input data-bbox="746 790 874 857" type="button" value="Cancel"/>

Dans ce realm :

menu "**manage/users**" / "**add user**"

Add user


ID	<input data-bbox="494 1120 855 1176" type="text"/>
Created At	<input data-bbox="494 1198 855 1254" type="text"/>
Username *	<input data-bbox="494 1276 855 1332" type="text" value="user1"/>
Email	<input data-bbox="494 1355 855 1411" type="text" value="jean.bon@mycompany.com"/>
First Name	<input data-bbox="494 1433 855 1489" type="text" value="jean"/>
Last Name	<input data-bbox="494 1512 855 1568" type="text" value="Bon"/>
User Enabled ?	<input checked="" data-bbox="494 1601 619 1653" type="checkbox"/>

Save

Sélectionner onglet "Credentials" de user1

password=**pwd** or ...

Temporary off

User1 

Details Attributes **Credentials** Role M

Manage Credentials

Position	Type
----------	------

Set Password

Password

Password Confirmation

Temporary ☐ OFF

Save/Set password

Eventuelle vérification de la configuration

- **logout** (as admin) (menu en haut , à droite)
- http://localhost:8080_ou_8989/auth/realms/myrealm/account/#/
- **signIn**
- se connecter en tant que **user1/pwd**
- **verifier/visualiser** (*Personal Info*)
- **sign out**

+ configuration d'un éventuel groupe d'utilisateurs (ex : "admin_of_myrealm")

Manage / Users / View all users / / Edit + rattachement à un group

Exemple :

utilisateur (**admin1/pwd1**) et (**admin2/pwd**) membres du groupe "admin_of_myrealm"

2.3. Initialisation d'un nouveau client (application)

http://localhost:8080_ou_8989/auth/admin

Se reconnecter en tant qu'admin/admin

Sur "**myRealm**"

menu "**Configure / Clients**" (app)

Create

...

2.4. configuration client/app officiel et test :

NB : l'application prédéfinie <https://www.keycloak.org/app/> a été officiellement prévue pour effectuer des tests .

clientID=**myclient** , client-protocol : openid-connect

root url=<https://www.keycloak.org/app/>

Add Client

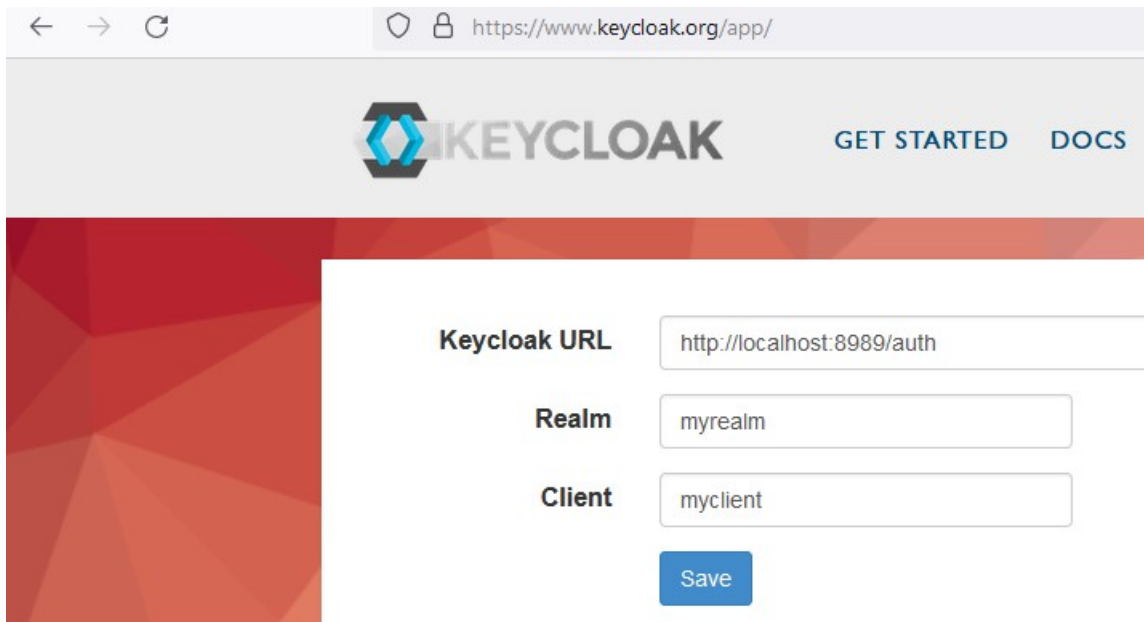
Import	<input data-bbox="512 1066 726 1128" type="button" value="Select file"/>
Client ID * ?	<input type="text" value="myclient"/>
Client Protocol ?	<input type="text" value="openid-connect"/>
Root URL ?	<input type="text" value="https://www.keycloak.org/app/"/>
<input data-bbox="512 1518 616 1581" type="button" value="Save"/> <input data-bbox="627 1518 754 1581" type="button" value="Cancel"/>	

Save

sign out

test :

