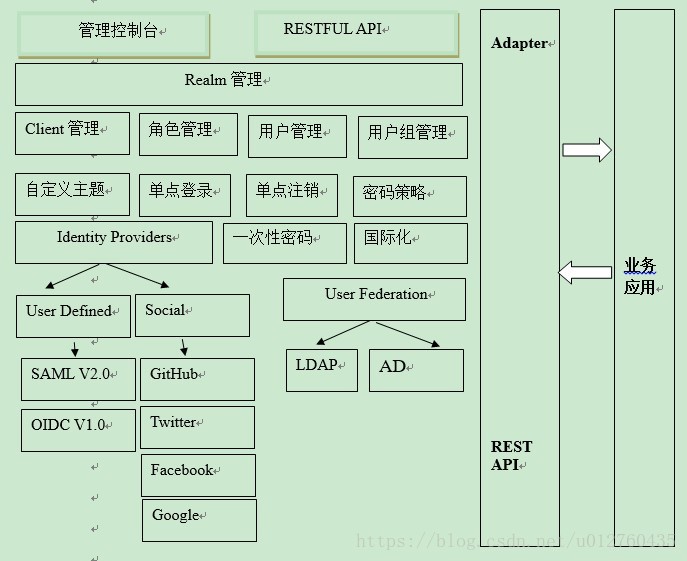
Keycloak使用记录



1. Realm

默认生成master realm，是一个特别的与其它realm不同的realm。在master realm中的用户有管理realms的权限。当创建一个realm时，自动创建很多角色，授予细粒度的权限用于访问该realm。可以创建超级用户，也可以创建只能访问某个具体的realm。

在master realm有两上realm role，admin和create-realm。

每个realm在master realm中有一个client作为代表，它们的名称用<realm name>-realm表示。这些client拥有client 级别角色，这些角色是关于管理单个realm的各种角色。如：view-realm、view-users、view-clients等。

每个Realm有一个专用的管理console,访问url为/auth/admin/{realm-name}/console。每个Realm有一个内置的client realm-management，定义了很多client 级别的角色，用于批准realm

可以创建多个Realm，每个Realm之间是隔离，单点登录无效。

1. Client管理

**Consent Required：与oauth2类似，一个确认授权明细页面。**

**Direct Access Grants Enabled：**

支持手动录入

confidential  请求参数包括secret,适合后端认证。生成access\_token、refresh\_token

public 如confidential类似，请求参数不同，是通过帐户密码进行认证。

Service Account 当Access Type选择confidential 时显示。Oauth2的[Client Credentials Grant](https://www.keycloak.org/docs/latest/server_admin/index.html" \l "_client_credentials_grant)，可以在Service Account Roles Tab进行角色和client角色的配置。

Policy Decision Strategies

**Unanimous，默认的策略，所有的policy必须评估为正面的决定，最终才能通过。**

**Affirmative，至少有一个决定是正面的。**

**Consensus，正面的决定数量大于负面的决定。**

Client Scopes

客户端范围管理

Audience 在mappers中的mapper type中选择audience选项。

端点

**/realms/{realm-name}/protocol/openid-connect/auth**

curl -X POST -H "Content-Type: application/x-www-form-urlencoded" -d 'grant\_type=password&client\_id=ROOMIS&username=123&password=123' "http://keycloak域名IP/auth/realms/{realms名字}/protocol/openid-connect/token" | jq

This is the URL endpoint for obtaining a temporary code in the Authorization Code Flow or for obtaining tokens via the Implicit Flow, Direct Grants, or Client Grants.

**/realms/{realm-name}/protocol/openid-connect/token**

This is the URL endpoint for the Authorization Code Flow to turn a temporary code into a token.

**/realms/{realm-name}/protocol/openid-connect/logout**

This is the URL endpoint for performing logouts.

**/realms/{realm-name}/protocol/openid-connect/userinfo**

This is the URL endpoint for the User Info service described in the OIDC specification.

**/realms/{realm-name}/protocol/openid-connect/revoke**

This is the URL endpoint for OAuth 2.0 Token Revocation described in [RFC7009](https://tools.ietf.org/html/rfc7009).

In all of these replace *{realm-name}* with the name of the realm.

角色（Role）

角色分为Realm和client，均可以绑定复合角色，即可以有下级角色。

在用户详情页，进行角色与用户的绑定。

角色可以设置默认角色，新创建的用户，默认绑定这些角色。

认证流(authentication flows)

服务端开发(Server Developer)

用户存储SPI

使用User Storage SPI 写keycoloak连接外部用户数据库和凭证存储的扩展。

编写User Storage SPI的实现来联系外部用户存储和内部的用户对象模型，keycloak用于登录的用户并且管理他们。

当Keycloak运行时，需要查找一个用户，比如当用户登录时，它将执行多个步骤定位该用户，首先从用户内存中查看，如果在内存中找到就使用它。否则，在Keycloak本地数据库查找，如果没有找到，则循环从用户存储SPI实现中查找，直到其中一个返回该用户。Provider查询外部用户存储，并且映射外部数据到Keycloak的用户元模型。

用户存储SPI提供者实现也可以执行复杂的查询、CRUD操作。验证管理凭证，或者一次对用户执行批量的的更新操作。这取决于外部存储的性能。

用户存储SPI的实现打包，部署和其它Java EE组件类似。没有被默认使用，而是必须对每个Realm进行开启和配置，在用户联邦Tab管理界面。

<dependency>

<groupId>org.keycloak</groupId>

<artifactId>keycloak-server-spi</artifactId>

<version>11.0.0</version>

<scope>provided</scope>

</dependency>