🍉Spring Authorization Server (5) 授权服务器【用户、客户端信息】扩展

爱吃西瓜的胖娃 2023-09-19 497 阅读6分钟 关注

> 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

一切要从授权服务的配置说起

> DefaultSecurityConfig
>
> java 复制代码
>
> ```java
> @Bean
> public UserDetailsService users() {
>         UserDetails user = User.withDefaultPasswordEncoder()
>                         .username("user1")
>                         .password("password")
>                         .roles("USER")
>                         .build();
>         return new InMemoryUserDetailsManager(user);
> }
> ```
>
> AuthorizationServerConfig
>
> java 复制代码

```
        RegisteredClient registeredClient = RegisteredClient.withId(UUID.randomUUID().t
                .clientId("messaging-client")
                .clientSecret("{noop}secret")
                .clientAuthenticationMethod(ClientAuthenticationMethod.CLIENT_S
                .authorizationGrantType(AuthorizationGrantType.AUTHORIZATION_CO
                .authorizationGrantType(AuthorizationGrantType.REFRESH_TOKEN)
                .authorizationGrantType(AuthorizationGrantType.CLIENT_CREDENTIA
                .redirectUri("http://127.0.0.1:8080/login/oauth2/code/messaging
                .redirectUri("http://127.0.0.1:8080/authorized")
                .postLogoutRedirectUri("http://127.0.0.1:8080/logged-out")
                .scope(OidcScopes.OPENID)
                .scope(OidcScopes.PROFILE)
                .scope("message.read")
                .scope("message.write")
                .clientSettings(ClientSettings.builder().requireAuthorizationCo
                .build();

        RegisteredClient deviceClient = RegisteredClient.withId(UUID.randomUUID().toStr
                .clientId("device-messaging-client")
                .clientAuthenticationMethod(ClientAuthenticationMethod.NONE)
                .authorizationGrantType(AuthorizationGrantType.DEVICE_CODE)
                .authorizationGrantType(AuthorizationGrantType.REFRESH_TOKEN)
                .scope("message.read")
                .scope("message.write")
                .build();

        // Save registered client's in db as if in-memory
        JdbcRegisteredClientRepository registeredClientRepository = new JdbcRegisteredC
        registeredClientRepository.save(registeredClient);
        registeredClientRepository.save(deviceClient);

        return registeredClientRepository;
    }
```

用户信息、客户端配置肯定不能是基于 Memory 存储是吧，特别是用户信息，客户端数据也不多，存内存影响不大，不过我们还是都存数据库。

最近公司都不用Mysql了，再是因为这次搭建的Spring Cloud 架构 整体都会用 PostgreSQL 去做持久化存储，用PostgreSQL 的原因很简单 优势比Mysql 更多 ，存储和查询、数据结构上也有更多的支持。

2022.0.0.0、Spring Cloud 2022.0.3 搭建的 oauth2 微服务架构。

PostgreSQL 此次涉及到的sql脚本

`sys_registered_client` 客户端信息表

sql 复制代码

```sql
DROP TABLE IF EXISTS sys_registered_client;
CREATE TABLE sys_registered_client (
 id varchar(64)  NOT NULL,
 client_id varchar(100)  NOT NULL,
 client_id_issued_at timestamp(6),
 client_secret varchar(200) ,
 client_secret_expires_at timestamp(6),
 client_name varchar(200)  NOT NULL,
 client_authentication_methods jsonb,
 authorization_grant_types jsonb,
 redirect_uris jsonb,
 post_logout_redirect_uris jsonb,
 scopes jsonb,
 client_settings json,
 token_settings json
)
;

-- ----------------------------
-- Records of sys_registered_client
-- ----------------------------
INSERT INTO sys_registered_client VALUES ('1702591381795115010', 'device-messaging-client
INSERT INTO sys_registered_client VALUES ('1703682313609162754', 'messaging-client', NULL

-- ----------------------------
-- Indexes structure for table sys_registered_client
-- ----------------------------
CREATE UNIQUE INDEX sys_registered_client_unique_index ON sys_registered_client USING btr
 client_id  pg_catalog.text_ops ASC NULLS LAST
);
COMMENT ON INDEX sys_registered_client_unique_index IS 'sys_registered_client 唯一索引';

-- ----------------------------
-- Primary Key structure for table sys_registered_client
```

