🍉Spring Authorization Server (9) 授权服务的授权信息存储方式扩展

爱吃西瓜的胖娃 2023-10-30 877 阅读4分钟

> 架构版本
>
> Spring Boot 3.1
>
> Spring Authorization Server 1.1.1
>
> spring-cloud 2022.0.3
>
> spring-cloud-alibaba 2022.0.0.0
>
> 完整代码👉 **watermelon-cloud**

什么是授权信息?

> 在Spring Authorization Server 中授权信息指的是客户端应用程序请求访问受保护资源时所需要的权限信息。这些信息通常包括客户端ID、客户端密钥、授权类型和范围等。oauth2_authorization 表里面就存储的授权信息，包含有access_token、refesh_token、过期时间等关键数据字段，有兴趣的可以再去详细看看这个表的其他字段数据。

授权信息存储方式为什么要去扩展?

因为我们前面扩展的 `PhoneCaptchaAuthenticationToken` is not in the allowlist 序列化的时候出现了异常😂，玩什么扩展嘛，花里胡哨的，看看原因再去想怎么解决这个问题。

>

```
.authorization.defaultauth.support.phone.PhoneCaptchaAuthenticationToken is not in the al.
using Jackson annotations or by providing a Mixin. If the serialization is only done by a
.com/spring-projects/spring-security/issues/4370 for details
```

为什么 `UsernamePasswordAuthenticationToken` 内置的就行 `AuthenticationToken` 就可以呢，自定义的 `PhoneCaptchaAuthenticationToken` 就出现 is not in the allowlist 很疑惑啊。

> 再看看关键的错误信息
> JdbcOAuth2AuthorizationService$OAuth2AuthorizationRowMapper.parseMap(JdbcOAuth2AuthorizationService.java:517)
> `JdbcOAuth2AuthorizationService` 517行看看有啥

java 复制代码

```java
private Map<String, Object> parseMap(String data) {
  try {
          return this.objectMapper.readValue(data, new >TypeReference<Map<String, Object>
  } catch (Exception ex) {
         throw new IllegalArgumentException(ex.getMessage(), ex);
  }
}
```

转换出错了，那我们对比看看内置的 `UsernamePasswordAuthenticationToken` 与 `PhoneCaptchaAuthenticationToken` 存储的数据到底有什么差异

> PhoneCaptchaAuthenticationToken 时的data数据

java 复制代码

```java
{"@class":"java.util.Collections$UnmodifiableMap","java.security.Principal":
{"@class":"com.watermelon.authorization.defaultauth.support.phone.PhoneCaptchaAuthenticat
"authorities":["java.util.Collections$UnmodifiableRandomAccessList",
[{"@class":"org.springframework.security.core.authority.SimpleGrantedAuthority","authorit
{"@class":"org.springframework.security.core.authority.SimpleGrantedAuthority","authority
```

```
{bcrypt}$2a$10$2sGumFFLA./mT.d7w6awleE9Y9KsPL.CjwzvyvlHB5fblCBYCX/di",
"avatar":null,"status":1,"authorities":["java.util.ArrayList",
[{"@class":"org.springframework.security.core.authority.SimpleGrantedAuthority","authorit
{"@class":"org.springframework.security.core.authority.SimpleGrantedAuthority","authority
{"@class":"org.springframework.security.core.authority.SimpleGrantedAuthority","authority
"accountNonLocked":true},"credentials":null,"name":"18682678995"},"org.springframework.se
{"@class":"org.springframework.security.oauth2.core.endpoint.OAuth2AuthorizationRequest",
{"value":"authorization_code"},"responseType":{"value":"code"},"clientId":"messaging-clie
["java.util.Collections$UnmodifiableSet",["openid","profile"]],"state":"-Vrwm1Muxpcgwj7ST
{"@class":"java.util.Collections$UnmodifiableMap","nonce":"vHmRCCodKwOltSXWCtlS_XZPQyLTTq
response_type=code&client_id=messaging-client&scope=openid%20profile&state=-Vrwm1Muxpcgwj
```

