

1. Keycloak (serveur OAuth2 et OIDC)

keycloak est un serveur d'autorisation **oauth2** et **oidc** basé sur une machine virtuelle java.
Les anciennes versions de keycloak étaient basées sur jboss wildfly et java >=8.
Les versions récentes de keycloak sont basées sur **Quarkus** et sont plus optimisées.

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/21.0.1/keycloak-21.0.1.zip>
- Extraire le contenu de l'archive dans un répertoire (ex: c:\prog\keycloak-21.0.1)
- Modifier si besoin le numéro de port dans `keycloak-21.0.1\conf/keycloak.conf`
ajouter **http-port=8989** dans conf/keycloak.conf (par défaut 8080)

1.2. Démarrage du serveur keycloak :

lancer_keycloak.bat

```
bin/kc.bat start-dev
```

```
pause
```

URL: <http://localhost:8080/auth/> ou bien <http://localhost:8989/auth/> (anciennes versions)

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

1.3. Première connexion

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



Administration Console

Please create an initial admin user
to get started.

Username

admin

Password

•••••

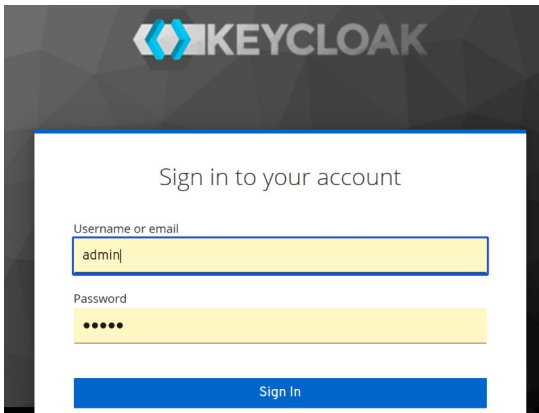
Password confirmation

•••••

Create

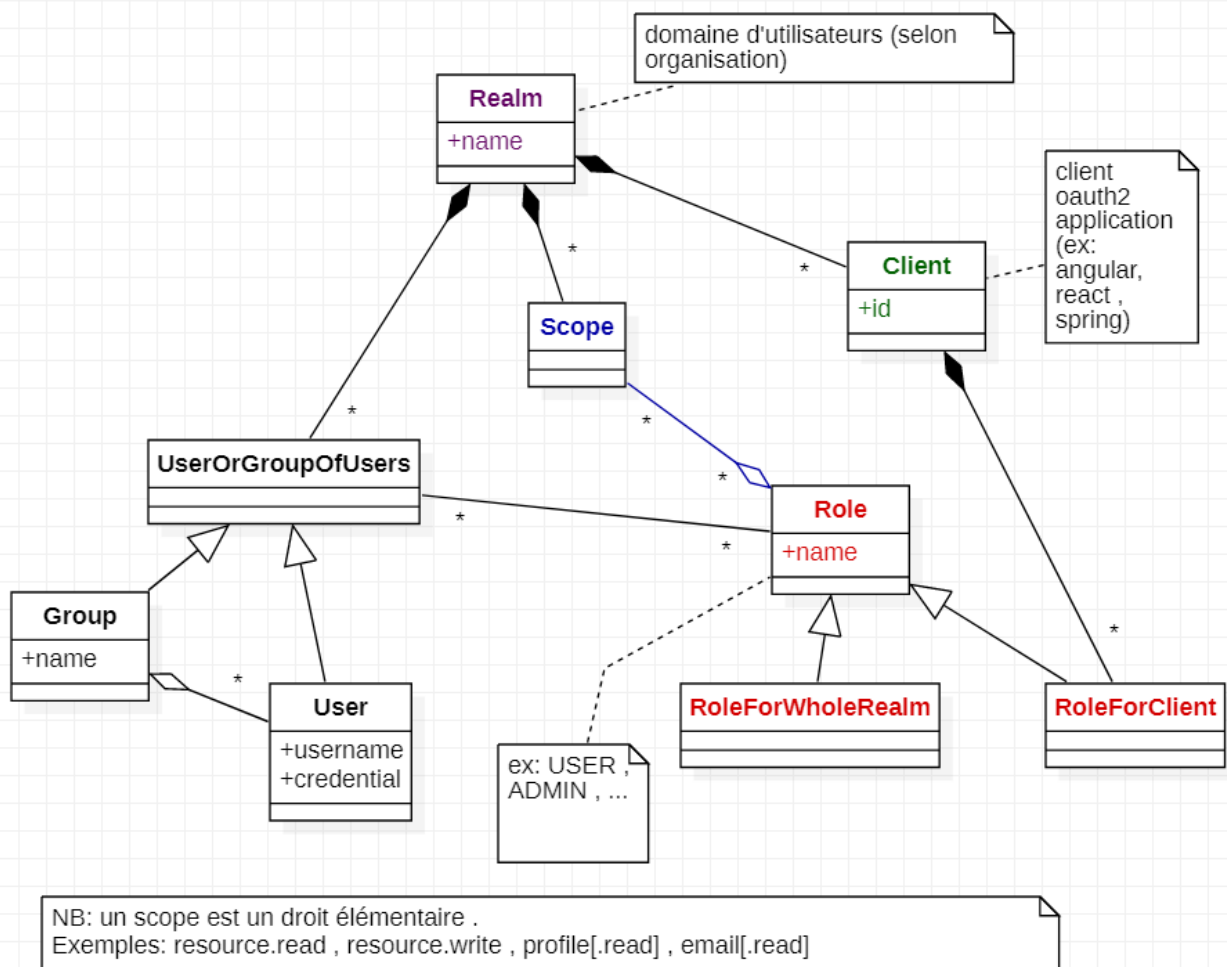
2. Administration de keycloak

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'etre 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 type="text" value="myrealm"/> |
