SpringAOP失效的原因

在spring中使用@Transactional、@Cacheable或者自定义的AOP注解时，会发现几个问题：

> *1.在对象内部的方法中调用该对象的其他使用AOP机制的方法，被调用方法的AOP注解失效*

```
public class TicketService{
   //买火车票
   @Transactional
   public void buyTrainTicket(Ticket ticket){
      System.out.println("买到了火车票");
      try {
   //在同一个类中的方法，调用 aop注解（@Transactional 注解也是aop 注解） 的方法，会使aop 注解失效.
      //此时如果 sendMessage()的发送消息动作失败抛出异常，“消息存入数据库“动作不会回滚。
         sendMessage();
      } catch (Exception  e) {
         logger.warn("发送消息异常");
      }
   }

   //买到车票后发送消息
   @Transactional
   public void sendMessage(){
      System.out.println("消息存入数据库");
      System.out.println("执行发送消息动作");
   }
}
```

```
//使用缓存，查询时先查询缓存，缓存中查询不到时，调用数据库。
@Cacheable(value = "User")
public User getUserById(Integer id){
      System.out.println("查询数据库");
   return UserDao.getUserById(id);
}

//在同一个类中的方法，调用 aop注解（@Cacheable 注解也是aop 注解） 的方法，会使aop 注解失效
public User getUser(Integer id){
   //此时注解失效，getUserById 方法不会去缓存中查询数据，会直接查询数据库。
   return getUserById(id);
}
```

原因：

Spring AOP使用java动态代理和Cglib代理来创建AOP代理，没有接口的类使用cglib代理。

aop里面主要通过m.invoke(target,args);来实现调用原来的方法。

我们知道当方法被代理时，其实是动态生成了一个代理对象，代理对象去执行invoke方法，在调用被代理对象的方法的时候执行了一些其他的动作。

所以在被代理对象的方法中调用被代理对象的其他方法的时候。其实是没有用代理调用，使用了被代理对象的本身去掉用的。

解决方法：

```
//通过AopContext.currentProxy()获取当前代理对象。
AopContext.currentProxy();
```

修改范例

修改XML 新增如下语句；先开启cglib代理,开启 exposeProxy = true,暴露代理对象

```
<aop:aspectj-autoproxy proxy-target-class="true" expose-proxy="true"/>
```