Spring Authorization Server (7) 第三方平台账号存储

爱吃西瓜的胖娃 2023-09-21 297 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

有这样一个常见的应用场景

> 当选择第三方登录的时候，第三方认证、授权成功后会回调到我方服务端，那么这个时候我们一定会将第三方平台账号&我方账号做一个关联，现在市面上都是这样玩的。例如你选择微信登录完了以后是不是还是得绑定手机号，这个绑定手机号的操作也就是做了第三方平台账号&我方系统账号做的关联操作。

当然除了扩展点，以下还会讲到架构设计😉

寻找扩展点

> 以gitee 为例， gitee授权成功后的回调 `http://192.168.56.1:9000/login/oauth2/code/gitee`

```java
public class OAuth2LoginAuthenticationFilter extends AbstractAuthenticationProcessingFilt

    public static final String DEFAULT_FILTER_PROCESSES_URI = "/login/oauth2/code/*";

    @Override
    public Authentication attemptAuthentication(HttpServletRequest request, HttpServletRes
            throws AuthenticationException {
        //...省略

        OAuth2LoginAuthenticationToken authenticationResult = (OAuth2LoginAuthenticationTo
        //...省略
    }
}
```

其中 `OAuth2LoginAuthenticationToken authenticationResult = (OAuth2LoginAuthenticationToken) this.getAuthenticationManager().authenticate(authenticationRequest);`

跟源码继续往下，对应的就是 `OAuth2LoginAuthenticationProvider`

```java
public class OAuth2LoginAuthenticationProvider implements AuthenticationProvider {

    private final OAuth2AuthorizationCodeAuthenticationProvider authorizationCodeAuthentic

    private final OAuth2UserService<OAuth2UserRequest, OAuth2User> userService;

    @Override
    public Authentication authenticate(Authentication authentication) throws Authenticatio
        //...省略
        OAuth2User oauth2User = this.userService.loadUser(new OAuth2UserRequest(
                loginAuthenticationToken.getClientRegistration(), accessToken, additionalP
        //...省略
    }
}
```

this.userService.loadUser() 继续跟到实现就是 `DefaultOAuth2UserService`

```java
public class DefaultOAuth2UserService implements OAuth2UserService<OAuth2UserRequest, OA
```

```
        //...省略

        //这个就是向gitee发起的请求了
        RequestEntity<?> request = this.requestEntityConverter.convert(userRequest);
        ResponseEntity<Map<String, Object>> response = getResponse(userRequest, request);
        Map<String, Object> userAttributes = response.getBody();
        Set<GrantedAuthority> authorities = new LinkedHashSet<>();
        authorities.add(new OAuth2UserAuthority(userAttributes));
        OAuth2AccessToken token = userRequest.getAccessToken();
        for (String authority : token.getScopes()) {
            authorities.add(new SimpleGrantedAuthority("SCOPE_" + authority));
        }
        //最后将gitee获取的用户信息 封装成一个 DefaultOAuth2User返回
        return new DefaultOAuth2User(authorities, userAttributes, userNameAttributeName);
    }
}
```

DefaultOAuth2UserService实际上是处理第三方平台返回的用户信息。

那如何去扩展 🤔?

要将信息存到我们自己系统，在返回之后去做处理就行了😁

直接继承 `DefaultOAuth2UserService` 去扩展 实现扩展 😀

第三方平台账号存储扩展

PostgrepSql 存储表

`sys_third_user` 第三方用户信息表

sql 复制代码

```sql
DROP TABLE IF EXISTS sys_third_user;
CREATE TABLE sys_third_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_third_user_id_seq'::regclass),
 unique_id varchar(128)  NOT NULL,
 name varchar(64)  NOT NULL,
```

```
 user_id int8 NOT NULL
)
;
COMMENT ON COLUMN sys_third_user.create_time IS '创建时间';
COMMENT ON COLUMN sys_third_user.modified_time IS '修改时间';
COMMENT ON COLUMN sys_third_user.id IS 'id';
COMMENT ON COLUMN sys_third_user.unique_id IS '第三方平台唯一id';
COMMENT ON COLUMN sys_third_user.name IS '用户名称';
COMMENT ON COLUMN sys_third_user.platform IS '平台类型(WX:微信；QQ:QQ)';
COMMENT ON COLUMN sys_third_user.avatar IS '头像';
COMMENT ON COLUMN sys_third_user.user_id IS '用户id';
COMMENT ON TABLE sys_third_user IS '第三方用户表';

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

AccountPlatform 枚举

java 复制代码

