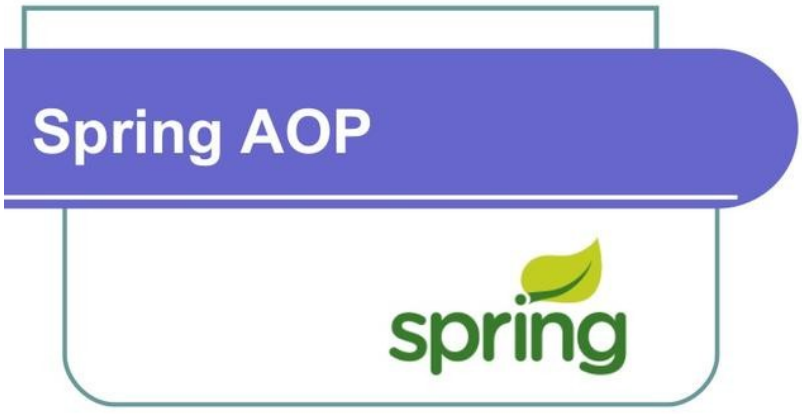
46 Spring AOP 总结及面试热点题目集萃

更新时间：2020-09-04 12:00:04

> 从不浪费时间的人，没有工夫抱怨时间不够。——杰弗逊



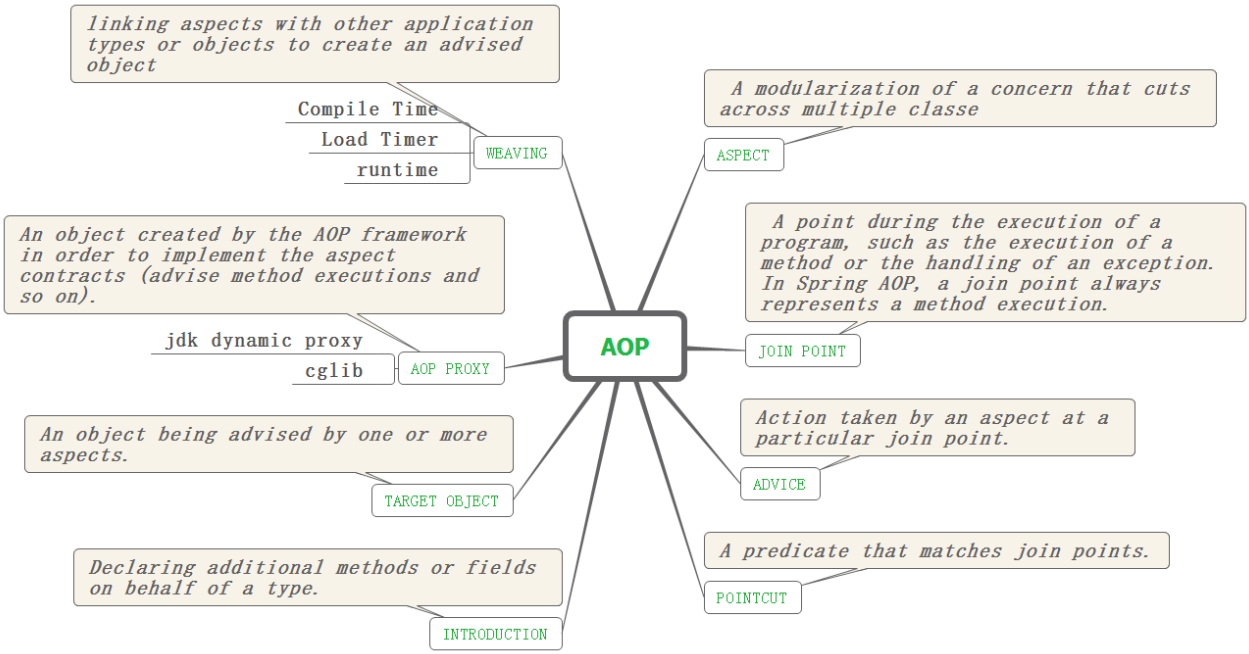
Spring AOP

Spring AOP总结

Spring AOP术语

AOP（Aspect-Oriented Programming），即面向切面编程，它与 OOP（Object-Oriented Programming, 面向对象编程）相辅相成，提供了与 OOP 不同的抽象软件结构的视角。在 OOP 中，我们以类（Class）作为我们的基本单元，而 AOP 中的基本单元是 Aspect（切面）。

