#### 一、基础

1. AOP功能：aop希望把分散在各个业务逻辑代码中的相同代码通过横向切割的方式抽取到一个独立的模块中，还业务逻辑一个清晰的世界。
2. AOP术语：

1）连接点：程序执行的某个特定位置，spring仅支持方法的连接点。连接点由两部分组成：方法表示的程序执行点、用相对位置表示方位。

2）切点：使用类和方法作为连接点的查询条件，连接点是方法执行前、后。而切点定位到某个方法

3）增强：增强是织入目标连接点上的一段程序代码。

4）目标对象：要增强的目标类。

5）织入：将增强添加到目标类的具体连接点上的过程。分为编译器织入，类装载期织入、动态代理织入；spring采用动态织入，AspectJ采用编译器织入和类装载期织入。

6）代理：一个类被AOP织入增强后，就产生一个结果类，它是融合了原类和增强逻辑的代理类。

7）切面：由切点和增强组成。

#### 基于@AspectJ和Schema的AOP

1.注解

@Retention(RetentionPolicy.RUNTIME)//声明注解的保留期限是在运行期间  而SOURCE：保留在源码文件中；CLASS：类的字节码文件中；  
@Target(ElementType.METHOD)//表示注解只能在目标类的方法上；TYPE、FIELD,METHOD,PARAMETER...  
public @interface Test {  
    //成员必须以无参、无抛出异常的方式声明；可以通过default指定默认值；如果只有一个成员，必须取名为value()；  
    boolean value() default true;  
}

2.使用@AspectJ

1）定义切面

@Aspect       //通过注解将其标识为一个切面  
public class PreGreetingAspect{  
   @Before("execution(\* greetTo(..))")//定义切点和增强类型  
   public void beforeGreeting(){     //增强的横切逻辑  
      System.out.println("How are you");  
   }  
}

2）配置使用@AspectJ

xmlns:aop="http://www.springframework.org/schema/aop"  
xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd  
          http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd">  
   <aop:aspectj-autoproxy/>

Waiter target = new NaiveWaiter();  
AspectJProxyFactory factory = new AspectJProxyFactory();  
factory.setTarget(target);//设置目标对象  
factory.addAspect(PreGreetingAspect.class);//添加切面  
Waiter proxy = factory.getProxy();//生成织入切面的代理对象  
proxy.greetTo("John");  
proxy.serveTo("John");

<aop:aspectj-autoproxy/>自动为spring容器中那些标注@AspectJ切面的Bean创建代理。有个proxy-target-class属性，默认为false，表示使用JDK动态代理，当设置为true时表示使用CGLib动态代理。

**3.@AspectJ语法基础**

1）通配符：\*匹配任意字符，但是只能匹配一个元素、 ..匹配任意字符，可以匹配多个元素、+表示按类型匹配指定类的所有类，必须在类的后面。

2）运算符：&& || !

3）增强类型：@Before前置增强、@AfterReturning后置增强、@Around环绕增强、@AfaterThrowing抛出增强、@After不管是抛出还是正常退出都增强、@DeclareParents引介增强（给服务员增加个收银员）；

4）切点函数之@annotation()表示标注了某个注解的所有方法

    @AfterReturning("@annotation(com.smart.anno.NeedTest)")  
    public void atAnnotaionTest(){  
     System.out.println("atAnnotaionTest() executed!");  
    }  
    @NeedTest  
   public void serveTo(String clientName){  
     System.out.println("NaiveWaiter:serving "+clientName+"...");  
   }

  切点函数之execution()

execution(<修饰符> <返回类型> <方法名>(<参数>) )

**4.基于Schema配置切面**

<?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-4.0.xsd  
           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd">  
  
   <aop:config proxy-target-class="true">  
       <aop:advisor advice-ref="testAdvice"  pointcut="execution(\* com..\*.Waiter.greetTo(..))"/>    
      <aop:aspect ref="adviceMethods">  
         <aop:before method="preGreeting"  
            pointcut="target(com.smart.NaiveWaiter) and args(name)"  
            arg-names="name" />  
         <aop:after-returning method="afterReturning"  
            pointcut="target(com.smart.SmartSeller)" returning="retVal" />  
         <aop:around method="aroundMethod"  
            pointcut="execution(\* serveTo(..)) and within(com.smart.Waiter)" />  
         <aop:after-throwing method="afterThrowingMethod"  
            pointcut="target(com.smart.SmartSeller) and execution(\* checkBill(..))"  
            throwing="iae" />  
         <aop:after method="afterMethod"  
            pointcut="execution(\* com..\*.Waiter.greetTo(..))" />  
  
      </aop:aspect>  
   </aop:config>  
    <bean id="testAdvice" class="com.smart.schema.TestBeforeAdvice"/>  
   <bean id="adviceMethods" class="com.smart.schema.AdviceMethods" />  
   <bean id="naiveWaiter" class="com.smart.NaiveWaiter" />  
   <bean id="naughtyWaiter" class="com.smart.NaughtyWaiter" />  
   <bean id="seller" class="com.smart.SmartSeller" />  
</beans>

<aop:config>内可以定义多个切面，<aop:aspect ref="增强方法的bean">表示一个切面，切面可以包含多个增强<aop:befor pointcut="execution()" method="">

<Advisor>仅包含一个切点和一个增强

<aop:advisor advice-ref="增强类的bean" pointcut="execution(\* com..\*.Waiter.greeto(..))"/>