# 1 AOP

## AOP是什么

AOP是面向切面编程，通过动态代理的方式为程序添加统一的公共功能。

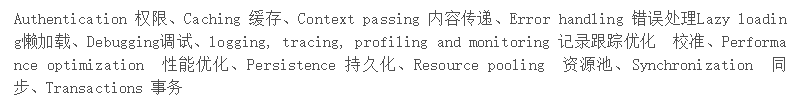
## AOP的优点

1. 拓展功能不需要对源代码进行修改操作。
2. 解耦，将公共或重复的逻辑从源代码中抽离出来。

## AOP的应用场景

权限，缓存，日志，事务等。

图1



## 1.4 AOP术语

原文链接：<https://www.cnblogs.com/lihuidu/p/5802611.html>

通知/增强（Advice）：就是想植入的功能。

连接点（JoinPoint）：Spring允许通知的地方。Spring只支持方法连接点，包括方法前，方法后，方法抛出异常等。

切点（PointCut）：不是所有的方法都需要加入通知，从连接点中选取需要加入增强的连接点，这些加入增强的连接点就是切点。

切面（Aspect）：切点和通知的结合。切点表示在哪里植入，通知表示植入的功能。

目标（target）：被通知的类。

代理（proxy）：植入通知后的目标类。

## 1.5 AOP 实现方式

原文链接：<https://www.cnblogs.com/best/p/5736422.html>

### 1.5.1 基于XML配置的AOP

目标类（图1），通知（图2），切面集成切点和通知（图3）。

图1

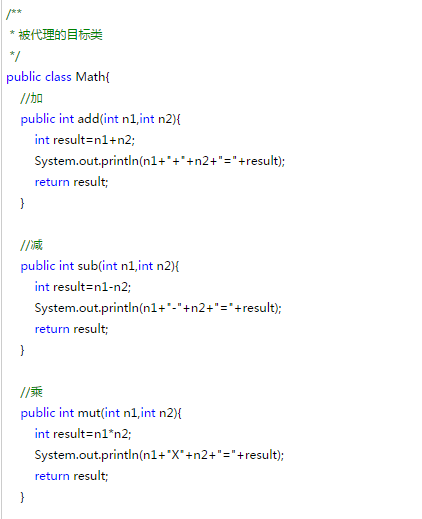


图2

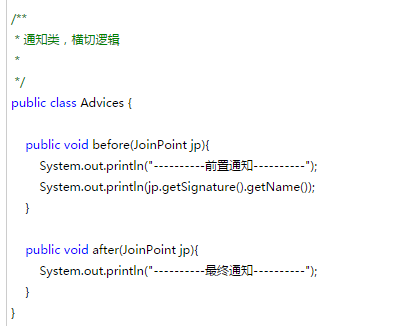


图3



### 1.5.2 基于注解的AOP

在增强中直接定义切面等逻辑。

图1



# 2 IOC

## IOC是什么

IOC是控制反转，传统设计中，通常由调用者来创建被调用者的实例对象。在Spring中将这项工作由Spring完成，不由调用者完成，

# 3 Spring bean的作用域

1. singleton：单例模式，Spring的默认bean作用域，每个bean只有1个实例。

图1



1. prototype：原型模式，每次获取bean（无论何种方式）时，都创建一个bean的实例。
2. request：请求模式，每次的HTTP请求获取bean时，都创建一个bean的实例。且该实例仅在当前HTTP请求中有效，请求结束后，该实例会销毁。
3. session：会话模式，同一个HTTP SESSION，返回同一个bean的实例，不同的SESSION返回不同的实例。实例仅在当前SESSION中有效。
4. global session：全局会话模式，同一个全局的HTTP SESSION，返回同一个bean的实例。该模式仅在portlet context（使用Portlet容器）中有效。

# 4 Spring事务

## 4.1 Spring事务的类型

1. 编程式事务：在代码中硬编码（使用TransactionTemplate（推荐）或者底层的PlatformTransactionManager），不推荐使用。
2. 声明式事务：建立在AOP之上，本质是对目标方法前后进行拦截，然后在目标方法开始之前创建或加入一个事务，目标方法结束后根据执行情况提交或回滚事务。