UsernamePasswordAuthenticationToken的data数据

java 复制代码

```
{"@class":"java.util.Collections$UnmodifiableMap","java.security.Principal":
{"@class":"org.springframework.security.authentication.UsernamePasswordAuthenticationToke
[{"@class":"org.springframework.security.core.authority.SimpleGrantedAuthority","authorit
{"@class":"org.springframework.security.core.authority.SimpleGrantedAuthority","authority
{"@class":"org.springframework.security.core.authority.SimpleGrantedAuthority","authority
{"@class":"org.springframework.security.web.authentication.WebAuthenticationDetails","rem
{"@class":"com.watermelon.authorization.defaultauth.builtin.dto.SysUserDto","id":1,"phone
{bcrypt}$2a$10$pJYa8tfSmDysF7pz5EVJ3.qg7Q8G3qNS00KSCurw5VpUfIVoksR4K","avatar":null,"stat
[{"@class":"org.springframework.security.core.authority.SimpleGrantedAuthority","authorit
{"@class":"org.springframework.security.core.authority.SimpleGrantedAuthority","authority
{"@class":"org.springframework.security.core.authority.SimpleGrantedAuthority","authority
"accountNonLocked":true},"credentials":null},"org.springframework.security.oauth2.core.en
{"@class":"org.springframework.security.oauth2.core.endpoint.OAuth2AuthorizationRequest",
{"value":"authorization_code"},"responseType":{"value":"code"},"clientId":"messaging-clie
["openid","profile"]],"state":"6o4EXBxk_kN9U8jof0rGaz5t4UjB64h5Xc076gGEORg=","additionalP
{"@class":"java.util.Collections$UnmodifiableMap","nonce":"REhLXzkG6XBFP8vmQGuMkWcYiQ0vvk
"authorizationRequestUri":"http://192.168.56.1:9000/oauth2/authorize?response_type=code&c
```

以上除了class不一样,似乎结构大致都一样，怎么就不支持转换了呢？难道因为 `PhoneCaptchaAuthenticationToken` 不是亲生的，JdbcOAuth2AuthorizationService#parseMap()就不支持了😄？。

> 1:用 `UsernamePasswordAuthenticationToken` 替换 `PhoneCaptchaAuthenticationToken`
> 2:重新一个 `JdbcOAuth2AuthorizationService` 来进行存储和转换

选择第2种方案，因为后期可能还扩展其他的 `AuthenticationToken` ，再加上授权信息想使用redis进行存储。
那就开始干吧。
JdbcOAuth2AuthorizationService 实现了 OAuth2AuthorizationService 接口，同样实现它干就完事了

