使用 AOP 进行权限控制

目录

1 背景

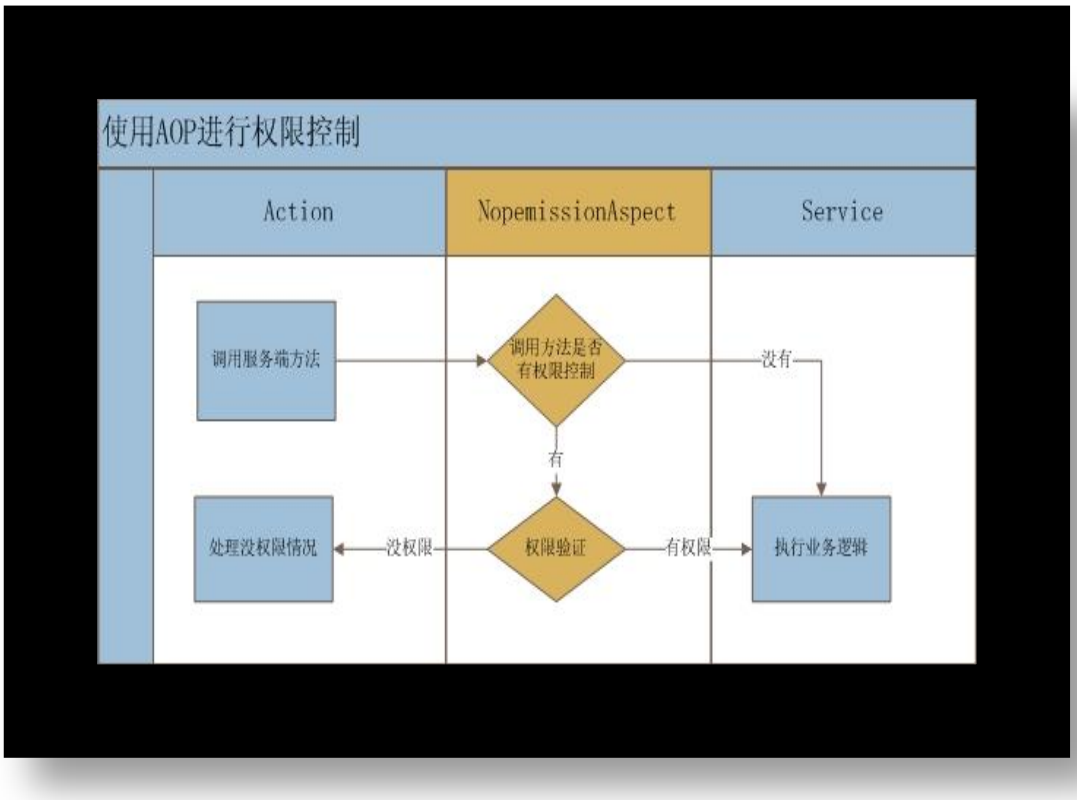
- 代码分散：权限控制代码散落在各个类中。
- 耦合度高：权限判断和业务逻辑耦合在一起。

2 解决方案

- 代码集中：将所有权限控制代码放在一个切面里。
- 松耦合：使用 AOP 进行拦截，判断用户是否有权限访问某个方法。

3 整体设计

3.1 流程设计



当 Action 调用 Service 方法的时候，系统首先会判断当前方法是否有权限控制，然后判断用户是否有权限访问该方法，如果没有则抛出异常。

3.2 基于注解的拦截

那么如何判断调用的方法是否有权限控制呢？通过在方法上配置了一个注解，来表示需要某个权限才能访问，如：

```
@VersionPermission(version=POPULAR,experienceType=ExperienceType.FOLLOW)
public boolean createFollow()
```

当访问该方法前，会判断用户是否具有注解里配置的权限，如没有权限，则抛出 NoPermissionException 异常，让调用者自行处理没有权限的情况。

4 如何实现

本文使用 AspectJ 注解来定义切面。

4.1 引入 AspectJ 注解

在 POM 里加入

```
<dependency>
    <groupId>XX.XX.XX</groupId>
    <artifactId>misc.aspectj</artifactId>
    <version>1.6.6</version>
    <type>libd</type>
</dependency>
```

注意：如果是 JDK1.6 版本，必须使用 aspectj1.6.6 版本。

4.2 启用 Spring 对 AspectJ 的支持

在 bean.xml 里加入

```
<aop:aspectj-autoproxy/>
```

4.3 增加权限注解

VersionPermission 注解用来表示用户是否具有某个产品版本或者某种体验券。

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface VersionPermission {
    /**
     * 用户的产品版本
     * @return
     */
    ProductType version();
    /**
     * 体验券类型
     * @return
     */
    ExperienceType experienceType();
}
```

产品版本：

```
public enum ProductType {
    NOTOPENED,// 没开通
    POPULAR// 标准版
    }
```

体验券类型：

```
public enum ExperienceType {
        /**
```

```
         * 智能跟进
         */
        FOLLOW;
    }
```

4.4 增加权限切面

权限切面。作为权限控制的总控，所有权限代码都应该写在这里。以下只是展示对版本的权限控制。

```
@Aspect
public class PermissionAspect {

    /**
     * 切入点，拦截所有配置VersionPermission注解的方法
     */
    @Pointcut("@annotation(security.VersionPermission)")
    public void allVersionPermission() {
    }

    /**
     * 版本权限检查，检查用户是否拥有指定产品版本或指定体验券
     *
     * @param versionPermission   版本权限注解
     * @throws NoPermissionException   没有权限抛出该异常
     */
    @Before("PermissionAspect.allVersionPermission() && @annotation(versionPermission)")
    public void checkVersionPermission(VersionPermission versionPermission) {

        if (!checkVersionPermission(versionPermission.version(),
                versionPermission.experienceType())) {
            throw new NoPermissionException("no permission");
        }

    }
```

- @Aspect:表示当前类是一个 AOP 切面。
- @Pointcut 表示我要拦截哪些方法。
- @Before 表示在某个方法执行之前，我要执行下面这个方法。

注意：需要将 PermissionAspect 作为 bean 配置到 bean.xml 里。

5 如何扩展

如果要增加新的权限控制，需要新增一个注解（见 4.3）和一个切面（见 4.4），然后在方法上配置多个注解，如下：

```
@VersionPermission(version=POPULAR,experienceType=ExperienceType.FOLLOW)
@RightCodePermission(code=”createFollow”)
public boolean createFollow()
```

AOP 可以应用的场景非常多，如方法参数判断，性能监控，事务处理和日志记录等。