## 反射是否破坏了封装？

一般情况下，我们并不能对类的私有字段进行操作，利用反射也不例外，但有的时候，例如要序列化的时候，我们又必须有能力去处理这些字段，这时候，我们就需要调用AccessibleObject上的setAccessible()方法来允许这种访问，而由于反射类中的Field，Method和Constructor继承自AccessibleObject，因此，通过在这些类上调用setAccessible()方法，我们可以实现对这些字段的操作。但有的时候这将会成为一个安全隐患，为此，我们可以启用java.security.manager来判断程序是否具有调用setAccessible()的权限。默认情况下，内核API和扩展目录的代码具有该权限，而类路径或通过URLClassLoader加载的应用程序不拥有此权限

给你孩武不是让你用的。。。。。。而是用来唬人的

你把钥匙丢家里了，你就得用暴力一点的手段去获得，这就是反射的意义！！

debug的时候，需要控制流程中的变量这个时候私有的属性

就是这样修改的吧

存在就说明他们考虑过这个东西，而且说不定以后根据这个特性又出现新的技术也说不定呢

这是一个很值得探讨的问题，许多人接触反射时，对反射功能之强大都会抱有怀疑，感觉严重破坏了封装的性质。可是，什么是封装，什么是安全呢？

封装，是将具体的实现细节隐藏，而把功能作为整体提供给类的外部使用，也就是说，公有方法能够完成类所具有的功能。当别人使用这个类时，如果通过反射直接调用私有方法，可能根本实现不了类的功能，甚至可能会出错，因此通过反射调用私有方法可以说是没有任何用处的，开发人员没有必要故意去破坏封装好的类。从这点上看，封装性并没有被破坏。

所谓安全，如果意思是保护实现源码不被别人看见，那没有作用。不用反射也能轻易获取源码。

所以我以为反射机制只是提供了一种强大的功能，使得开发者能在封装之外，按照特定的需要实现一些功能。就好比核技术，虽然造核弹很危险吧，但造核电站还是很有用处的（这个比喻似乎不是很恰当，将就将就）。

封装，是将具体的实现细节隐藏，而把功能作为整体提供给类的外部使用，也就是说，公有方法能够完成类所具有的功能。当别人使用这个类时，如果通过反射直接调用私有方法，可能根本实现不了类的功能，甚至可能会出错，因此通过反射调用私有方法可以说是没有任何用处的，开发人员没有必要故意去破坏封装好的类。从这点上看，封装性并没有被破坏。

封装性是指对外隐藏对象的属性和实现细节，仅对外提供公共的访问方式。反射是通过对象找到类，既然找到类了，那么我们就可以得到这个类的成员结构了，例如这个类的属性和方法，即使是private的也能得到，你想，现在这个类我都得到了，那么这个类中的所以东西我肯定是都得到了，我现在只是得到了这个类的成员，并没有说是在外部访问这个类的private的东西。这并没有破坏面向对象的封装性。