> RedisOAuth2AuthorizationServiceImpl

java 复制代码

```java
@Component
public class RedisOAuth2AuthorizationServiceImpl implements OAuth2AuthorizationService {

    private final static String AUTHORIZATION_TYPE = "authorization_type";

    private final static String OAUTH2_PARAMETER_NAME_ID = "id";

    private final static Long TIMEOUT = 600L;

    @Resource
    private RedisTemplate<String, Object> redisTemplate;

    @Override
    public void save(OAuth2Authorization authorization) {
        Assert.notNull(authorization, "authorization cannot be null");
        redisTemplate.setValueSerializer(RedisSerializer.java());
        redisTemplate.opsForValue().set(buildAuthorizationKey(OAUTH2_PARAMETER_NAME_ID, aut
        if (isState(authorization)) {
            String state = authorization.getAttribute(OAuth2ParameterNames.STATE);
            String isStateKey = buildAuthorizationKey(OAuth2ParameterNames.STATE, state);
            redisTemplate.setValueSerializer(RedisSerializer.java());
            redisTemplate.opsForValue().set(isStateKey, authorization, TIMEOUT, TimeUnit.SE
        }
        if (isAuthorizationCode(authorization)) {
            OAuth2Authorization.Token<OAuth2AuthorizationCode> authorizationCode =
```

```
            Instant expiresAt = authorizationCode.getToken().getExpiresAt();//过期时间
            Instant issuedAt = authorizationCode.getToken().getIssuedAt();//发放token的时间
            Date expiresAtDate = Date.from(expiresAt);
            Date issuedAtDate = Date.from(issuedAt);
            redisTemplate.setValueSerializer(RedisSerializer.java());
            redisTemplate.opsForValue().set(isAuthorizationCodeKey, authorization, TIMEOUT,
        }
        if (isAccessToken(authorization)) {
            OAuth2Authorization.Token<OAuth2AccessToken> accessToken =
                    authorization.getToken(OAuth2AccessToken.class);
            String tokenValue = accessToken.getToken().getTokenValue();
            String isAccessTokenKey = buildAuthorizationKey(OAuth2ParameterNames.ACCESS_TOI
            Instant expiresAt = accessToken.getToken().getExpiresAt();//过期时间
            Instant issuedAt = accessToken.getToken().getIssuedAt();//发放token的时间
            Date expiresAtDate = Date.from(expiresAt);
            Date issuedAtDate = Date.from(issuedAt);
            redisTemplate.setValueSerializer(RedisSerializer.java());
            redisTemplate.opsForValue().set(isAccessTokenKey, authorization, TIMEOUT, Timel
        }
        if (isRefreshToken(authorization)) {
            OAuth2Authorization.Token<OAuth2RefreshToken> refreshToken =
                    authorization.getToken(OAuth2RefreshToken.class);
            String tokenValue = refreshToken.getToken().getTokenValue();
            String isRefreshTokenKey = buildAuthorizationKey(OAuth2ParameterNames.REFRESH_T
            Instant expiresAt = refreshToken.getToken().getExpiresAt();//过期时间
            Instant issuedAt = refreshToken.getToken().getIssuedAt();//发放token的时间
            Date expiresAtDate = Date.from(expiresAt);
            Date issuedAtDate = Date.from(issuedAt);
            redisTemplate.setValueSerializer(RedisSerializer.java());
            redisTemplate.opsForValue().set(isRefreshTokenKey, authorization, TIMEOUT, Time

        }
        if (isIdToken(authorization)) {
            OAuth2Authorization.Token<OidcIdToken> idToken =
                    authorization.getToken(OidcIdToken.class);
            String tokenValue = idToken.getToken().getTokenValue();
            String isIdTokenKey = buildAuthorizationKey(OidcParameterNames.ID_TOKEN, tokenV
            Instant expiresAt = idToken.getToken().getExpiresAt();//过期时间
            Instant issuedAt = idToken.getToken().getIssuedAt();//发放token的时间
            Date expiresAtDate = Date.from(expiresAt);
            Date issuedAtDate = Date.from(issuedAt);
            redisTemplate.setValueSerializer(RedisSerializer.java());
            redisTemplate.opsForValue().set(isIdTokenKey, authorization, TIMEOUT, TimeUnit
        }
        if (isDeviceCode(authorization)) {
            OAuth2Authorization.Token<OAuth2DeviceCode> deviceCode =
```

```
            String isDeviceCodeKey = buildAuthorizationKey(OAuth2ParameterNames.DEVICE_CODI
            Instant expiresAt = deviceCode.getToken().getExpiresAt();//过期时间
            Instant issuedAt = deviceCode.getToken().getIssuedAt();//发放token的时间
            Date expiresAtDate = Date.from(expiresAt);
            Date issuedAtDate = Date.from(issuedAt);
            redisTemplate.setValueSerializer(RedisSerializer.java());
            redisTemplate.opsForValue().set(isDeviceCodeKey, authorization, TIMEOUT, TimeUr
        }
        if (isUserCode(authorization)) {
            OAuth2Authorization.Token<OAuth2UserCode> userCode =
                    authorization.getToken(OAuth2UserCode.class);
            String tokenValue = userCode.getToken().getTokenValue();
            String isUserCodeKey = buildAuthorizationKey(OAuth2ParameterNames.USER_CODE, to
            Instant expiresAt = userCode.getToken().getExpiresAt();//过期时间
            Instant issuedAt = userCode.getToken().getIssuedAt();//发放token的时间
            Date expiresAtDate = Date.from(expiresAt);
            Date issuedAtDate = Date.from(issuedAt);
            redisTemplate.setValueSerializer(RedisSerializer.java());
            redisTemplate.opsForValue().set(isUserCodeKey, authorization, TIMEOUT, TimeUnit
