AOP的原理就是java的动态代理机制

InvocationHandler 调用处理器

|  |
| --- |
| 每一个动态代理类都必须要实现InvocationHandler这个接口，并且每个代理类的实例都关联到了一个handler，当我们通过代理对象调用 一个方法的时候，这个方法的调用就会被转发为由InvocationHandler这个接口的 invoke 方法来进行调用。我们来看看InvocationHandler这个接口的唯一一个方法 invoke 方法： |

|  |
| --- |
|  |

|  |
| --- |
| Object invoke(Object proxy, Method method, Object[] args) throws Throwable  **proxy**:**指代我们所代理的那个真实对象**  **method**:**指代的是我们所要调用真实对象的某个方法的Method对象**  **args**:**指代的是调用真实对象某个方法时接受的参数** |

|  |
| --- |
| public static Object newProxyInstance(ClassLoader loader, Class<?>[] interfaces, InvocationHandler h) throws IllegalArgumentException  **loader**:　　**一个ClassLoader对象，定义了由哪个ClassLoader对象来对生成的代理对象进行加载**  **interfaces**:　　**一个Interface对象的数组，表示的是我将要给我需要代理的对象提供一组什么接口，如果我提供了一组接口给它，那么这个代理对象就宣称实现了该接口(多态)，这样我就能调用这组接口中的方法了**  **h**:　　**一个InvocationHandler对象，表示的是当我这个动态代理对象在调用方法的时候，会关联到哪一个InvocationHandler对象上** |