

## 公共字段自动填充

=====

## 公共字段自动填充

-----

### ### 1.问题分析

在进行新增员工、菜品功能时需要设置创建时间、更新时间、创建人、修改人字段，在编辑员工和菜品分类时需要设置修改时间、修改人的字段。这些字段属于公共字段，也就是很多表都会有的字段。如下：

**序号**	**字段名**	**含义**	**数据类型**
1	create\_time	创建时间	datetime
2	create\_user	创建人id	bigint
3	update\_time	修改时间	datetime
4	update\_user	修改人id	bigint

目前，在项目中这些字段都是在每一个业务方法中进行赋值，编码相对冗余、繁琐，那能不能对于这些公共字段在某个地方统一处理，来简化开发呢？

### ### 2.实现思路

在不改变原有业务代码，实现功能增强，就要使用AOP来实现功能增强，来完成公共字段填充功能。

#### #### 分析：

- > \*\*先确定切入点：要增强谁\*\*
- >
- >
- > 使用excution，根据方法名选择
- >

>  
> 使用@annotation, 使用注解选择  
>  
>  
> 连接使用: 连接符 && ||  
>  
>  
> **\*\*确定什么时候增强: 通知类型\*\***  
>  
>  
> @Before: 前置通知, 在目标方法执行之前先执行  
>  
>  
> @AfterReturing: 后置通知, 在目标方法正常执行完成之后在执行  
>  
>  
> @AfterThrowing: 异常通知, 在目标方法抛出异常之后在执行  
>  
>  
> @After: 最终通知, 在目标方法执行之后, 无论有没有异常都会执行  
>  
>  
> @Around: 环绕通知  
>  
>  
> **\*\*确定要怎么增强, 通知方法里要做的事情\*\***

#### 实现:

> **\*\*先确定切入点: 要增强谁 (切入点表达式) \*\***  
>  
>  
> Mapper里的`insert`和`update`方法, 方法名可能并不规范, 根据方法名选择切入点可能不合适。  
>  
>  
> 为了更灵活的选中在增强的方法, 使用注解的方式: 哪个方法要增强, 就在哪个方法上加注解 @注解(操作类型)  
>  
>  
> **\*\*需要确定如何增强: 通知\*\***  
>  
>  
> **\*\*通知类型: \*\*** 什么时候对Mapper的方法增强? 在目标方法执行之前增强, 使用@Before

```
>
>
> **如何增强: **
>
>
> 获取目标方法上的注解, 得到其中的操作类型
>
>
> ***判断如果目标方法操作类型是insert新增: ***
>
>
> 获取方法的实参entity对象
>
>
> 给entity对象设置createTime、updateTime、createUser、updateUser属性值
>
>
> ***如果目标方法操作类型是update修改: ***
>
>
> 获取方法的实参entity对象
>
>
> 给entity对象设置updateTime、updateUser属性值
```

#### 编码步骤:

1. 先定义注解 @AutoFill, 给注解添加一个参数是OperationType
2. 在需要增强的方法上添加注解 @AutoFill(OperationType.INSERT或UPDATE)
3. 编写切面类:

类上加注解@Component @Aspect

类里通过方法:

```
...
@Before("execution(选中mapper的方法) && @annotation(autoFill)")
public void autoFill(JoinPoint jp, AutoFill autoFill){
    根据autoFill获取操作类型
    如果操作类型是INSERT:
```

```
    获取entity对象的createTime、updateTime、createUser、updateUser的  
    set方法  
    使用反射分别调用每个set方法，设置值  
    如果操作类型是UPDATE：  
    获取entity对象的updateTime、updateUser的set方法  
    使用反射分别调用每个set方法，设置值  
}
```

...

### ### 3.代码实现

#### 1. \*\*自定义注解@AutoFill\*\*

...

```
@Target(ElementType.METHOD)  
@Retention(RetentionPolicy.RUNTIME)  
public @interface AutoFill {  
    //数据库操作类型： UPDATE INSERT  
    OperationType value();  
}
```

...

**\*\*数据库操作类型\*\***

...

```
public enum OperationType {  
  
    /**  
     * 更新操作  
     */  
    UPDATE,  
  
    /**  
     * 插入操作  
     */  
    INSERT  
}
```

...

2. \*\*在Mapper层 插入、修改方法添加注解@AutoFill\*\*
3. \*\*自定义切面AutoFillAspect\*\*

```

...
@Aspect
@Component
public class AutoFillAspect {
    /*
     * 1. 在切入点表达式里：如果想要缩小spring的扫描范围，可以使用多切入点表达式连接 连接符 &&
     * 选择要增强的方法：
     *     只扫描mapper层里的方法 并且 有AutoFill注解的方法
     * 2. 在注解方式切入点的通知方法，如果想在通知方法里获取到注解对象可以写成：
     *     2.1 在方法上直接加注解类型的形参
     *     2.2 修改切入点表达式
     *     原来：@annotation(注解的全限定类名)
     *     改成：@annotation(方法的注解类型形参名)
     * 注意：如果通知方法有多个形参的话，那么JoinPoint类型的参数要放到第一个
     *
     *
     * Java里如果要调用一个方法，有两种方式
     * 方式1：对象名.方法名(实参)
     * 方式2：反射调用方法
     *     先获取目标类的字节码
     *     在获取想要调用的方法
     *     最后反射执行这个方法
     */
    @Before("execution(* com.sky.mapper.*.*(..)) && @annotation(autoFill)")
    public void autoFill(JoinPoint joinPoint,AutoFill autoFill){//JoinPoint 得到要被增强的目标方法
        //获取目标方法的实参
        Object[] args = joinPoint.getArgs();
        // 为了提高程序的健壮性，加一个判断：如果方法没有实参，就什么都不做
        if(args == null || args.length == 0){
            return;
        }
        Object entity = args[0];
        Class clazz = entity.getClass();

        //如果目标方法操作类型是INSERT：设置四个值，否则是操作Update，设置两个值
        if(autoFill.value() == OperationType.INSERT){

```