        }
    }

    @Override
    public void remove(OAuth2Authorization authorization) {
        List<String> keys = new ArrayList<>();
        String idKey = buildAuthorizationKey(OAUTH2_PARAMETER_NAME_ID, authorization.getId
        keys.add(idKey);
        if (isState(authorization)) {
            String state = authorization.getAttribute(OAuth2ParameterNames.STATE);
            String isStateKey = buildAuthorizationKey(OAuth2ParameterNames.STATE, state);
            keys.add(isStateKey);
        }
        if (isAuthorizationCode(authorization)) {
            OAuth2Authorization.Token<OAuth2AuthorizationCode> authorizationCode =
                    authorization.getToken(OAuth2AuthorizationCode.class);
            String tokenValue = authorizationCode.getToken().getTokenValue();
            String isAuthorizationCodeKey = buildAuthorizationKey(OAuth2ParameterNames.CODI
            keys.add(isAuthorizationCodeKey);
        }
        if (isAccessToken(authorization)) {
            OAuth2Authorization.Token<OAuth2AccessToken> accessToken =
                    authorization.getToken(OAuth2AccessToken.class);
            String tokenValue = accessToken.getToken().getTokenValue();
            String isAccessTokenKey = buildAuthorizationKey(OAuth2ParameterNames.ACCESS_TOI
            keys.add(isAccessTokenKey);
        }
```

```
            String tokenValue = refreshToken.getToken().getTokenValue();
            String isRefreshTokenKey = buildAuthorizationKey(OAuth2ParameterNames.REFRESH_
            keys.add(isRefreshTokenKey);
        }
        if (isIdToken(authorization)) {
            OAuth2Authorization.Token<OidcIdToken> idToken =
                    authorization.getToken(OidcIdToken.class);
            String tokenValue = idToken.getToken().getTokenValue();
            String isIdTokenKey = buildAuthorizationKey(OidcParameterNames.ID_TOKEN, token\
            keys.add(isIdTokenKey);
        }
        if (isDeviceCode(authorization)) {
            OAuth2Authorization.Token<OAuth2DeviceCode> deviceCode =
                    authorization.getToken(OAuth2DeviceCode.class);

            String tokenValue = deviceCode.getToken().getTokenValue();
            String isDeviceCodeKey = buildAuthorizationKey(OAuth2ParameterNames.DEVICE_CODI
            keys.add(isDeviceCodeKey);
        }
        if (isUserCode(authorization)) {
            OAuth2Authorization.Token<OAuth2UserCode> userCode =
                    authorization.getToken(OAuth2UserCode.class);
            String tokenValue = userCode.getToken().getTokenValue();
            String isUserCodeKey = buildAuthorizationKey(OAuth2ParameterNames.USER_CODE, to
            keys.add(isUserCodeKey);
        }
        redisTemplate.delete(keys);
    }

    @Override
    public OAuth2Authorization findById(String id) {
        return (OAuth2Authorization) Optional.ofNullable(redisTemplate.opsForValue().get(bu
    }

    @Override
    public OAuth2Authorization findByToken(String token, OAuth2TokenType tokenType) {
        Assert.hasText(token, "token cannot be empty");
        Assert.notNull(tokenType, "tokenType cannot be empty");
        redisTemplate.setValueSerializer(RedisSerializer.java());
        return (OAuth2Authorization) redisTemplate.opsForValue().get(buildAuthorizationKey(
    }


    private boolean isState(OAuth2Authorization authorization) {
        return Objects.nonNull(authorization.getAttribute(OAuth2ParameterNames.STATE));
    }
```

```
        OAuth2Authorization.Token<OAuth2AuthorizationCode> authorizationCode =
                authorization.getToken(OAuth2AuthorizationCode.class);
        return Objects.nonNull(authorizationCode);
    }

    private boolean isAccessToken(OAuth2Authorization authorization) {
        OAuth2Authorization.Token<OAuth2AccessToken> accessToken =
                authorization.getToken(OAuth2AccessToken.class);
        return Objects.nonNull(accessToken) && Objects.nonNull(accessToken.getToken().getT
    }

    private boolean isRefreshToken(OAuth2Authorization authorization) {
        OAuth2Authorization.Token<OAuth2RefreshToken> refreshToken =
                authorization.getToken(OAuth2RefreshToken.class);
        return Objects.nonNull(refreshToken) && Objects.nonNull(refreshToken.getToken().get
    }

    private boolean isIdToken(OAuth2Authorization authorization) {
        OAuth2Authorization.Token<OidcIdToken> idToken =
                authorization.getToken(OidcIdToken.class);
        return Objects.nonNull(idToken) && Objects.nonNull(idToken.getToken().getTokenValue
    }

    private boolean isDeviceCode(OAuth2Authorization authorization) {
        OAuth2Authorization.Token<OAuth2DeviceCode> deviceCode =
                authorization.getToken(OAuth2DeviceCode.class);
        return Objects.nonNull(deviceCode) && Objects.nonNull(deviceCode.getToken().getTok
    }

    private boolean isUserCode(OAuth2Authorization authorization) {
        OAuth2Authorization.Token<OAuth2UserCode> userCode =
                authorization.getToken(OAuth2UserCode.class);
        return Objects.nonNull(userCode) && Objects.nonNull(userCode.getToken().getTokenVa
    }

    /**
     * redis key 构建
     *
     * @param type  授权类型
     * @param value 授权值
     * @return
     */
    private String buildAuthorizationKey(String type, String value) {
        return AUTHORIZATION_TYPE.concat("::").concat(type).concat("::").concat(value);
```