Se connecter sur <https://www.keycloak.org/app/>



The screenshot shows the Keycloak web interface in a browser. The address bar displays <https://www.keycloak.org/app/>. The page features the Keycloak logo and navigation links for "GET STARTED" and "DOCS". A large red geometric pattern is on the left. The main content area contains a form with the following fields:

- Keycloak URL**:
- Realm**:
- Client**:
- Save**: A blue button.

save

sign in (ex : user1/pwd)

Hello, jean Bon

sign out

2.5. Configuration d'une nouvelle application cliente

S'être préalablement connecté en tant qu'administrateur (ex: <http://localhost:8989/auth/> , console d'administration, admin/admin)

Sur "*myRealm*"

menu "**Configure / Clients**" (app)

Create

Exemple 1 (appli springBoot en mode dev):

client-id: *webappclient1* , client-protocol : openid-connect

root url=<http://localhost:8081>

Exemple 2 (appli angular en mode dev):

client-id: *webappclient2* , client-protocol : openid-connect

root url=<http://localhost:4200>

Save

Settings (exemples) :

name=WebAppClient1 ou WebAppClient2 ou ...

access_type=**confidential** (plutôt pour spring boot) ou **public** (plutôt pour angular)

service_account_enabled=on (true) , authorization_enabled=on (true) ,

Save

NB :Si access_type=**confidential** alors **client-secret** est à récupérer dans onglet "**credentials**"
(exemple: ee03791f-9dfc-49f5-8ec9-fe6b2ba79875)

2.6. Création de nouveaux scopes

S'être préalablement connecté en tant qu'administrateur (ex: <http://localhost:8989/auth/> , console d'administration, admin/admin)

Sélectionner menu "**Myrealm/configure/Client Scopes**"

Create **resource.read** (description= read confidential resource) Save

Create **resource.write** Save

Create **resource.delete** Save

2.7. Configuration de Roles spécifique à une application cliente

S'être préalablement connecté en tant qu'administrateur (ex: <http://localhost:8989/auth/> , console d'administration, admin/admin)

Sélectionner un client (ex ; webappclient1)

In "**Role**" tab:

create new "**USER**" role ans **save**

In "**Mapper**" tab:

add-builtIn / "**username**"

Edit Token Claim Name from 'preferred_username' to "**sub**" (advised for spring oauth2)

save

Sélectionner menu "**Myrealm/configure/Client Scopes**"

Pour chacun des scopes à rattacher au rôle USER , associer role USER du client "webappclient1" dans onglet "scope"

The screenshot displays the Keycloak Admin Console interface for configuring scope mappings. The left sidebar shows the navigation menu with 'Client Scopes' selected. The main content area is titled 'Resource.read' and 'resource.read Scope Mappings'. It features two tabs: 'Settings' and 'Scope'. The 'Scope' tab is active, showing a table with four columns: 'Realm Roles', 'Available Roles', 'Assigned Roles', and 'Effective Roles'. The 'Realm Roles' section is currently empty. The 'Client Roles' section shows the client 'webappclient1' selected. Under 'Available Roles', 'uma_protection' is listed. Under 'Assigned Roles', the role 'USER' is assigned. The 'Effective Roles' column is empty for both sections.

Sélectionner menu "Myrealm/configure/Client/webappclient1"

Sélectionner onglet "client scopes"

Faire passer les 2 nouveaux scopes dans "Assigned Default Client Scopes" ;

[Clients](#) > webappclient1

Webappclient1

[Settings](#) [Credentials](#) [Keys](#) [Roles](#) [Client Scopes ?](#) [Mappers ?](#)

[Revocation](#) [Sessions ?](#) [Offline Access ?](#) [Clustering](#) [Installation ?](#)

[Setup ?](#) [Evaluate ?](#)

Default Client Scopes ?

Available Client Scopes ?

Add selected »

Assigned Default Client Scopes ?

email
profile
resource.read
resource.write
roles

« Remove selected

NB :

La configuration explicitée ci dessus peut être adaptée/simplifiée via des rôles globaux à l'ensemble d'un realm .

1. Menu **myrealm/configure/Roles** pour configurer des rôles de portée globale au realm (ex : ADMIN_CRUD ou ...)
2. Menu **myrealm/configure/ClientScopes** , sélection d'un scope puis onglet **Scope** pour sélectionner les rôles incluant le scope courant .

2.8. Affectation de rôles aux utilisateurs ou aux groupes

En retournant sur partie "Manage / Users" de la console d'admin keycloak,

Users/ user1 / Edit.... / Roles Mappings

Sélectionner si besoin clientRoles: webappclient1 ou webappclient1

et passer "USER" ou "ADMIN_CRUD" de "availables" à "assigned" puis "save".