AOP 给人难以理解的一个关键点是它的概念比较多，而且坑爹的是，这些概念经过了中文翻译后，变得面目全非，相同的一个术语，在不同的翻译下，含义总有着各种莫名其妙的差别。鉴于此，我们直接拿官方的说明来看。



形象描述 AOP 可以参看《图说AOP—妈妈再也不担心我概念混淆了》。

Spring AOP通用应用场景

1. Authentication（认证）：检测用户是否登录，未登录则返回登录页面；
2. Caching（缓存）；
3. Context passing（上下文传递）；
4. Error handling（错误处理），如 SpringMVC 中 @ControllerAdvice 注解；
5. Lazy loading（延迟加载）；
6. Debugging（调试）；
7. logging，tracing，profiling and monitoring（记录跟踪，优化，校准）；
8. Performance optimization（性能优化）；
9. Persistence（持久化）；
10. Resource pooling（资源池）；
11. Synchronization（同步）；
12. Transactions（事务）；
13. Logging（日志）。

Spring AOP的使用

使用 Spring AOP 可以基于两种方式，一种则是中规中矩的xml配置方式，另一种是比较方便和强大的注解方式。

XML 方式示例：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:aop="http://www.springframework.org/schema/aop"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
 http://www.springframework.org/schema/aop
 http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">

 <aop:aspectj-autoproxy />

 <!-- Target -->
 <bean id="hello" class="com.davidwang456.test.HelloService" />

 <!-- Aspect -->
 <bean id="logAspect" class="com.davidwang456.test.LogAspect" />
</beans>
```

注解方式：

