Spring中对AOP的4中实现方法

# Spring提供了4种实现AOP的方式：

## 经典的基于代理的AOP(单独文件介绍)

## 基于注解驱动的切面开发方式

## 纯POJO切面

## 注入式AspectJ切面

# 基于注解驱动的切面开发方式

用AspectJ注解声明切面

　　要在Spring中声明AspectJ切面，只需要在IOC容器中将切面声明为bean实例。当在Spring IOC容器中初始化AspectJ切面之后，Spring IOC容器就会为那些与AspectJ切面相匹配的bean创建代理。

**在AspectJ注解中，切面只是一个带有@AspectJ注解的Java类**

**通知是标注有某种注解的简单的Java方法**

# 经典的基于代理的AOP

## Spring支持五种类型的通知：

**Before(前)**  org.apringframework.aop.**MethodBeforeAdvice**

**After-returning(返回后)** org.springframework.aop.**AfterReturningAdvice**

**After-throwing(抛出后)** org.springframework.aop.**ThrowsAdvice**

**Arround(周围)** org.aopaliance.intercept.**MethodInterceptor**

**Introduction(引入)** org.springframework.aop.**IntroductionInterceptor**

## 具体的实现：

主要是把ProxyFactoryBean的属性学习，这样就可以顺利的配置xml文件了。

这种方式需要利用Spring的AOP包，同样需要依赖**aopalliance.jar**包，不过不依赖aspectjrt.jar和aspectweaver.jar包。

**两种方法：代理接口和对类进行代理。**

利用**ProxyFactoryBean**进行代理。下面介绍该类。

org.springframework.aop.framework.ProxyFactoryBean

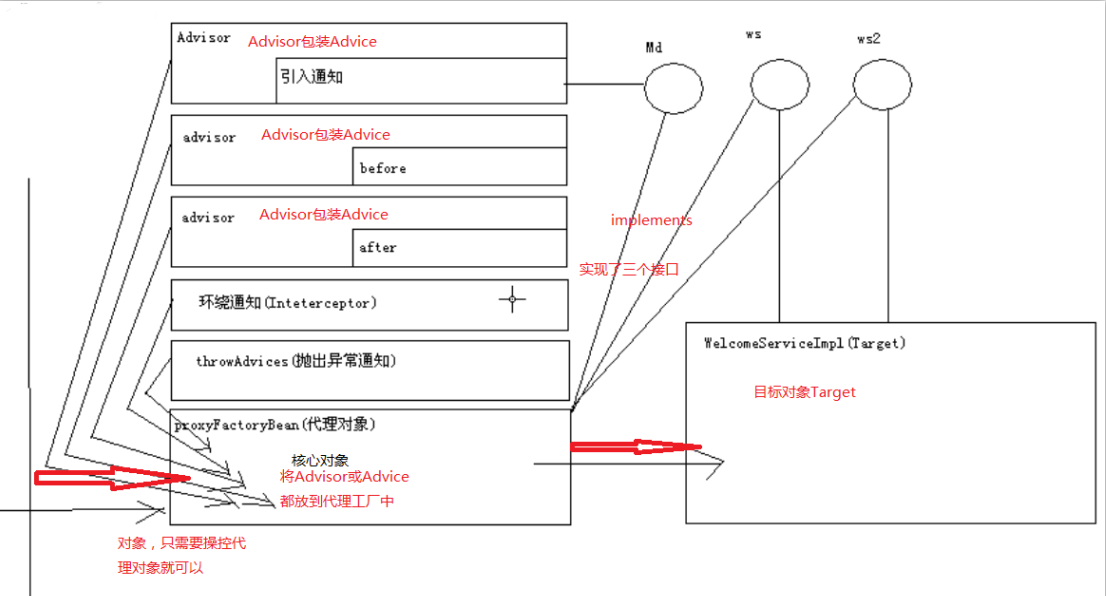
**代理接口：JDK的动态代理策略。**

1.创建通知：实现这几个接口，把其中的方法实现了

2.定义切点和通知者：在Spring配制文件中配置这些信息

3.使用ProxyFactoryBean来生成代理

## 图解：



# ProxyFactoryBean代理工厂对象

## 简单介绍

org.springframework.aop.framework.ProxyFactoryBean

用于经典的基于代理的AOP实现方式。

## ProxyFactory与ProxyFactoryBean的区别

两者都是Class类，具体什么区别不知道。

## ProxyFactoryBean介绍

存在于包org.springframework.aop.framework



继承关系：



实现的接口：





## 作用：

**FactoryBean** implementation that builds an AOP proxy based on beans in Spring BeanFactory.

FactoryBean是一个接口。

## ProxyFactoryBean的属性（重要）

在配置文件中配置**ProxyFactoryBean**，一般需要配置下面几个属性。

一般接口代理的只会用到三个：proxyInterfaces、interceptorNames、target。

### proxyInterfaces：代理的接口；（通过代理接口方式）

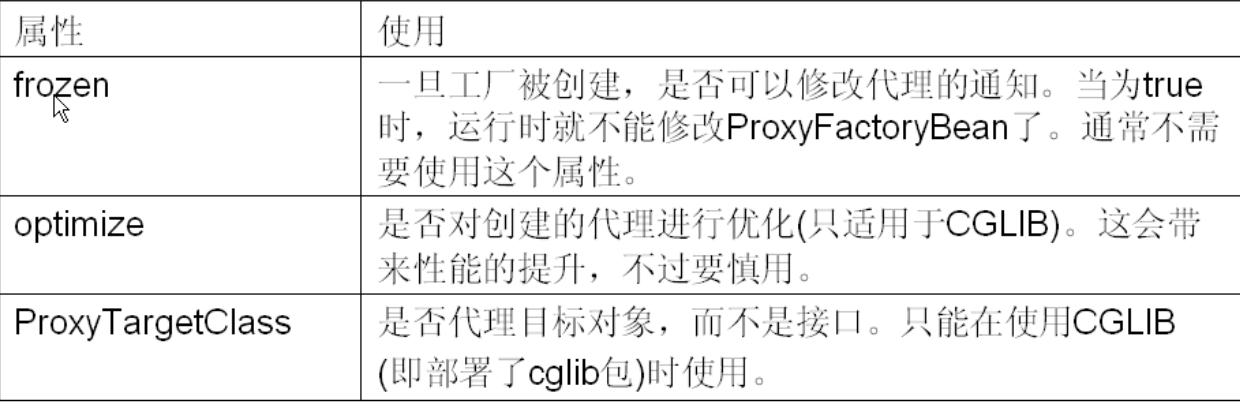
### interceptorNames：说明有哪些拦截器Interceptor、Advisor、Advice等通知类的名字。

### target：代理的目标对象，具体的一个Bean对象，该Bean实现了proxyInterfaces中声明的接口。

### proxyTargetClass：直接代理类。

### singleton、aopProxyFactory、exposeProxy、frozen、optimize等一般不常用。





使用示例：