| Enabled | <input checked="" type="checkbox"/> ON |
| | <input type="button" value="Create"/> <input type="button" value="Cancel"/> |

Dans ce realm :

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

Add user


| | |
|----------------|---|
| ID | <input type="text"/> |
| Created At | <input type="text"/> |
| Username * | <input type="text" value="user1"/> |
| Email | <input type="text" value="jean.bon@mycompany.com"/> |
| First Name | <input type="text" value="jean"/> |
| Last Name | <input type="text" value="Bon"/> |
| User Enabled ? | <input checked="" type="checkbox"/> ON |

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/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 :

utilisateurs (**admin1/pwd1**) et (**admin2/pwd2**) membres du groupe "**admin_of_myrealm**"
utilisateurs (**mgr1/pwd1**) et (**mgr2/pwd2**) membres du groupe "**manager_of_myrealm**"

2.3. Initialisation d'un nouveau client (application)

http://localhost:8080_ou_8989/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

Authentication flow : Standard flow

Set Valid redirect= <https://www.keycloak.org/app/>*

Web origins= <https://www.keycloak.org>

Valid redirect URIs ?

<https://www.keycloak.org/app/>*

[+ Add valid redirect URIs](#)

Valid post logout
redirect URIs ?

[+ Add valid post logout redirect URIs](#)

Web origins ?