```
package com.davidwang456.test;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.core.annotation.Order;
@Aspect
@Order(5)
public class LogAspect{
 @Before("execution(* sayHello(..))")
 public void beforeHello() {
  System.out.println("how are you !");
 }
}
```

> **Tips**：多个 Aspect 可以通过实现 ordered 接口或者 @Order 注解来控制 Aspect 的级别，级别越小，越先执行。

参见上节《6-7：Spring中多个AOP如何协调执行？》

Spring AOP 的原理与设计模式

策略模式

Spring AOP 内部使用 Cglib 或者 JDK 动态代理来生成代理类，Spring AOP 的策略模式实现有两种，默认使用 JdkDynamicAopProxy，使用 CGLIG 的三种情况：

- ProxyConfig 中的optimize标识被置为 true；
- ProxyConfig 中的 proxyTargetClass 标识被置为 true；
- 目标类没有可用的代理接口即目标类没有实现接口。

参见《Spring AOP策略模式使用及示例实战》。

代理模式

由于对接口而不是类进行编程是一种良好的实践，所以业务类通常实现一个或多个业务接口。所以想实现 AOP 可以利用 JDK 的动态代理。JDK 的动态代理有两个主类：

java.lang.reflect.**Proxy**：创建代理实例；
java.lang.reflect.**InvocationHandler**：增强的业务逻辑写在里面。

参加《离开 Spring AOP，我们如何实现 AOP 功能？》

Spring AOP扩展

学习 Spring AOP，AspectJ 的使用是个绕不过去的坎。少了这一部分，一些 Spring AOP 的源码或者原理总少了点什么，不太容易懂。《离开了SpringAOP，我们如何切面编程？》这篇文章通过抛开 Spring AOP 框架单独来使用 Aspect 来加深大家对 AOP 的认识。

Spring AOP踩坑经历

当需要在 Spring MVC Controller 层设置 AOP 时：

在低版本（3.1 之前）的 Spring 中，那么需要将 `<aop:aspectj-autoproxy proxy-target-class="true"/>` 配置到 dispatcher-servlet.xml（MVC 文件）当中并且设置包扫描规则为 **"use-default-filters="true"**。 将横切关注点封装为切面，切面中方法的注入是根据切点表达式来决定的。

高版本的 Spring 中，Spring 支持将 DispatcherServlet 注入到根 ApplicationContext，而不用创建自己的 WebApplicationContext，这时使用同一套 ApplicationContext 就不会出现上面的问题了。

参看《Spring控制器Controller如何设置AOP？》

Spring AOP热点面试题目集萃

1. 说一下你对 Spring AOP 的理解

在软件业，AOP 为 Aspect Oriented Programming 的缩写，意为：面向切面编程，通过预编译方式和运行期间动态代理实现程序功能的统一维护的一种技术。AOP 是 OOP 的延续，是软件开发中的一个热点，也是 Spring 框架中的一个重要内容，是函数式编程的一种衍生范型。利用 AOP 可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率。

2. Spring AOP 中 Aspect 是什么？

参考《图说 AOP—妈妈再也不担心我概念混淆了》。

3. 解释一下 Spring AOP 中关注点（concern）和横切点关注点（cross-cutting concern）的不同之处

关注点是我们希望在应用程序的特定模块中具有的行为。关注点可以定义为我们想要实现的功能。

横切关注点是一个适用于整个应用程序的关注点，它影响整个应用程序。例如：日志、安全和数据传输适用于应用程序的每个模块，因此它们是横切关注点。

4. spring AOP 中 Join Point 是什么？

参考《图说 AOP—妈妈再也不担心我概念混淆了》。

5. Spring AOP 中 Advice 是什么？

参考《图说 AOP—妈妈再也不担心我概念混淆了》

6. Spring AOP 中 Pointcut 是什么？

参考《图说 AOP—妈妈再也不担心我概念混淆了》

7. Spring AOP 中 Introduction 是什么？

Introduction 允许我们向现有类添加新方法或属性。

8. Spring AOP 中 Target object 是什么？

参考《图说 AOP—妈妈再也不担心我概念混淆了》

9. Spring 中的代理是什么？

代理是在将通知应用到目标对象后创建的对象。当您考虑客户端对象时，目标对象和代理对象是相同的。

10. Spring AOP 有哪些自动代理的不同类型？

BeanNameAutoProxyCreator：bean名称自动代理创建器；

DefaultAdvisorAutoProxyCreator：默认通知者自动代理创建器；

MetadataAutoproxying：元数据自动代理织入，将切面和其他应用类型或对象连接起来创建一个通知对象的过程。

11. Spring AOP 中织入是什么？织入有哪些不同的应用？

参考《图说 AOP—妈妈再也不担心我概念混淆了》

织入是将切面与其他类型应用程序或对象链接起来以创建 Advised 对象的过程。织入可以在编译时、加载时或运行时完成。

12. 解释一下 Spring AOP 中基于 XML Schema 的切面实现

Spring AOP 常用的实现方式有两种：**采用声明的方式来实现（基于 XML），采用注解的方式来实现（基于 AspectJ）**。

基于 XML Schema 的切面实现一般可以配置 Pointcut，如下图：

```
<!-- 声明一个业务类 -->
<bean id="userManager" class="com.spring.service.impl.UserManagerServiceImpl">
    <property name="name" value="lixiaoxi"></property>
</bean>

 <!-- 声明通知类 -->
<bean id="aspectBean" class="[redacted]" />

<aop:config>
 <aop:aspect ref="aspectBean">
    <aop:pointcut id="pointcut" expression="execution(* [redacted].*(..))"/>