```java
public enum AccountPlatform {
    WX,
    QQ,
    GITEE,
    GITHUB;
}
```

OAuth2ThirdUserDto

> 定义第三方平台的用户信息实体，用构造方法的原因是因为想把此次扩展的定义为一个stater，不引入lombok，不是low 😂

```java
public class OAuth2ThirdUserDto implements Serializable {
    /**
     * 第三方平台唯一id
     */
    private String uniqueId;
    /**
     * 用户名称
     */
    private String name;
    /**
     * 平台类型(WX:微信；QQ:QQ)
     */
    private AccountPlatform platform;
    /**
     * 头像
     */
    private String avatar;


    public OAuth2ThirdUserDto(String uniqueId, String name, AccountPlatform platform, Stri
        this.uniqueId = uniqueId;
        this.name = name;
        this.platform = platform;
        this.avatar = avatar;
    }


    public String getUniqueId() {
        return uniqueId;
    }

    public void setUniqueId(String uniqueId) {
        this.uniqueId = uniqueId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public AccountPlatform getPlatform() {
        return platform;
    }
```

```
    }

    public String getAvatar() {
        return avatar;
    }

    public void setAvatar(String avatar) {
        this.avatar = avatar;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        OAuth2ThirdUserDto that = (OAuth2ThirdUserDto) o;
        return Objects.equals(uniqueId, that.uniqueId) && Objects.equals(name, that.name) {
    }

    @Override
    public int hashCode() {
        return Objects.hash(uniqueId, name, platform, avatar);
    }
}
```

OAuth2UserConvert

定义转换的接口，多个平台的情况下，采用一个策略模式

java 复制代码

```
//第三方用户转换接口定义
public interface OAuth2UserConvert {

    default AccountPlatform platform() {
        return null;
    }
    /**
     * 第三方用户信息统一转换为 OAuth2ThirdUserDto
     * @param oAuth2User
     * @param userNameAttributeName 额外的属性
```

```
    Pair<OAuth2ThirdUserDto, LinkedHashMap<String,Object>> convert(OAuth2User oAuth2User,S
}
```

GiteeOAuth2UserConvert

gitee的OAuth2UserConvert实现

java 复制代码

```java
public class GiteeOAuth2UserConvert implements OAuth2UserConvert {

    private final static String AVATAR_URL = "avatar_url";

    private final static String UNIQUE_ID = "id";

    private final static String NAME = "name";

    private final static String EMAIL = "email";

    private final static String PLATFORM = "platform";

    @Override
    public AccountPlatform platform() {
        return AccountPlatform.GITEE;
    }
    @Override
    public Pair<OAuth2ThirdUserDto, LinkedHashMap<String, Object>> convert(OAuth2User oAutl
        String avatarUrl = Optional.ofNullable(oAuth2User.getAttribute(AVATAR_URL)).map(Ob
        String uniqueId = Optional.ofNullable(oAuth2User.getAttribute(UNIQUE_ID)).map(Obje
        String name = Optional.ofNullable(oAuth2User.getAttribute(NAME)).map(Object::toStr
        String email = Optional.ofNullable(oAuth2User.getAttribute(EMAIL)).map(Object::toS
        Object nameAttributeValue = Optional.ofNullable(userNameAttributeName).map(oAuth2U

        LinkedHashMap<String, Object> userAttributesLinkedHashMap = new LinkedHashMap<>();
        //根据需要取所需要字段
        userAttributesLinkedHashMap.put(UNIQUE_ID, uniqueId);
        userAttributesLinkedHashMap.put(NAME, name);
        userAttributesLinkedHashMap.put(EMAIL, email);
        userAttributesLinkedHashMap.put(AVATAR_URL, avatarUrl);
        userAttributesLinkedHashMap.put(userNameAttributeName, nameAttributeValue);
        userAttributesLinkedHashMap.put(PLATFORM, this.platform().name());
        OAuth2ThirdUserDto oAuth2ThirdUserDto = new OAuth2ThirdUserDto(uniqueId, name, Acc
        return new Pair<>(oAuth2ThirdUserDto, userAttributesLinkedHashMap);
    }
}
```

OAuth2UserConvertContext

> OAuth2UserConvert 的context 管理

java 复制代码

```java
public class OAuth2UserConvertContext {

    private Map<AccountPlatform, OAuth2UserConvert> oAuth2UserConvertMap;
    /**
     * 加载 OAuth2UserConvert
     * @param oAuth2UserConvertList
     */
    public OAuth2UserConvertContext(List<OAuth2UserConvert> oAuth2UserConvertList) {
        this.oAuth2UserConvertMap = oAuth2UserConvertList.stream().collect(Collectors.toMap

