关于两个常见注解 @Override & @Deprecated

> 注解：是一种面向对象的注释

问：你为什么要写注释

- 起到一个标记和标注的作用
- 方便我们自己后续或者别人查看你代码的时候能够读得懂

```
// name="yykk" limit=10 timout=1
@Limiter(name="yykk",limit=10,timout=1 )
public void unlock() {
    lock.unlock();
}
```

问：为什么要要去用注解呢？

- 就是因为：我们可以通过反射获取这个类，获取方法，获取属性，获取包名、获取参数。它们都可以通过反射获取到注解信息。
- 也同时告诉你一个道理：注解可以定义在这些类，方法、属性、包名、参数中。

问：注解给谁看呢？怎么获取呢？

- 注解处理和注释都起到一个标记的作用，但是注释没办法获取，但是注解可以通过获取类，方法，属性，参数，包名进行获取
- 一句话：给反射看，用反射来获取。

问：场景

- 架构中spring框架，获取你自己去进行拦截通用处理的时候
- 日志拦截需要明确告诉方法执行的逻辑是什么？ @Log + LogAspect(这个类中肯定通过类，方法、属性、包名、参数任意一种方式获取注解信息，进行业务员逻辑处理。)
- 权限控制 @CheckLogin + AuthAspect (这个类中肯定通过类，方法、属性、包名、参数任意一种方式获取注解信息，进行业务员逻辑处理。)
- 限流控制等 @Limiter + LimterAspect(这个类中肯定通过类，方法、属性、包名、参数任意一种方式获取注解信息，进行业务员逻辑处理。)

```
// 保存用户方法
@Log(desc="保存用户方法")
public void saveuser() {

}

@Log(desc="修改用户方法")
public void 修改user() {

}
```

@Override

它是所以子类继承父类的抽象方法的一种标记。在子类的重写方法上写上@Override 告诉子类这个方法是重写父类的方法，这样更新明确。

- 当然你也可以不写，但是建议写上去，因为往往在开发中，子类的方法很多，重写的方法很多，所以为了区分建议大家写上@Override

@Deprecated

- 如果@Deprecated类，代表这个类，即将废弃和使用
- 如果@Deprecated方法，代表这个方法，即将废弃和使用
- 如果@Deprecated属性，代表这个属性，即将废弃和使用

总结：

- 如果未来在开发中，如果那你类即将要改写，要重构，或者有新的替代的解决方案，你可以把旧的类，方法，属性增加@Deprecated告诉的开发者，这个可能未来有新的替代解决方案，告诉可以未来关注新的解决方案，
- 如果被标记@Deprecated的类，方法，属性还是可以使用。建议开发者去下载最新的替代解决方案。把替换掉。