    <aop:before method="doBefore" pointcut-ref="pointcut"/>
    <aop:after-returning method="doAfterReturning" pointcut-ref="pointcut" returning="result"/>
    <aop:after method="doAfter" pointcut-ref="pointcut" />
    <aop:around method="doAround" pointcut-ref="pointcut"/>
    <aop:after-throwing method="doAfterThrowing" pointcut-ref="pointcut" throwing="ex"/>
  </aop:aspect>
</aop:config>
```

13. 解释一下 Spring AOP 基于 @AspectJ 的 Aspect 实现机制

使用 @AspectJ 声明的注解是使用 JDK5 定义的注解形式，满足 Java 的规范。其实现为 AnnotationAwareAspectJAutoProxyCreator。

14. Spring AOP 的通知类型有哪些？

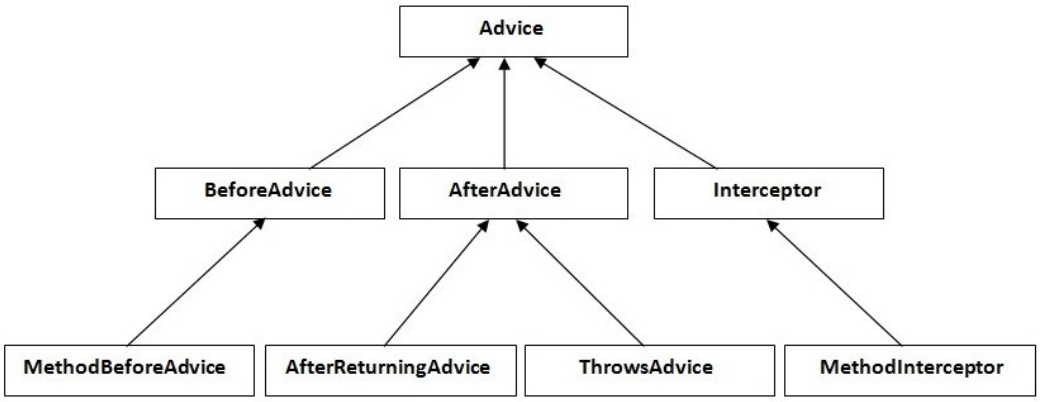
Spring AOP 通知（advice）分成 5 类：

- **前置通知（Before advice）**：在连接点前面执行，前置通知不会影响连接点的执行，除非此处抛出异常；
- **正常返回通知（After returning advice）**：在连接点正常执行完成后执行，如果连接点抛出异常，则不会执

行；

- **异常返回通知（After throwing advice）**：在连接点抛出异常后执行；
- **返回通知（After (finally) advice）**：在连接点执行完成后执行，不管是正常执行完成，还是抛出异常，都会执行返回通知中的内容；
- **环绕通知（Around advice）**： 环绕通知围绕在连接点前后，比如一个方法调用的前后。这是最强大的通知类型，能在方法调用前后自定义一些操作。

> **Tips:** 环绕通知还需要负责决定是继续处理 join point（调用 ProceedingJoinPoint 的 proceed 方法）还是中断执行。



15. 什么是 AOP 联盟?

AOP 联盟是一个开源项目，旨在促进 AOP 的采用。AOP 联盟的目标是定义一组通用的组件和接口，以改进不同 AOP 实现之间的互操作性。

16. 在 Spring AOP 中 proxy-target-class 属性起到什么作用?

要强制使用 CGLIB 代理，请将元素的 proxy-target-class 属性的值设置为 true。

```
<aop:config proxy-target-class="true">
 <!-- other beans defined here... -->
</aop:config>
```

17. How Spring AOP works?

Spring AOP 是基于代理的。Spring 使用 JDK 代理（最好是在代理的目标至少实现一个接口时）或 CGLIB 代理（当目标对象不实现任何接口时）来为给定的目标 bean 创建代理。Spring AOP 在 runtime 时织入切面。不过，您可以通过 AspectJ 设置使 Spring 来在加载时织入切面。

18. Spring AOP 中主要适用了哪些设计模式?

AOP 主要基于代理设计模式。它还使用了策略、单例、模板方法和装饰类设计模式。

19. Spring AOP 和 AspectJ 有哪些不同点？

默认情况下，Spring AOP 在 runtime 时织入切面，而 AspectJ 在 complile 时织入切面。Spring AOP 无法适用于 final 类或 final/static 方法，因为它可能无法通过子类创建代理。AspectJ 可以在 final 类中织入，因为它在 compile time 时织入切面，而不依赖于代理模式。

}

45 Spring中多个AOP如何协调执行？

47 Spring总结及常见面试问题与答案集锦