    }
    /**
     * 获取实例
     * @param platform
     * @return
     */
    public OAuth2UserConvert getInstance(AccountPlatform platform) {
        if (platform == null) {
            throw new SystemException("平台类型不能为空");
        }
        OAuth2UserConvert oAuth2UserConvert = oAuth2UserConvertMap.get(platform);
        if (oAuth2UserConvert == null) {
            throw new SystemException("暂不支持[" + platform + "]平台类型");
        }
        return oAuth2UserConvert;
    }
}
```

Oauth2UserStorage

```java
//第三方平台保存接口定义
public interface Oauth2UserStorage {

    /**
     * 保存
     * @param auth2ThirdUserDto
     */
    void save(OAuth2ThirdUserDto auth2ThirdUserDto);

}
```

DefaultOauth2UserStorage

定义这个接口的好处在于，其他人也可以引入这个 stater ，然后就不用再去实现spring官方的接口去做了，实现 `DefaultOauth2UserStorage` 定义的接口就可以了。

java 复制代码

```java
public class DefaultOauth2UserStorage implements Oauth2UserStorage {
    @Override
    public void save(OAuth2ThirdUserDto auth2ThirdUserDto) {

    }
}
```

ExtDefaultOAuth2UserService

用户信息保存默认实现 ， 来看看吧，直接继承 `DefaultOAuth2UserService`

java 复制代码

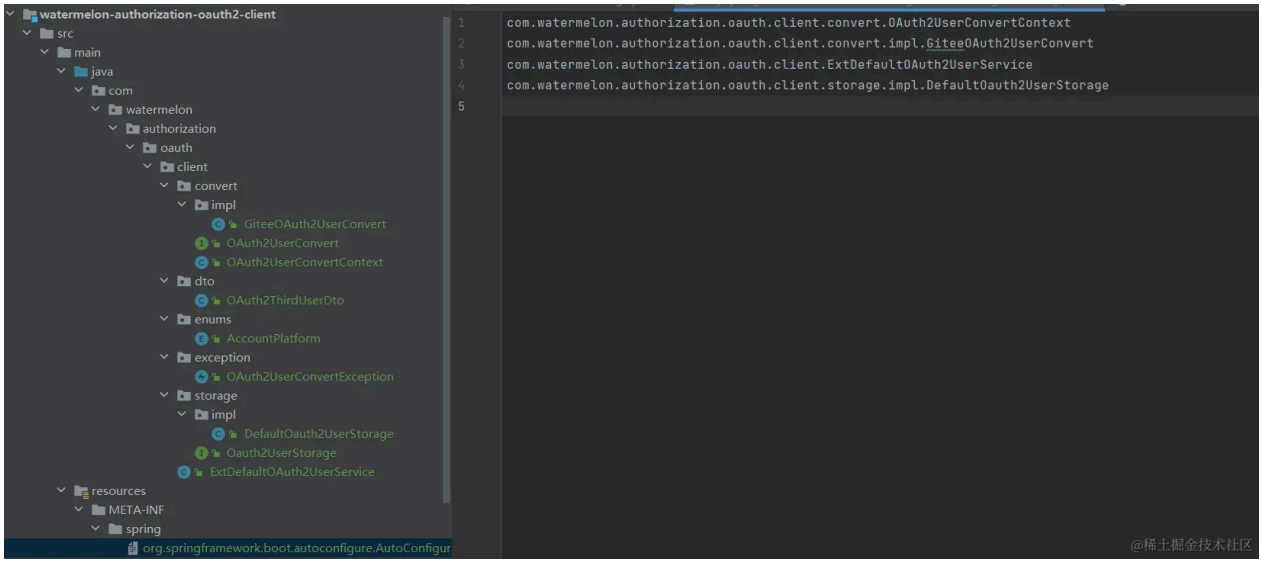
```java
public class ExtDefaultOAuth2UserService extends DefaultOAuth2UserService {

    public final OAuth2UserConvertContext oAuth2UserConvertContext;

    public final Oauth2UserStorage oauth2UserStorage;

    public ExtDefaultOAuth2UserService(OAuth2UserConvertContext oAuth2UserConvertContext, (
        this.oAuth2UserConvertContext = oAuth2UserConvertContext;
```

