```
<aop:config>
 <aop:pointcut/>
 <aop;advisor/>
 <aop:aspect/>
 //当有多个元素是必须按照该顺序
</aop:config>

```

切面的引入

```
<bean id="aBean" class="切面类的全路径"/>
<aop:config>
 <aop:aspect id="切面类名称" ref="aBean"/>
</aop:config>
```

切入点的引入

```
<aop:config>
 <aop:aspect id="切面类名称" ref="aBean">
 <aop:pointcut id="切入点名称" expression="切入点表达式"/>
 </aop:aspect>
</aop:config>
<bean id="aBean" class="切面类的全路径"/>
```

通知的引入

```
<bean id="aBean" class="切面类的全路径"/>
<aop:config>
 <aop:aspect id="切面类名称" ref="aBean">
 <aop:pointcut id="切入点名称" expression="切入点表达式"/>
 //前置通知引入
 <aop:before method="切面类中的方法名" pointcut-ref="切入点名称"/>
 //返回后的通知
 <aop:after-returning
 pointcout-ref="切入点名称" method="切面类中的方法名"/>
  //异常后的通知
  <aop:after-throwing
  pointcout-ref="切入点名称" method="切面类中的方法名"/>
  //后置通知(finally)
  <aop:before method="切面类中的方法名" pointcut-ref="切入点名称"/>

  //环绕通知
  <aop:around
```

```
 pointcout-ref="切入点名称" method="切面类中的方法名"/>
 //带参数的自定义通知
 <aop:around
 pointcout="切入点表达式.方法名(数据类型,数据类型，...) 可以选填{and args(参数名，参数名,...)}" method="切面类中的方法名"/>
 //若想指定接收的参数名，可以将{}中内容加在表达式后
 </aop:aspect>
</aop:config>
```

环绕通知的方法

```
//该方法的第一个参数必须是ProceedingJoinPoint类型的
public Object 环绕通知方法名(ProceedingJoinPoint pjp){
 //在调用方法之前
 System.out.print("这是在方法之前输出的");
 //指的是具体业务方法执行
 Object 对象 = pjp.proceed();
 return 对象
 //在调用方法之后
 System.out.print("这是在方法之后输出的");
}
//带参数的自定义通知方法
public Object 环绕通知方法名(ProceedingJoinPoint pjp,数据类型 参数名,){
 //打印参数
 System.out.print(参数名);
 //在调用方法之前
 System.out.print("这是在方法之前输出的");
 //指的是具体业务方法执行
 Object 对象名 = pjp.proceed();
 return 对象
 //在调用方法之后
 System.out.print("这是在方法之后输出的");
}
```