> `redisTemplate.setValueSerializer(RedisSerializer.java())` 用 `RedisSerializer` 的原因是因为 `OAuth2Authorization` 有些字段类型的原因，用其他的就会抛一些序列化异常的。

选择用 `@Component` 注入，之前 `@Bean` 注入的JdbcOAuth2AuthorizationService 就需要删除掉

> java 复制代码
>
> ```java
> @Bean
> public OAuth2AuthorizationService authorizationService(JdbcTemplate jdbcTemplate,
>                                                        RegisteredClientRepository registe
>     return new JdbcOAuth2AuthorizationService(jdbcTemplate, registeredClientRepository);
> }
> ```
>
> 把它删除掉

这个问题就解决了，从开始到现在 用 spring-authorization-server 的过程很曲折😞。

标签： Spring Boot Spring Cloud 话题： 日新计划更文活动

本文收录于以下专栏

🔥Spring Authorization Server 精讲 专栏目录

本专栏将深入讲解Spring Authorization Server在实践中的扩展点，希...

56 订阅 · 11 篇文章

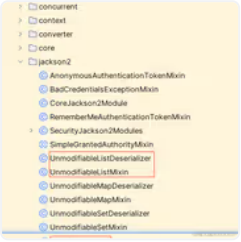
订阅

上一篇 Spring Authorization Ser... 下一篇 Spring Authorization Ser...

