企业级框架 Spring

Unit06

Spring AOP

Spring AOP

- Aspect Oriented Programming 被称为面向方面编程或面向切面编程，它可以在不修改原有组件代码逻辑情况下追加新增功能。
- AOP关注点是切面，通过配置可以动态将共通处理添加到切面位置。好处是实现组件重复利用，将共通组件与目标对象解耦，提高灵活性。

Spring AOP

* Aspect切面

  指的是共通处理点，可以将功能切入到多个目标对象方法上

* JoinPoint连接点

  指的是切入组件在目标对象上作用点

* Pointcut切入点

  切入点是连接点的集合，采用表达式指定

Spring AOP

* Target目标对象
* Advice通知

指的是切面组件在连接点上执行动作的时机(例如：执行事务，或记录日志)

* AutoProxy动态代理

采用了AOP之后，容器返回的对象是代理对象。用户在使用时，由代理对象调用切入组件和目标对象的功能。

a. 目标对象有接口采用JDK代理、

b. 目标对象没有接口采用CGLIB代理

Spring AOP

Spring提供了5种类型的通知，具体如下

* 前置通知

  <aop:before>在目标方法调用之前执行。不能阻止后续执行，除非抛出异常

* 后置通知

  <aop:after-returning>在目标方法调用之后执行。目标方法**正常结束**才执行。如果抛出异常，则不执行。

* 最终通知

  <aop:after>在目标方法调用之后执行。目标方法**正常或异常**都执行。

Spring AOP

* 异常通知

  <aop:after-throwing>在目标方法调用发生异常之后执行。

* 环绕通知

  <aop:around>在目标方法调用之前和之后执行。

Spring AOP

* 5种类型的通知，在内部调用的时机：

```
try{
    环绕前置处理
    调用前置通知
    调用目标对象方法
    环绕后置处理
    调用后置通知
}catch(Exception e){
    调用异常通知
}finally{
    调用最终通知
}
```

Spring AOP

- AOP注解配置使用的方法如下
  - 创建方面组件
  - 声明方面组件

    <aop:aspectj-autoproxy proxy-target-class="true"/>

    使用@Component注解将其声明为组件

    使用@Aspect注解将其声明为方面组件
  - 切入方面组件

    在方面组件的方法上，使用通知注解和切入点表达式将方面组件作用到目标组件的方法上

    @Before("within(com.xdl.controller..*)")

    public void log() { ... ... }

总结和答疑