JWT认证

AuthenticationFilter.java processJwtAuth处理JWT认证

ClientAssertion.java ClientAssertion类用于验证JWT断言的有效性 load实现了完整逻辑

Jwt.java JWT类继承自JsonWebResponse，实现了JWT的基本结构和解析功能

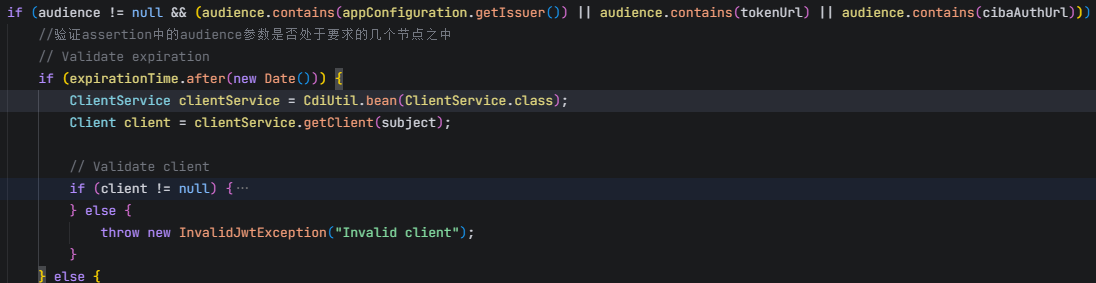
ClientAuthnRequest.java ClientAuthnRequest类用于客户端生成断言

首先通过getCliams.getClaimsStringList获取audience列表，可知返回的形式是List<String>类型，

从getClaimsStringList源码来看，该函数是根据key是什么声明值，给出该声明值列表，他对于audience是一个和audience是多个的情况都有涉及

jwt.getClaims() 是访问 JWT 声明的入口点，它返回一个 JwtClaims 对象，该对象提供了丰富的方法来处理和访问 JWT 中的各种声明。

接着就是验证audience是否处于声明值列表之中



ClientAssertion.java load函数

List<String> audience = jwt.getClaims().getClaimAsStringList(JwtClaimName.AUDIENCE);

jwt获取JwtClaims类型的对象，然后接着获取jwt的audience的列表

JwtClaims是一个LinkedHashMap<String, Object>，用于存储JWT的所有声明

getClaimAsStringList函数用于获取

if (audience != null && (audience.contains(appConfiguration.getIssuer()) || audience.contains(tokenUrl) || audience.contains(cibaAuthUrl))) {

验证JWT中的audience声明中提取的audience列表是否包含以下几个有效目标

服务器的颁发者标识符 令牌端点URL CIBA认证端点URL

JwtClaimSet.java getClaimAsStringList

public List<String> getClaimAsStringList(String key) {

        List<String> list = new ArrayList<>();

        Object value = getClaim(key);

        try {

            if (value instanceof JSONArray) {

                JSONArray jsonArray = (JSONArray) value;

                for (int i = 0; i < jsonArray.length(); i++) {

                    list.add(jsonArray.getString(i));

                }

            } else {

                String claim = getClaimAsString(key);

                if (claim != null) {

                    list.add(claim);

                }

            }

        } catch (JSONException e) {

            // ignore

        }

        return list;

    }

功能：根据key是什么声明值，给出声明值的列表

例如getClaim

验证了assertion是否过期

if (expirationTime.after(new Date())) {