关于spring中的aop的解释 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/182/2021_2022__E5_85_B3_E4_BA_8Espri_c104_182512.htm AOP是OOP的延续，是Aspect Oriented Programming的缩写，意思是面向方面编程。AOP实际是GoF设计模式的延续，设计模式孜孜不倦追求的是调用者和被调用者之间的解耦，AOP可以说也是这种目标的一种实现。 举例：假设有在一个应用系统中，有一个共享的数据必须被并发同时访问，首先，将这个数据封装在数据对象中，称为Data Class，同时，将有多个访问类，专门用于在同一时刻访问这同一个数据对象。 为了完成上述并发访问同一资源的功能，需要引入锁Lock的概念，也就是说，某个时刻，当有一个访问类访问这个数据对象时，这个数据对象必须上锁Locked，用完后就立即解锁unLocked，再供其它访问类访问。 使用传统的编程习惯，我们会创建一个抽象类，所有的访问类继承这个抽象父类，如下：abstract class Worker{ abstract void locked(). abstract void accessDataObject(). abstract void unlocked().} 缺点：* accessDataObject()方法需要有“锁”状态之类的相关代码。 * Java只提供了单继承，因此具体访问类只能继承这个父类，如果具体访问类还要继承其它父类，比如另外一个如Worker的父类，将无法方便实现。 * 重用被打折扣，具体访问类因为也包含“锁”状态之类的相关代码，只能被重用在相关有“锁”的场合，重用范围很窄。 仔细研究这个应用的“锁”，它其实有下列特性：*“锁”功能不是具体访问类的首要或主要功能，访问类主要功能是访问数据对象，例如读取数据或更改动作。“锁”行为其实是和

具体访问类的主要功能可以独立、区分开来的“锁”功能其实是这个系统的一个纵向切面，涉及许多类、许多类的方法。因此，一个新的程序结构应该是关注系统的纵向切面，例如这个应用的“锁”功能，这个新的程序结构就是aspect（方面）在这个应用中，“锁”方面（aspect）应该有以下职责：提供一些必备的功能，对被访问对象实现加锁或解锁功能。以保证所有在修改数据对象的操作之前能够调用lock()加锁，在它使用完成后，调用unlock()解锁。AOP应用范围 很明显，AOP非常适合开发J2EE容器服务器，目前JBoss 4.0正是使用AOP框架进行开发。具体功能如下：Authentication 权限Caching 缓存Context passing 内容传递Error handling 错误处理Lazy loading 懒加载Debugging 调试logging, tracing, profiling and monitoring 记录跟踪 优化 校准 Performance optimization 性能优化Persistence 持久化Resource pooling 资源池Synchronization 同步Transactions 事务AOP有必要吗？当然，上述应用范例在没有使用AOP情况下，也得到了解决，例如JBoss 3.XXX也提供了上述应用功能，但是没有使用AOP。但是，使用AOP可以让我们从一个更高的抽象概念来理解软件系统，AOP也许提供一种有价值的工具。可以这么说：因为使用AOP结构，现在JBoss 4.0的源码要比JBoss 3.X容易理解多了，这对于一个大型复杂系统来说是非常重要的。从另外一个方面说，好像不是所有的人都需要关心AOP，它可能是一种架构设计的选择，如果选择J2EE系统，AOP关注的上述通用方面都已经被J2EE容器实现了，J2EE应用系统开发者可能需要更多地关注行业应用方面aspect。