Spring中的AOP

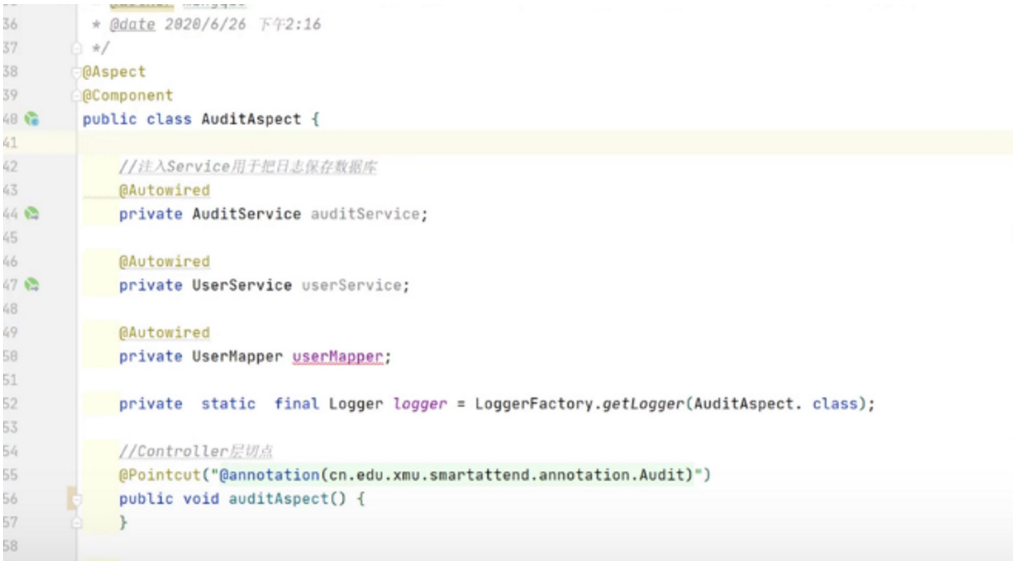
2021年10月9日　　17:13

Spring框架如何实现AOP

Spring框架主要通过动态代理机制来实现AOP，可以在动态代理的基础上，根据AOP Aspect中的定义去把沙子掺在动态代理上，这就使得在Spring中，AOP的切点基本都是在方法上面

具体实现

首先要定义一个切面Aspect，它并没有定义一个新的语法，而是利用了类的语法
我们使用**@Aspect**的注解，写在类的前面，这样类和类的属性都是可以用的
通过这样一个注解，来定义说这个类是一个切面



```
 * @date 2020/6/26 下午2:16
 */
@Aspect
@Component
public class AuditAspect {

    //注入Service用于把日志保存数据库
    @Autowired
    private AuditService auditService;

    @Autowired
    private UserService userService;

    @Autowired
    private UserMapper userMapper;

    private  static  final Logger logger = LoggerFactory.getLogger(AuditAspect. class);

    //Controller层切点
    @Pointcut("@annotation(cn.edu.xmu.smartattend.annotation.Audit)")
    public void auditAspect() {
    }
```

在切面中，我们要定义两个东西——Advice和PointCut
Point是一个条件，用来判断哪些连接点是我们要掺沙子的地方
使用**@Pointcut**标签来定义切点，条件写在注解的属性中
这个注解要么放在属性前，要么放在方法前，我们通常**把这个注解放在方法前**，而**这个方法是没有内容的**
因为这个方法就是为了去定义这样的一个切点

```
//Controller层切点
@Pointcut("@annotation(cn.edu.xmu.smartattend.annotation.Audit)")
public void auditAspect() {
}
```

Advice其实是一个方法，是用一组注解来定义的，需要定义Advice的执行时间，方法体中的内容就是Advice需要做的事情

```java
/**
 * 前置通知 用于拦截Controller层记录用户的操作
 *
 * @param joinPoint 切点
 */
@Before("auditAspect()")
public void doBefore(JoinPoint joinPoint) {
}

//配置controller环绕通知,使用在方法aspect()上注册的切入点
@Around("auditAspect()")
public Object around(JoinPoint joinPoint){
    long start = System.currentTimeMillis();
    String operationTime = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss").format(new Date());
    MethodSignature ms = (MethodSignature) joinPoint.getSignature();
    Method method = ms.getMethod();
    // 获取注解的参数信息
    Audit auditAnno = method.getAnnotation(Audit.class);
    String operationName = auditAnno.name();
    boolean login = auditAnno.login();
```

切点的定义

切点是**定义在一个方法前面**的，定义时要用到AspectJ的定义符，我们常用的有四个定义符：

定义符	描述
args()	Join point是方法，且参数是某些特定类型
@args()	Join point是方法，且其参数用特定注解标注
execution()	Join point是方法，其方法名称满足特定条件
@annotation	Joint point用特定注解标注

@annotation表示方法前加了特定的注解

例：

execution：方法满足特定条件

第一个*：方法需要满足什么返回类型

xmu.restdemo.service.*：包和类名，这里表示restdemo的service包下的所有方法

(..)：方法可以有任意参数

主要有加@Audit标签的地方，就是要调用JoinPoint的地方

Advice注解

主要有五个标签，用来定义Advice在什么时候执行

1. @Before：JoinPoint方法执行之前

2. @After：JoinPoint方法执行之后
3. @AfterReturning：在JoinPoint方法执行完成，并且结果返回之后
4. @AfterThrowing：在JoinPoint方法抛出特定异常之后
5. @Around：在JoinPoint方法执行之前和执行之后

至于Advice要做什么，就是Advice方法中定义的部分

我们通常用Advice做以下这些事情：

在方法执行之前去判断方法的参数是否符合一定要求（这里的Advice是在方法执行之前就检查，不仅是检查参数合法性，还可以检查参数里的内容，比如一个枚举型，或者检查一个用户的权限，如果检查不合法，那么我们就直接把它拦下来，进行一定处理，再交给之后的方法去执行）

拿到一些值，再把它投到方法的参数上

在方法执行完成之后，把方法的返回值做一些过滤、修改，再返回给客户端