用代码学习Spring：IoC、AOP 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__E7_94_A8_E4_BB_A3_E7_A0_81_E5_c104_182515.htm 1 从http://www.springframework.org下载Spring2 用eclipse新建Java项目3 建立我们的业务方法接口public interface BusinessObject { public void doSomething(). public void doAnotherThing().}import org.apache.commons.logging.Log.import org.apache.commons.logging.LogFactory.public interface BusinessObject { public void doSomething(). public void doAnotherThing().}import org.apache.commons.logging.Log.import org.apache.commons.logging.LogFactory.4 实现业务方法，注意这是的setWords使用了依赖注入，所谓依赖注入就是把配置文件中的字符串什么的在程序运行时“ 自动 ”放到我们的程序中来。如果不是这样，我们就只能在代码中固化这些东西，从而违背了面向对象的依赖倒置原则，还有一种满足依赖倒置的方法，即依赖查询，这就是所谓的factory模式，即在代码中请求某种抽象的东西，然后根据配置得到它，但这种办法向对于依赖注入多了对环境的依赖，且代码冗余，EJB的JNDI查询就属于这种。另外我们的Spring配置文件是以bean为核心的，就是我们写的一个类，在XML中描述它的名称、位置和涵盖的内容、关系。public class BusinessObjectImpl implements BusinessObject { private String words. public void setWords(String words){ this.words = words. } public void doSomething() { Log log = LogFactory.getLog(this.getClass()).

log.info(words). } public void doAnotherThing() { Log log = LogFactory.getLog(this.getClass()). log.info("Another thing"). }}public class BusinessObjectImpl implements BusinessObject { private String words. public void setWords(String words){ this.words = words. } public void doSomething() { Log log = LogFactory.getLog(this.getClass()). log.info(words). } public void doAnotherThing() { Log log = LogFactory.getLog(this.getClass()). log.info("Another thing"). }}5 建立一个运行方法类，从配置文件spring-beans.xml中读入bo这个类的定义，然后实例化一个对象import org.springframework.beans.factory.xml.XmlBeanFactory.import org.springframework.core.io.ClassPathResource.public class Main { public static void main(String[] args){ XmlBeanFactory xbf = new XmlBeanFactory(new ClassPathResource("spring-beans.xml")). BusinessObject bo = (BusinessObject)xbf.getBean("bo"). bo.doSomething(). bo.doAnotherThing(). }}import org.springframework.beans.factory.xml.XmlBeanFactory.import org.springframework.core.io.ClassPathResource.public class Main { public static void main(String[] args){ XmlBeanFactory xbf = new XmlBeanFactory(new ClassPathResource("spring-beans.xml")). BusinessObject bo = (BusinessObject)xbf.getBean("bo"). bo.doSomething(). bo.doAnotherThing(). }}6 建立一个拦截器类invoke是MethodInterceptor必须实现的方法，表示拦截时的动作，大家仔细体会代码中的含义import org.aopalliance.intercept.MethodInterceptor.import org.aopalliance.intercept.MethodInvocation.import

org.apache.commons.logging.Log.import org.apache.commons.logging.LogFactory.public class MyInterceptor implements MethodInterceptor { private String before, after. public void setAfter(String after) { this.after = after. } public void setBefore(String before) { this.before = before. } public Object invoke(MethodInvocation invocation) throws Throwable { Log log = LogFactory.getLog(this.getClass()). log.info(before). Object rval = invocation.proceed(). log.info(after). return rval. }}import org.aopalliance.intercept.MethodInterceptor.import org.aopalliance.intercept.MethodInvocation.import org.apache.commons.logging.Log.import org.apache.commons.logging.LogFactory.public class MyInterceptor implements MethodInterceptor { private String before, after. public void setAfter(String after) { this.after = after. } public void setBefore(String before) { this.before = before. } public Object invoke(MethodInvocation invocation) throws Throwable { Log log = LogFactory.getLog(this.getClass()). log.info(before). Object rval = invocation.proceed(). log.info(after). return rval. }}7 建立配置文件组织上面的类之间的关系，AOP有切入点和增强这两个重要的概念，把两个概念结合到一起，就是一个在某个方法执行的时候附加执行，切入点表示在哪里附加，增强表示附加什么，配置文件中的myPointcut表示切入点，myInterceptor表示增强的内容，myAdvisor表示增强器，即两者的结合，在bo这个bean中，我们把这个增强器附加到了bo这个bean上。 正在执行业务方法 执行业务方法前 执行业务方法后 BusinessObject.doSomething BusinessObject

myInterceptor myAdvisor 正在执行业务方法 执行业务方法前 执行业务方法后 BusinessObject.doSomething BusinessObject myInterceptor myAdvisor 8 运行Main类，观察控制台输出结果，重新审查代码，反思为什么会出现这种结果。 