声明式事务又分为：基于XML（tx和aop）和基于注解（@Transactional）的声明式事务。

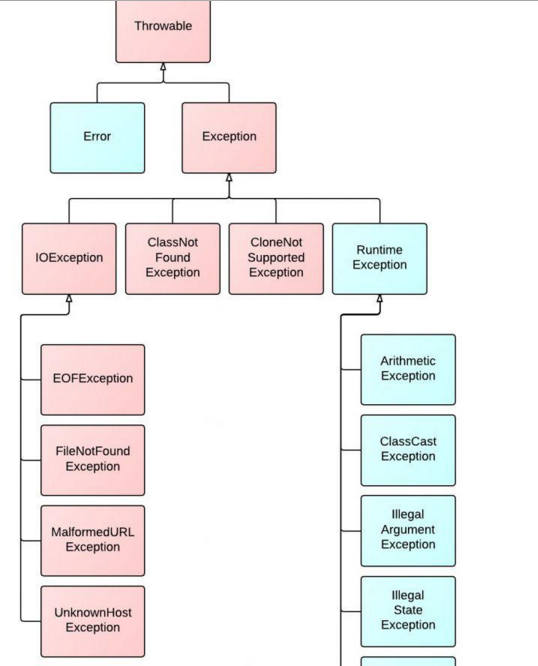
## 4.2 声明式事务的缺点

只能做到方法级别，无法细粒度到代码块级别，但可以将代码库独立为方法解决。

## 4.2 @Transaction的rollbackFor

经测试，rollbackFor不写的时候，只回滚运行时异常RuntimeException和错误Error，如图1的蓝色部分。

图1



## 4.3 Spring的事务传播行为

### 4.3.1 PROPAGATION\_REQUIRED

PROPAGATION\_REQUIRED：Spring默认的事务传播行为，如果当前存在事务，则加入该事务；如果当前不存在事务，则创建一个事务。

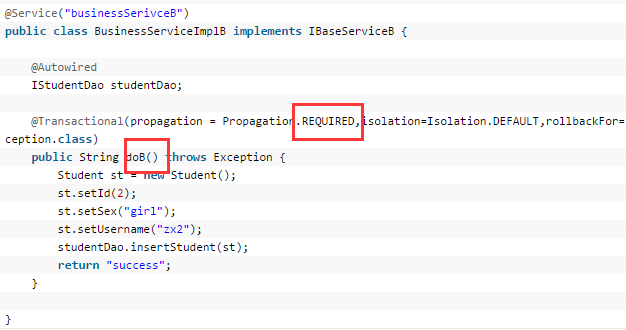
（注意：加入该事务表示处于同一个事务中）

如图1和图2，A和B处于同一个事务，B虽然提交了，但因为A抛出了异常，所以A和B都要回滚。

图1



图2



### 4.3.2 PROPAGATION\_SUPPORTS

PROPAGATION\_SUPPORTS：如果当前存在事务，则加入该事务；如果当前不存在事务，则以非事务执行。

### 4.3.3 PROPAGATION\_MANDATORY

PROPAGATION\_MANDATORY：如果当前存在事务，则加入该事务；如果当前不存在事务，则抛出异常。

### 4.3.4 PROPAGATION\_REQUIRED\_NEW

PROPAGATION\_REQUIRED\_NEW：无论当前是否存在事务，都创建一个事务。

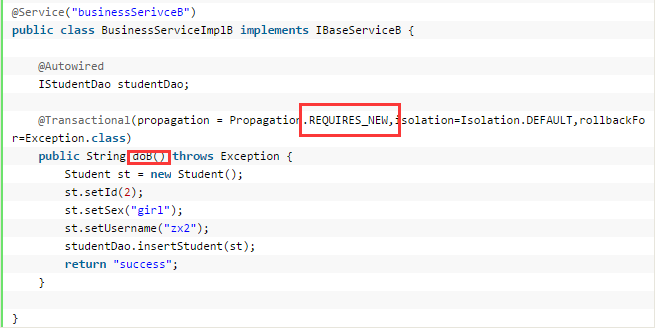
（注意：创建事务表示处于2个不同的事务中）

如图1和图2，A和B处于不同的事务，B提交了，A因为除以0的操作抛出异常，A回滚了，但B不会回滚，因为A和B在不同的事务中。

图1



图2



### 4.3.5 **PROPAGATION\_NEVER**

PROPAGATION\_NEVER：以非事务执行；如果当前存在事务，则抛出异常。

### 4.3.6 PROPAGATION\_NOT\_SUPPORTS

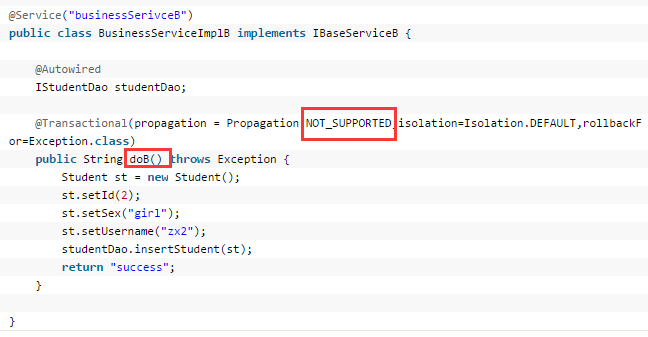