```
12      @Override
13      public OAuth2User loadUser(OAuth2UserRequest userRequest) throws >OAuth2AuthenticationI
14          OAuth2User oAuth2User = super.loadUser(userRequest);
15          String userNameAttributeName = userRequest.getClientRegistration().getProviderDeta:
16                  .getUserNameAttributeName();
17          AccountPlatform platform = >this.loginPlatformConvert(userRequest.getClientRegistr:
18          //将 OAuth2User 根据不同的平台 转成统一的 第三方用户了
19          Pair<OAuth2ThirdUserDto, LinkedHashMap<String, Object>> oAuth2ThirdUserConvertPair
20                  .convert(oAuth2User, userNameAttributeName);
21          LinkedHashMap<String, Object> userAttributes = oAuth2ThirdUserConvertPair.getValue(
22          //这个地方保存逻辑了
23          oauth2UserStorage.save(oAuth2ThirdUserConvertPair.getKey());
24          return new DefaultOAuth2User(oAuth2User.getAuthorities(), userAttributes, userNameA
25      }
26
27      /**
28       * registrationId 转换平台枚举
29       * @param registrationId
30       * @return
31      */
32      synchronized private AccountPlatform loginPlatformConvert(String registrationId) {
33          return switch (registrationId) {
34              case "gitee" -> AccountPlatform.GITEE;
35              case "wechat" -> AccountPlatform.WECHAT;//todo Convert
36              case "qq" -> AccountPlatform.QQ;//todo Convert
37              default -> throw new OAuth2UserConvertException("暂不支持该客户端[" + registrati
38          };
39      }
40  }
```

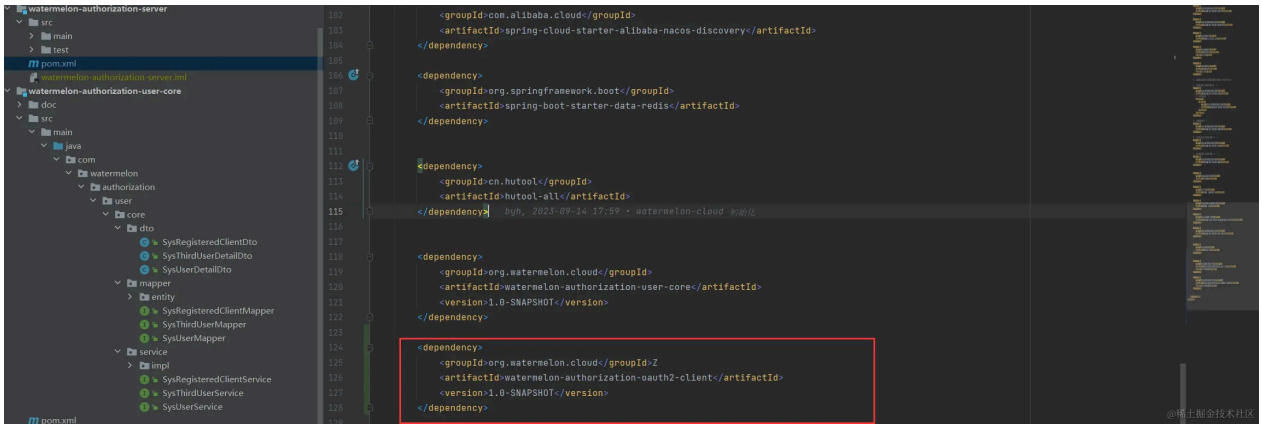
扩展的class就全部搞定了

> 在 `watermelon-authorization` 模块下重写定义了 `watermelon-authorization-oauth2-client` 模块，这样的好处是可以单独作为一个 stater 给其他任何项目使用。

watermelon-authorization-oauth2-client完整版本就是这样了，那这样的stater就完成了



那就在**watermelon-authorization-server** 中引入我们自己定义好的 **watermelon-authorization-server** 依赖。



watermelon-authorization-server 中使用，就需要实现Oauth2UserStorage接口就能搞定了

基于PostgrepSql 去存储 用的mybatis-plus 那就这样实现一个吧

java 复制代码

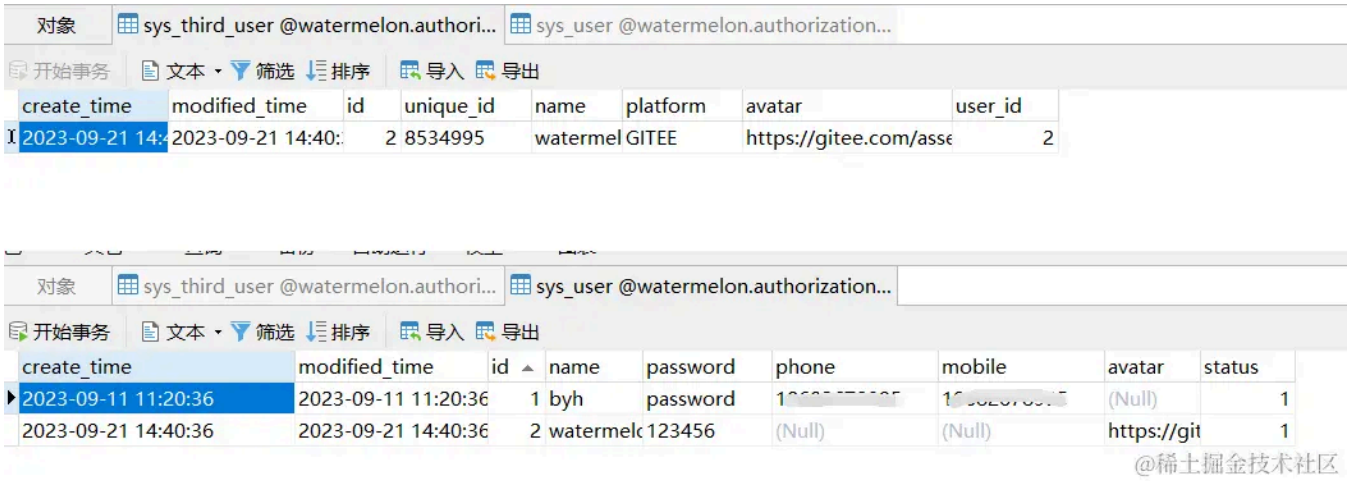
```java
@Primary
@Service("mybatisOauth2UserStorage")
@RequiredArgsConstructor
public class MybatisOauth2UserStorage implements Oauth2UserStorage {