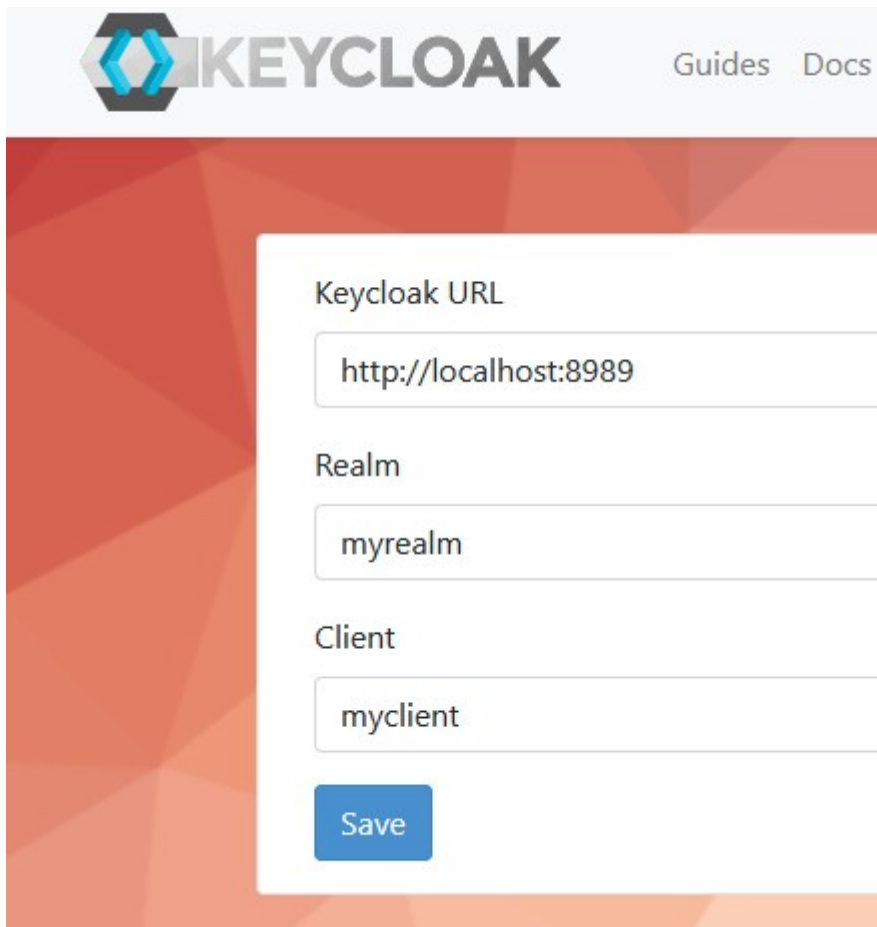
<https://www.keycloak.org>

[+ Add web origins](#)

Save

test :

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

The image shows the Keycloak login interface. At the top, there is a header with the Keycloak logo (a blue and grey hexagon) and the word "KEYCLOAK" in bold. To the right of the logo are links for "Guides" and "Docs". Below the header is a large red and orange geometric pattern. In the center, there is a white login form. The form has three input fields: "Keycloak URL" with the value "http://localhost:8989", "Realm" with the value "myrealm", and "Client" with the value "myclient". Below these fields is a blue "Save" button.

KEYCLOAK Guides Docs

Keycloak URL

Realm

Client

save

sign in (ex : *user1/pwd*)

Hello, jean Bon

sign out

2.5. Configuration d'une nouvelle application cliente

S'etre préalablement connecté en tant qu'administrateur (ex: <http://localhost:8989/admin>, 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

callback uri= <http://localhost:8081>

webOrigin= <http://localhost:8081>

Exemple 2 (appli angular en mode dev):

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

valid redirect uris= <http://localhost:4200/silent-refresh.html> <http://localhost:4200/ngs-loggedIn>

valid post logout Uris : <http://localhost:4200/ngs-logInOut>

webOrigins= <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'etre préalablement connecté en tant qu'administrateur (ex: <http://localhost:8989> , console d'administration, admin/admin)

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

Choisir généralement *AssignedType=Optional*

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

Create *resource.write* (description= save or update resource) **Save**

Create *resource.delete* (description= delete resource) **Save**

2.7. Associer des scopes à une appli cliente

Sélectionner Clients/...client...

onglet "client scopes" et "Add client scope"

webappclient2

OpenID Connect

Clients are applications and services that can request authentication of a user.

Settings

Roles

Client scopes

Sessions

Advanced

Setup

Evaluate

▼

Name

🔍

Search by name

→

Add client scope

Change type to

▼

⋮

| <input type="checkbox"/> | Assigned client scope | Assigned type | Description |
|--------------------------|-------------------------|---------------|--|
| <input type="checkbox"/> | webappclient2-dedicated | none ▼ | Dedicated scope and mappers for this client |
| <input type="checkbox"/> | acr | Default ▼ | OpenID Connect scope for add acr (authentication c |
| <input type="checkbox"/> | address | Optional ▼ | OpenID Connect built-in scope: address |
| <input type="checkbox"/> | email | Default ▼ | OpenID Connect built-in scope: email |
| <input type="checkbox"/> | microprofile-jwt | Optional ▼ | Microprofile - JWT built-in scope |
| <input type="checkbox"/> | offline_access | Optional ▼ | OpenID Connect built-in scope: offline_access |
| <input type="checkbox"/> | phone | Optional ▼ | OpenID Connect built-in scope: phone |
| <input type="checkbox"/> | profile | Default ▼ | OpenID Connect built-in scope: profile |
| <input type="checkbox"/> | resource.delete | Default ▼ | delete resource |
| <input type="checkbox"/> | resource.read | Default ▼ | read confidential resource |
| <input type="checkbox"/> | resource.write | Default ▼ | save or update resource |

Choisir "Default" ou "Optional" (au niveau des futurs consentements proposés).

