

```
1 <aop:config>
2   <aop:pointcut/>
3   <aop:advisor/>
4   <aop:aspect/>
5   //当有多个元素是必须按照该顺序
6 </aop:config>
7
```

切面的引入

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

切入点的引入

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

通知的引入

```
1 <bean id="aBean" class="切面类的全路径"/>
2 <aop:config>
3   <aop:aspect id="切面类名称" ref="aBean">
4     <aop:pointcut id="切入点名称" expression="切入点表达式"/>
5     //前置通知引入
6     <aop:before method="切面类中的方法名" pointcut-ref="切入点名称"/>
7     //返回后的通知
8     <aop:after-returning
9       pointcut-ref="切入点名称" method="切面类中的方法名"/>
10    //异常后的通知
11    <aop:after-throwing
12      pointcut-ref="切入点名称" method="切面类中的方法名"/>
13    //后置通知(finally)
14    <aop:after method="切面类中的方法名" pointcut-ref="切入点名称"/>
15
16    //环绕通知
17    <aop:around
```

```

18 pointcut-ref="切入点名称" method="切面类中的方法名"/>
19 //带参数的自定义通知
20 <aop:around
21 pointcut="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"/>
22 //若想指定接收的参数名, 可以将{}中内容加在表达式后
23 </aop:aspect>
24 </aop:config>

```

环绕通知的方法

```

1 //该方法的第一个参数必须是ProceedingJoinPoint类型的
2 public Object 环绕通知方法名(ProceedingJoinPoint pjp){
3 //在调用方法之前
4 System.out.print("这是在方法之前输出的");
5 //指的是具体业务方法执行
6 Object 对象 = pjp.proceed();
7 return 对象
8 //在调用方法之后
9 System.out.print("这是在方法之后输出的");
10 }
11 //带参数的自定义通知方法
12 public Object 环绕通知方法名(ProceedingJoinPoint pjp,数据类型 参数名,){
13 //打印参数
14 System.out.print(参数名);
15 //在调用方法之前
16 System.out.print("这是在方法之前输出的");
17 //指的是具体业务方法执行
18 Object 对象名 = pjp.proceed();
19 return 对象
20 //在调用方法之后
21 System.out.print("这是在方法之后输出的");
22 }

```