sys_user 用户表

sql 复制代码

```sql
DROP TABLE IF EXISTS sys_user;
CREATE TABLE sys_user (
 create_time timestamp(6) NOT NULL DEFAULT timezone('UTC-8'::text, (now())::timestamp(0) w
 modified_time timestamp(6) DEFAULT timezone('UTC-8'::text, (now())::timestamp(0) without
 id int8 NOT NULL DEFAULT nextval('sys_user_id_seq'::regclass),
 name varchar(64)  NOT NULL,
 password varchar(255) ,
 phone varchar(11)  NOT NULL,
 mobile varchar(255)  NOT NULL,
 avatar varchar(255) ,
 status int2 NOT NULL DEFAULT 1
)
;
COMMENT ON COLUMN sys_user.create_time IS '创建时间';
COMMENT ON COLUMN sys_user.modified_time IS '修改时间';
COMMENT ON COLUMN sys_user.id IS 'id';
COMMENT ON COLUMN sys_user.name IS '用户名称';
COMMENT ON COLUMN sys_user.password IS '密码';
COMMENT ON COLUMN sys_user.phone IS '手机号(未加密)';
COMMENT ON COLUMN sys_user.mobile IS '手机号(加密)';
COMMENT ON COLUMN sys_user.avatar IS '头像';
COMMENT ON COLUMN sys_user.status IS '账号状态(0:无效；1:有效)';
COMMENT ON TABLE sys_user IS '用户表';

-- ----------------------------
-- Primary Key structure for table sys_user
-- ----------------------------
ALTER TABLE sys_user ADD CONSTRAINT sys_user_pkey PRIMARY KEY (id);
```

mysql8.0 +版本的sql脚本👉 https://github.com/WatermelonPlanet/watermelon-cloud/tree/master/watermelon-authorization/watermelon-authorization-user-core/doc/sql/mysql

用户存储扩展

java 复制代码

```java
public interface UserDetailsService {


  UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;

}
```

UserDetailsService 原来是一个接口，自定义一个接口实现，so easy 🤔

开干 🤓

java 复制代码

```java
@Component
@RequiredArgsConstructor
public class UserDetailsServiceImpl implements UserDetailsService {

   private final SysUserService sysUserService;

   private final PasswordEncoder passwordEncoder;

   @Override
   public UserDetails loadUserByUsername(String username) throws UsernameNotFoundExceptio
       //如今这个世界 我们肯定都用手机号登录的了
       SysUserDetailDto sysUser = sysUserService.findOneByPhone(username);
       if (sysUser == null) {
           throw new UsernameNotFoundException("手机号： " + username + "未注册！");
       }
       //todo 后续可自行修改和完善
       List<GrantedAuthority> authorityList = AuthorityUtils.createAuthorityList("/oauth2,
       SysUserDto sysUserDto = new SysUserDto();
       sysUserDto.setUsername(username);
       sysUserDto.setAuthorities(authorityList);
       sysUserDto.setId(sysUser.getId());
       sysUserDto.setAvatar(sysUser.getAvatar());
       sysUserDto.setPassword(passwordEncoder.encode(sysUser.getPassword()));
       sysUserDto.setStatus(sysUser.getStatus());
       sysUserDto.setPhone(sysUser.getPhone());
       return sysUserDto;
```

> 这个扩展没啥技术含量，是的吧， `SysUserService` 是 基于 mybatis-plus 定义的service ，这就ok了，是的，以上这个扩展都很简单的。

UserDetails 扩展

java 复制代码

```java
@Data
@JsonSerialize
@JsonIgnoreProperties(ignoreUnknown = true)
public class SysUserDto implements UserDetails, Serializable {

    private static final long serialVersionUID = SpringSecurityCoreVersion.SERIAL_VERSION_U
    //id
    private  Long id;
    //手机号(未加密)
    private  String phone;
    //用户名
    private  String username;
    //用户名
    private  String password;
    //头像
    private  String avatar;
    //账号状态(0:无效；1:有效)
    private  Integer status;
    //权限
    private Collection<GrantedAuthority> authorities;


    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return this.authorities;
    }

    @Override
    public String getPassword() {
        return this.password;
    }

    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }
}
```

需要用 `@JsonSerialize`、 `@JsonIgnoreProperties(ignoreUnknown = true)` 处理JSON序列化和反序列化问题。否则 security 会抛异常。

PasswordEncoder 需要注入了，在 `DefaultSecurityConfig` 注入。

java 复制代码

```java
@Bean
public PasswordEncoder passwordEncoder() {
  return PasswordEncoderFactories.createDelegatingPasswordEncoder();
}
```

为什么要注入 `PasswordEncoderFactories.createDelegatingPasswordEncoder()` 是因为 /login 登录时有密码验证是在 `DaoAuthenticationProvider` 中 进行密码匹配验证的，所以 `UserDetailsServiceImpl`