PROPAGATION\_NOT\_SUPPORTS：以非事务执行；如果当前存在事务，则挂起该事务。

如图1和图2，B以非事务执行，并提交了，A以事务的方式执行，抛出异常后A回滚了，但B因为没有事务，所以没有回滚。

图1



图2

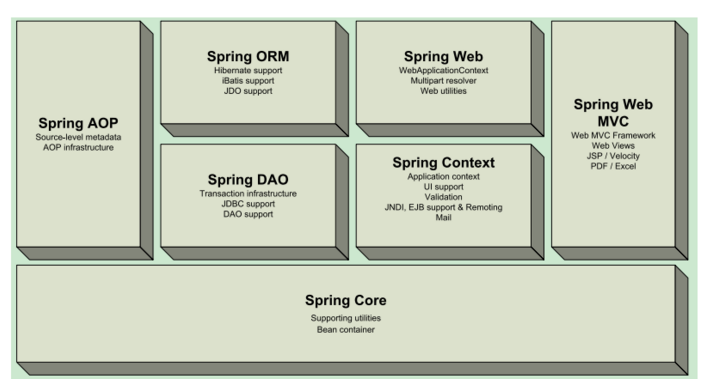


### 4.3.7 PROPAGATION\_NESTED

PROPAGATION\_NESTED：如果当前存在事务，则创建一个该事务的嵌套事务（也叫子事务，嵌套事务的特点：只有外部事务提交了，嵌套事务才能提交；嵌套事务异常回滚，外部事务不需要回滚）；如果当前不存在事务，则创建一个事务。

# 5 Spring模块

图1



Core：最基础部分，提供依赖注入管理bean。

Context：创建上下文，增加了国际化等功能。

Dao：封装了JDBC，消除了JDBC繁琐的代码。

ORM：继承了关系/对象API，包括hibernate，ibatis等。

AOP：面向切面编程。

Web：与其他web框架集成需要使用，比如Struts2。

MVC：提供了Model-View-Controller的分离模型。

# 6 Spring注解

## 6.1 @RequestParam，@RequestBody，@ResponseBody的区别

1. @RequestBody一般处理contentType是application/json或application/xml，将请求体中的数据绑定到参数中的字符串或bean上；@RequestParam一般处理contentType是application/x-www-form-urlcoded（contentType的默认值）。
2. @ResponseBody一般用来返回json或xml数据，使用后直接将json或xml数据写入输出流中，不会走视图处理器。

# 7 FactoryBean

FactoryBean是一个bean，可以用来生产其他类型的bean。

## 7.1 FactoryBean和BeanFactory的区别

1.BeanFactory是一个Bean的工厂，在Spring中，所有的bean都是通过BeanFactory管理的；FactoryBean是一个Java Bean，但它可以用来创建其他类型的bean。

图1

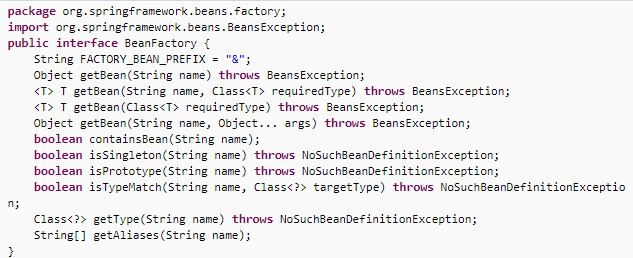


图2