Attention : affichage par page de 20 ok (par page de 10 avec quelques bugs)

NB : si scopes pas associés à une appli cliente , potentielle erreur : invalid scopes

2.8. Configuration de Roles spécifique à une application cliente

Il est possible de configurer des roles spécifiques à une application cliente ...

Mais c'est un peu compliqué...

2.9. Configuration de Roles pour tout le realm

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

Sélectionner un realm (ex ; myRealm)

Point d'entrée **Realm roles**

pour configurer des rôles de portée globale au realm (ex : **USER** , **MANAGE_RW** , **ADMIN_CRUD** ou ...)

2.10. Associations entre Roles et Scopes

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

Sélectionner un realm (ex ; myRealm)

Point d'entrée **Client scopes**

sélectionner un des scopes préalablement créé (ex : resource.delete)

Onglet "scope" puis **assign rôle**

[Client scopes](#) > Client scope details

resource.delete openid-connect

Settings

Mappers

Scope

 If there is no role scope mapping defined, each user is permitted to use this client scope. If there are

 Search by name



☒ Hide inherited roles

Assign role

Unassign



Name

Inherited



ADMIN_CRUD

False

Exemples :

ADMIN_CRUD à rattacher aux scopes **resource.read resource.write resource.delete**

USER à rattacher aux scopes **resource.read**

MANAGE_RW à rattacher aux scopes **resource.read resource.write**

2.11. Eventuelle configuration de groupes d'utilisateurs

Groups > Group details

admin_of_myrealm

Child groups

Members

Attributes

Role mapping

Add member

☐ Include sub-group users

| <input type="checkbox"/> | Name | Email | First name | Last name |
|--------------------------|--------|-------|------------|-----------|
| <input type="checkbox"/> | admin1 | – | alex | Therieur |
| <input type="checkbox"/> | admin2 | – | axelle | Aire |

2.12. Affectation de rôles aux utilisateurs ou aux groupes

En retournant sur partie "Manage / Users ou Groups" de la console d'admin keycloak, **sélectionner un groupe ou un utilisateur**
onglet "Role mapping" / Assign rôle

Groups > Group details

admin_of_myrealm

Child groups

Members

Attributes

Role mapping

Q Search by name



☒ Hide inherited roles

Assign role

Unassign

| <input type="checkbox"/> | Name | Inherited | Description |
|--------------------------|------------|-----------|--|
| <input type="checkbox"/> | MANAGE_RW | False | with resource.read and resource.write scopes |
| <input type="checkbox"/> | USER | False | Basic User |
| <input type="checkbox"/> | ADMIN_CRUD | False | administrator CRUD |

Users > User details

mgr1

Details

Attributes

Credentials

Role mapping

Groups

Consents

Identifi

Q Search by name



☒ Hide inherited roles

Assign role

Unassign

| <input type="checkbox"/> | Name | Inherited |
|--------------------------|-----------------------|-----------|
| <input type="checkbox"/> | MANAGE_RW | False |
| <input type="checkbox"/> | default-roles-myrealm | False |
| <input type="checkbox"/> | USER | False |

2.13. 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/realms/myrealm/.well-known/openid-configuration

2.14. import/export partiel d'un realm au format json

Sélectionner "myRealm"

puis menu **configure/Realm Settings**

Action "Partial import" et "*Partial export*"

→ ça génère un très gros fichier *realm-export.json* comportant la configuration des scopes/rôles et applications clientes .

Attention : la configuration des utilisateurs (username, password, ...) n'est pas exportée !!!!



2.15. import/export complet d'un realm , serveur arrêté

```
export keycloak myrealm.bat
```

```
bin/kc.bat export --dir backup/myrealm --realm myrealm
pause
```

fichiers générés :

keycloak-21.0.1 > backup > myrealm

| Nom | Modifié le | Type | Taille |
|--|------------------|--------------|--------|
|  myrealm-realm.json | 09/03/2023 12:33 | Fichier JSON | 74 Ko |
|  myrealm-users-0.json | 09/03/2023 12:33 | Fichier JSON | 5 Ko |

```
import keycloak myrealm.bat
```

```
bin/kc.bat import --dir backup/myrealm --override true
pause
```

Options à découvrir : <https://www.keycloak.org/server/importExport>

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/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/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);
}
}

```