Spring Authorization Server

É um projeto spring que fornece suporte para Oauth 2.0 e 2.1, OpenID JWT e suporta autoconfiguração via spring boot, substituindo o projeto Spring Oauth Server que foi descontinuado em maio de 2022.

Componentes

RegisteredClient

É uma classe que **representa um client registrado no authorization server** e é usada para gerar um token/authorization code. O client deve ser registrado no authorization server através do RegisteredClientRepository para usar algum fluxo de oauth.

Um RegisteredClient tem um client id, opcionalmente um client secret (dependendo se é public ou private) e metadados sobre o client como nome, redirect-uri, tipo de grant types, scopes que podem ser solicitados pelo client, etc

```java
public class RegisteredClient implements Serializable {
        private String id; ❶
        private String clientId; ❷
        private Instant clientIdIssuedAt; ❸
        private String clientSecret; ❹
        private Instant clientSecretExpiresAt; ❺
        private String clientName; ❻
        private Set<ClientAuthenticationMethod> clientAuthenticationMethods; ❼
        private Set<AuthorizationGrantType> authorizationGrantTypes; ❽
        private Set<String> redirectUris; ❾
        private Set<String> postLogoutRedirectUris; ❿
        private Set<String> scopes; ⓫
        private ClientSettings clientSettings; ⓬
        private TokenSettings tokenSettings; ⓭

        ...

}
```

RegisteredClientRepository

É um componente obrigatório, um **repository usado para criar, buscar e atualizar RegisteredClient**, as implementações são JdbcRegisteredClientRepository e InMemoryRegisteredClientRepository. É possível configurar a classe apenas definindo um bean ou pelo OAuth2AuthorizationServerConfigurer.

OAuth2Authorization

É uma classe que **representa uma autorização oauth2 concedida a um client por um resource owner** ou pelo próprio client quando usando client_credentials. Ela possui o(s) token(s) (OAuth2AccessToken, OAuth2RefreshToken e/ou OAuth2AuthorizationCode) e metadados sobre o client/resource owner.

Uma instância de OAuth2Authorization é considerada inativa se o token/refresh token expiraram, seja por ter acabado o lifetime ou por revoke. Toda implementação de `OAuth2Token` possui um método getClaims().

```
public class OAuth2Authorization implements Serializable {
        private String id; ❶
        private String registeredClientId; ❷
        private String principalName; ❸
        private AuthorizationGrantType authorizationGrantType; ❹
        private Set<String> authorizedScopes; ❺
        private Map<Class<? extends OAuth2Token>, Token<?>> tokens; ❻
        private Map<String, Object> attributes; ❼

        ...

}
```

OAuth2AuthorizationService

É o **equivalente ao UserDetailsService de spring security**, é o componente central, responsável por **salvar, remover e buscar OAuth2Authorization.** A implementação padrão InMemoryOAuth2AuthorizationService e também existe a implementação JdbcOAuth2AuthorizationService

OAuth2AuthorizationConsent

Representa o **consentimento fornecido a um client (scopes aprovados pelo RO).** Essa classe possui 3 atributos, o clientId do client, o principal name (nome do RO) e um set de GrantedAuthority (uma classe que representa um scope, claim, uma permissão, etc concedida por um RO).

OAuth2AuthorizationConsentService

Salva, remove e busca OAuth2AuthorizationConsent. Suas implementações são InMemoryOAuth2AuthorizationConsentService (implementação padrão) e JdbcOAuth2AuthorizationConsentService.

OAuth2TokenContext

Interface que **define métodos para obter informações de contexto de um token** OAuth2Token (que pode não ter sido gerado ainda) e é usada pelas classes OAuth2TokenGenerator e OAuth2TokenCustomizer. Seus métodos são:

```java
public interface OAuth2TokenContext extends Context {

        default RegisteredClient getRegisteredClient() ...  1

        default <T extends Authentication> T getPrincipal() ... 2

        default AuthorizationServerContext getAuthorizationServerContext() ...

        @Nullable
        default OAuth2Authorization getAuthorization() ...  4

        default Set<String> getAuthorizedScopes() ...  5

        default OAuth2TokenType getTokenType() ...  6

        default AuthorizationGrantType getAuthorizationGrantType() ...  7

        default <T extends Authentication> T getAuthorizationGrant() ...  8

        ...
}
```

OAuth2TokenGenerator

É responsável por **gerar um OAuth2Token por meio das informações contidas em OAuth2TokenContext**, tipo do token gerado depende do atributo OAuth2TokenType de OAuth2TokenContext que é obtido no método TokenSettings.getAccessTokenFormat() do RegisteredClient.

Suas implementações são:

- OAuth2RefreshTokenGenerator para refresh tokens
- JwtGenerator para tokens JWT
- OAuth2AccessTokenGenerator para token opacos

DelegatingOAuth2TokenGenerator que delega para OAuth2RefreshTokenGenerator e OAuth2AccessTokenGenerator é a implementações padrão disponibilizada, porém, se um bean do tipo JwtEncoder ou JWKSource<SecurityContext> estiver disponível JwtGenerator será incluído.

OAuth2TokenCustomizer

É uma **classe usada para customizar os atributos (acessíveis no OAuth2TokenContext) de um OAuth2Token** antes dele ser gerado pelo OAuth2TokenGenerator.

UserDetailsService

Quando se usa authorization code é **necessário definir um UserDetailsService** ou um Provider para autenticar o resource owner já que isso não é função de oauth2.