看看 PasswordEncoder createDelegatingPasswordEncoder 的内部

java 复制代码

```java
public static PasswordEncoder createDelegatingPasswordEncoder() {
          String encodingId = "bcrypt";
          Map<String, PasswordEncoder> encoders = new HashMap<>();
          encoders.put(encodingId, new BCryptPasswordEncoder());
          encoders.put("ldap", new org.springframework.security.crypto.password.LdapShaPa
          encoders.put("MD4", new org.springframework.security.crypto.password.Md4Passwor
          encoders.put("MD5", new org.springframework.security.crypto.password.MessageDig
          encoders.put("noop", org.springframework.security.crypto.password.NoOpPasswordE
          encoders.put("pbkdf2", Pbkdf2PasswordEncoder.defaultsForSpringSecurity_v5_5());
          encoders.put("pbkdf2@SpringSecurity_v5_8", Pbkdf2PasswordEncoder.defaultsForSpr
          encoders.put("scrypt", SCryptPasswordEncoder.defaultsForSpringSecurity_v4_1());
          encoders.put("scrypt@SpringSecurity_v5_8", SCryptPasswordEncoder.defaultsForSpr
          encoders.put("SHA-1", new org.springframework.security.crypto.password.MessageD
          encoders.put("SHA-256",
                          new org.springframework.security.crypto.password.MessageDigestP
          encoders.put("sha256", new org.springframework.security.crypto.password.Standar
          encoders.put("argon2", Argon2PasswordEncoder.defaultsForSpringSecurity_v5_2());
          encoders.put("argon2@SpringSecurity_v5_8", Argon2PasswordEncoder.defaultsForSpr
          return new DelegatingPasswordEncoder(encodingId, encoders);
    }
```

`PasswordEncoder` 默认是 **bcrypt** 对应的就是 BCryptPasswordEncoder，如果要替换 /login 中的 PasswordEncoder，有如下两种解决方案

①：再创建一个Filter 去做后续的验证流程，代码流程不走 `DaoAuthenticationProvider`

②：继承 `DaoAuthenticationProvider` 重写 additionalAuthenticationChecks() 方法，目的是重现注入 `passwordEncoder` 后再进行密码匹配。

客户端存储扩展

RegisteredClientRepository

```java
public interface RegisteredClientRepository {


  void save(RegisteredClient registeredClient);

  @Nullable
  RegisteredClient findById(String id);

  @Nullable
  RegisteredClient findByClientId(String clientId);

}
```

RegisteredClientRepository 也是一个接口，里面3个方法，spring 很喜欢用接口呢，我们撸一个实现就ok 😎

java 复制代码

```java
@Component
@RequiredArgsConstructor
public class MybatisRegisteredClientRepository implements RegisteredClientRepository {


   private static final String CLIENT_ID_NOT_EXIST_ERROR_CODE = "client not exist";

   private static final String ZONED_DATETIME_ZONE_ID = "Asia/Shanghai";

   private final SysRegisteredClientService sysRegisteredClientService;


   @Override
   public void save(RegisteredClient registeredClient) {
       SysRegisteredClientDto sysRegisteredClientDto = new SysRegisteredClientDto();
       sysRegisteredClientDto.setClientId(registeredClient.getClientId());
       sysRegisteredClientDto.setClientName(registeredClient.getClientName());
       sysRegisteredClientDto.setClientSecret(registeredClient.getClientSecret());
       if (registeredClient.getClientIdIssuedAt() != null) {
           sysRegisteredClientDto.setClientIdIssuedAt(registeredClient.getClientIdIssuedAt
       }
       if (registeredClient.getClientSecretExpiresAt() != null) {
           sysRegisteredClientDto.setClientSecretExpiresAt(registeredClient.getClientSecre
       }
       sysRegisteredClientDto.setClientAuthenticationMethods(registeredClient.getClientAut
       sysRegisteredClientDto.setAuthorizationGrantTypes(registeredClient.getAuthorizatio
       sysRegisteredClientDto.setRedirectUris(registeredClient.getRedirectUris());
```

```java
        sysRegisteredClientDto.setClientSettings(registeredClient.getClientSettings().getSe
        sysRegisteredClientService.saveClient(sysRegisteredClientDto);
    }

    @Override
    public RegisteredClient findById(String id) {
        SysRegisteredClientDto sysRegisteredClientDetailVo = sysRegisteredClientService.get
        if (sysRegisteredClientDetailVo == null) {
            throw new ClientAuthorizationException(new OAuth2Error(CLIENT_ID_NOT_EXIST_ERR
                    "Authorization client table data id not exist: " + id, null),
                    id);
        }
        return sysRegisteredClientDetailConvert(sysRegisteredClientDetailVo);
    }

    @Override
    public RegisteredClient findByClientId(String clientId) {
        SysRegisteredClientDto sysRegisteredClientDto = sysRegisteredClientService.getOneBy
        if (sysRegisteredClientDto == null) {
            throw new ClientAuthorizationException(new OAuth2Error(CLIENT_ID_NOT_EXIST_ERR
                    "Authorization client id not exist: " + clientId, null),
                    clientId);
        }
        return sysRegisteredClientDetailConvert(sysRegisteredClientDto);
    }

    /**
     * sysRegisteredClientDetailVo 转换为 RegisteredClient
     *
     * @param sysRegisteredClientDto
     * @return
     */
    private RegisteredClient sysRegisteredClientDetailConvert(SysRegisteredClientDto sysReg
        RegisteredClient.Builder builder = RegisteredClient
                .withId(sysRegisteredClientDto.getId())
                .clientId(sysRegisteredClientDto.getClientId())
                .clientSecret(sysRegisteredClientDto.getClientSecret())
                .clientIdIssuedAt(Optional.ofNullable(sysRegisteredClientDto.getClientIdIss
                        .map(d -> d.atZone(ZoneId.of(ZONED_DATETIME_ZONE_ID)).toInstant())
                        .orElse(null))
                .clientSecretExpiresAt(Optional.ofNullable(sysRegisteredClientDto.getClient
                        .map(d -> d.atZone(ZoneId.of(ZONED_DATETIME_ZONE_ID)).toInstant())
                        .orElse(null))
                .clientName(sysRegisteredClientDto.getClientName())
                .clientAuthenticationMethods(c ->
                        c.addAll(sysRegisteredClientDto.getClientAuthenticationMethods()
                                .stream().map(AuthorizationGrantType::new).collect(Collect
                ).redirectUris(r -> r.addAll(sysRegisteredClientDto.getRedirectUris()))
                .postLogoutRedirectUris(p -> p.addAll(sysRegisteredClientDto.getPostLogout
                .scopes(s -> s.addAll(sysRegisteredClientDto.getScopes()))
                .clientSettings(ClientSettings.builder().requireAuthorizationConsent(true)
//                 .tokenSettings(TokenSettings.builder().build());
        //todo clientSettings和 tokenSettings 根据需要后续自行修改
//                 .clientSettings(ClientSettings.withSettings(sysRegisteredClientDetailVo
        return builder.build();

    }
}
```