    private final SysThirdUserService sysThirdUserService;

```

```
10      @Override
11      public void save(OAuth2ThirdUserDto auth2ThirdUserDto) {
12          SysUseAddDto sysUseAddDto = new SysUseAddDto();
13          sysUseAddDto.setName(auth2ThirdUserDto.getName());
14          sysUseAddDto.setAvatar(auth2ThirdUserDto.getAvatar());
15          sysUseAddDto.setStatus(1);
16          Long sysUserId = sysUserService.save(sysUseAddDto);
17          SysThirdUserAddDto sysThirdUserAddDto = new SysThirdUserAddDto();
18          sysThirdUserAddDto.setUniqueId(auth2ThirdUserDto.getUniqueId());
19          sysThirdUserAddDto.setAvatar(auth2ThirdUserDto.getAvatar());
20          sysThirdUserAddDto.setPlatform(auth2ThirdUserDto.getPlatform());
21          sysThirdUserAddDto.setName(auth2ThirdUserDto.getName());
22          sysThirdUserAddDto.setUserId(sysUserId);
23          sysThirdUserService.save(sysThirdUserAddDto);
24      }
25  }
```

其他任何工程只要实现 `Oauth2UserStorag` 接口实现就可以了，这也是定义stater 的目的所在-为了通用。

如果有其他平台的用户转换，可以实现 `OAuth2UserConvert` 去自行实现即可

最后看看测试的最终结果



**完整的代码在： `https://github.com/WatermelonPlanet/watermelon-cloud` **中。

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 0

登录 / 注册 即可发布评论!

暂无评论数据

目录 收起

相关推荐

Spring Authorization Server (8) 授权服务的默认认证方式扩展
488阅读 · 2点赞

spring-authorization-server系列--Oauth介绍
28阅读 · 0点赞

Spring Authorization Server 密码模式
2.5k阅读 · 4点赞

Spring Authorization Server 入门教程
1.1k阅读 · 1点赞

Spring Authorization Server 全新授权服务器整合使用
3.3k阅读 · 16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
爱吃西瓜的胖娃 9月前 796 1 评论 Spring ... Spring ...

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

Spring Authorization Server 授权服务器

程序员Mark 1年前 6.5k 25 9 Spring ...

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...

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 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 (七) 登录添加图形验证码

叹雪飞花 1年前 3.6k 21 4 Spring ...

Spring Authorization Server 集成MySQL

hundanli 5月前 555 2 评论 Java Spring B...

Spring Authorization Server入门 (十六) Spring Cloud Gateway对接认证服务

叹雪飞花 11月前 3.5k 19 47 Spring ... Spring ... 安全

spring-authorization-server令牌放发源码解析

冷冷zz 2年前 2.8k 17 评论 后端 架构 面试

Spring Authorization Server入门 (十) 添加短信验证码方式登录

叹雪飞花 1年前 4.1k 22 9 Spring Spring ...