User1 

Details Attributes Credentials **Role Mappings** Groups Consents Sessions

Realm Roles	Available Roles ?	Assigned Roles ?	Effective Roles ?
	<div>Available Roles</div> <div>Add selected ></div>	<div>default-roles-myrealm</div> <div>« Remove selected</div>	<div>default-roles-myrealm offline_access uma_authorization</div>

Client Roles	Available Roles ?	Assigned Roles ?	Effective Roles ?
<div>webappclient1</div>	<div>uma_protection</div> <div>Add selected ></div>	<div>USER</div> <div>« Remove selected</div>	<div>USER</div>

Admin1 

Details Attributes Credentials **Role Mappings** Groups Consents Sessions

Realm Roles	Available Roles ?	Assigned Roles ?	Effective Roles ?
	<div>Available Roles</div> <div>Add selected ></div>	<div>default-roles-myrealm</div> <div>« Remove selected</div>	<div>ADMIN_CRUD BASIC_USER default-roles-myrealm MANAGE_RW offline_access</div>

2.9. Visualisation des configurations publiquement accessibles

NB: une fois tous les réglages effectués l'URL suivante permet de récupérer certains détails au format JSON:

http://localhost:8080_ou_8989/auth/realms/myrealm/.well-known/openid-configuration

3. Exemples de config d'appli connectées à keycloak

3.1. Exemples de configurations "Client App"

Pour frontEnd "angular/angular-oauth2-oidc"

//npm install angular-oauth2-oidc --save

```
const authCodeFlowConfig: AuthConfig = {
  // Url of the Identity Provider
  issuer: 'http://localhost:8989/auth/realms/myrealm',

  // URL of the SPA to redirect the user to after login
  redirectUri: window.location.origin + "/ngr-loggedIn",

  silentRefreshRedirectUri: window.location.origin + "/silent-refresh.html",
  useSilentRefresh: true,

  postLogoutRedirectUri : window.location.origin + "/ngr-logInOut",
  //ou /ngr-welcome ou ...

  // The SPA's id. The SPA is registered with this id at the auth-server
  // clientId: 'server.code',
  clientId: 'webappclient2',
  //clientSecret if necessary (not very useful for web SPA)
  //dummyClientSecret: 'ee3f886b-0b4d-4529-9d0c-e61ca4b91d96',
  responseType: 'code',

  // set the scope for the permissions the client should request
  // The first four are defined by OIDC.
  // Important: Request offline_access to get a refresh token
  // The api scope is a usecase specific one
  scope: 'openid profile resource.read resource.write resource.delete',

  showDebugInformation: true,
};
this.oauthService.configure(authCodeFlowConfig);
this.oauthService.oidc = true; // ID_Token

this.oauthService.setStorage(sessionStorage);

this.oauthService.loadDiscoveryDocumentAndTryLogin()
```

3.2. Exemples de configurations "Resource Server"

Pour backend "springBoot/springMvc"

application.yml

```
# localhost:8585/serverRest
server:
  servlet:
    context-path: /serverRest
    port: 8585

spring:
  datasource:
    driverClassName: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/deviseDB?
createDatabaseIfNotExist=true&serverTimezone=UTC
    username: root
    password: root
  jpa:
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    hibernate.ddl-auto: create
  data:
    jpa:
      repositories:
        enabled: true
  security:
    oauth2:
      resourceserver:
        jwt:
          issuer-uri: http://localhost:8989/auth/realms/myrealm
```

dans pom.xml

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
</dependency>
```

dansSecurityConfig.java

```
@Configuration
@Profile("asOAuth2ResourceServer")
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class AsOAuth2ResourceServerWebSecurityConfig extends
WebSecurityConfigurerAdapter {

    @Override
    protected void configure(final HttpSecurity http) throws Exception {
        http.authorizeRequests()
```

```

        .antMatchers("/", "/favicon.ico", "**/*.png", "**/*.gif", "**/*.svg",
        "**/*.jpg", "**/*.html", "**/*.css", "**/*.js").permitAll()
        .antMatchers("/devise-api/public/**").permitAll()
        .antMatchers("/read/**").hasAuthority("SCOPE_resource.read")
        .antMatchers("/write/**").hasAuthority("SCOPE_resource.write")
        .anyRequest().authenticated()
        .and().cors() //enable CORS (avec @CrossOrigin sur class @RestController)
        .and().csrf().disable()
        .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
        .and()
        .oauth2ResourceServer()
        .jwt();
    }
}

```

dans **DeviseRestCtrl.java**

```

....
@RestController
@CrossOrigin(origins = "**")
/*@CrossOrigin(origins = { "http://localhost:3000" , "http://localhost:4200" } , methods =
{ RequestMethod.GET , RequestMethod.POST , RequestMethod.PUT ,
RequestMethod.DELETE })*
@RequestMapping(value="/devise-api" , headers="Accept=application/json")
public class DeviseRestCtrl {

...
//localhost:8585/serverRest/devise-api/private/role_admin/devise/m1 (DELETE)
@PreAuthorize("hasAuthority('SCOPE_resource.delete')")
@DeleteMapping("/private/role_admin/devise/{codeDevise}")
public ResponseEntity<?> deleteDeviseByCode(@PathVariable("codeDevise") String codeDevise)
{
    serviceDevise.deleteDevise(codeDevise);
    Map<String,Object> mapRes = new HashMap<>();
    mapRes.put("message", "devise bien supprimée pour code="+codeDevise);
    //mapRes.put(autreClef, autreValeur);
    return new ResponseEntity<Object>(mapRes,HttpStatus.OK);
}
}

```