评论 3

wfhusb

如图



4月前 点赞 评论 ···

wfhusb

UsernamePasswordAuthenticationToken 之所以可以正常序列化和反序列化，是因为有 UsernamePasswordAuthenticationTokenDeserializer 与 UsernamePasswordAuthenticationTokenMixin，并且在 CoreJackson2Module 中被注册了， 只要对照 UsernamePasswordAuthenticationToken 的 JsonDeserializer 和 Mixin，自定义出PhoneCaptchaAuthenticationToken 对应的这两个类，再写个继承...

4月前 点赞 1 ···

爱吃西瓜的胖娃 作者 ：大佬研究颇深，感谢 👍

4月前 点赞 回复 ···

目录 收起

相关推荐

Spring Authorization Server 动态注册OAuthClient
434阅读 · 1点赞

Spring Authorization Server 快速上手
444阅读 · 0点赞

Spring Authorization Server入门 (十三) 实现联合身份认证，集成Github与Gitee的OAuth登录
3.0k阅读 · 17点赞

Spring Authorization Server 集成MySQL
555阅读 · 2点赞

精选内容

自从我读透了Spring的事务传播性，用户每次都兑奖成功了！！
掉头发的王富贵 · 240阅读 · 4点赞

深入理解微服务中的负载均衡算法与配置策略
努力的小雨 · 96阅读 · 6点赞

Spring Cloud全解析：注册中心之consul注册中心
isfox · 8阅读 · 0点赞

Spring Event 的幕后故事
不爱总结的麦穗 · 100阅读 · 4点赞

从单节点到集群：使用Redis解决负载均衡后WebSocket在线聊天室通信难题
翼飞 · 83阅读 · 1点赞

为你推荐

Spring Authorization Server (4) 客户端、资源服务、授权服务 源码加流程细讲 再也...
爱吃西瓜的胖娃 11月前 1.5k 9 10 Spring ... Spring ...

Spring Authorization Server (3) so seasy 集成第三方【gitee、github】oauth2登录
爱吃西瓜的胖娃 11月前 1.1k 3 3 Spring ... Spring ...

Spring Authorization Server Password授权扩展

hundanli 5月前 386 1 2 Java Spring B...

Spring Authorization Server 全新授权服务器整合使用

冷冷zz 3年前 3.3k 16 3 Java Spring B...

Spring Authorization Server入门 (二) Spring Boot整合Spring Authorization Server

叹雪飞花 1年前 9.4k 35 107 Java

Spring Authorization Server的使用

huan1993 3年前 7.5k 21 6 Spring 后端

Spring Authorization Server入门 (十二) 实现授权码模式使用前后端分离的登录页面

叹雪飞花 1年前 6.5k 28 74 后端 Spring ... Spring

Linkerd Service Mesh 授权策略(Server & ServerAuthorization)

为少 2年前 2.1k 4 1 云原生 Service... 后端

Spring Authorization Server入门 (十九) 基于Redis的Token、客户端信息和授权确认信...

叹雪飞花 9月前 2.0k 9 19 Spring ... 后端 Redis

Spring 官方发起Spring Authorization Server 项目

码农小胖哥 4年前 4.6k 7 11 Java Spring B...

Spring Authorization Server入门 (十五) 分离授权确认与设备码校验页面

叹雪飞花 1年前 2.0k 14 10 Spring ... Spring Vue.js

Spring Authorization Server入门 (十七) Vue项目使用授权码模式对接认证服务

叹雪飞花 11月前 1.2k 9 18 Vue.js 安全 Spring ...

Spring Authorization Server入门 (九) Spring Boot引入Resource Server对接认证服务

叹雪飞花 1年前 2.4k 16 9 Spring Spring ...

鸭鸭笔记-Spring Authorization Server

鸭鸭世界第一可爱 10月前 1.3k 19 17 后端 Spring ... Java

稀土掘金 首页

探索稀土掘金