以上用户、客户端基于PostgreSQL扩展都搞定了，so easy😁 然后注释或删除掉 `DefaultSecurityConfig` 、 `AuthorizationServerConfig` 先前 @Bean 方式注入的 UserDetailsService、RegisteredClientRepository。

最后聊聊 watermelon-cloud中的模块的设计

> **watermelon-authorization** 授权服务模块
> **-watermelon-authorization-server** 【授权服务】
> **-watermelon-authorization-user-core** 【用户、客户端相关】
>
> 为什么要模块化去做呢?
> 原因时因为：关于持久层的代码写在 `watermelon-authorization-server` 授权服务中，从责任划分来说，用户信息、客户端相关不属于授权服务，授权服务肯定是只干授权的事情，所以将用户、客户端相关单独分一个模块， `watermelon-authorization-server` 授权服务依赖用户、客户端相关时，引入依赖即可。

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

🔥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

登录 / 注册 即可发布评论!

最热 最新

黔农黄地

重新UserDetails，扩展了登录用户的属性信息，比如电话、邮箱、部门等属性，登录后在授权端Principal.userifo里面可以获取到扩展信息，在客户端却不能获取到这些信息，这是为什么？老师能否增加一篇示例文章来指导一下！😳

8月前 点赞 1

爱吃西瓜的胖娃 作者 : 客户端是从token里面获取的用户信息，没有的原因是因为jwt里面没有包含其中的信息 最后一篇文章有
最后建议文章一步步看

8月前 点赞 回复

早中晚zz

高质量

9月前 点赞 评论

目录 收起

相关推荐

🍉Spring Authorization Server (6) 授权服务器 授权类型扩展
423阅读 · 0点赞

🍉Spring Authorization Server (1) 认证、授权、oauth2概念和流程初步介绍
1.2k阅读 · 4点赞

Spring Authorization Server的使用
7.5k阅读 · 21点赞

🍉Spring Authorization Server (2) 授权服务、资源服务、客户端核心配置讲解
1.3k阅读 · 3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 (9) 授权服务的授权信息存储方式扩展
爱吃西瓜的胖娃 9月前 877 3 3 Spring ... Spring ...

🍉Spring Authorization Server (10) 授权服务的JWK密钥对生成和JWT信息扩展

Spring Authorization Server 授权服务器
程序员Mark 1年前 6.5k 25 9 Spring ...

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 Password授权扩展
hundanli 5月前 386 1 2 Java Spring B...

Spring Authorization Server的使用
huan1993 3年前 7.5k 21 6 Spring 后端

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

Spring Authorization Server + Oauth2 配置认证服务器与资源服务器
张蕊是胖胖 1年前 3.3k 11 6 后端

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

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

Linkerd Service Mesh 授权策略(Server & ServerAuthorization)
为少 2年前 2.1k 4 1 云原生 Service... 后端