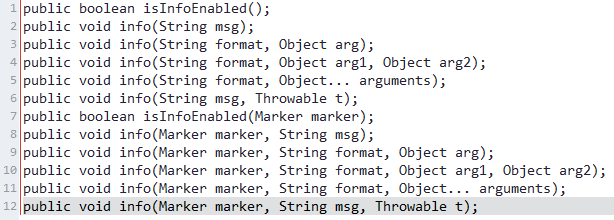
1. 如何使用
2. 使用LoggerFactory里提供的如下两个函数即可获取到Logger接口的实例：





1. 获取到上述Logger接口实例后即可进行log打印操作了，Logger接口定义了5个级别的log打印函数：trace、debug、info、warn、error。每个打印函数都有8个不同参数的多态函数，以info为例：



具体的打印逻辑由具体的logger实现类进行。

1. 分析

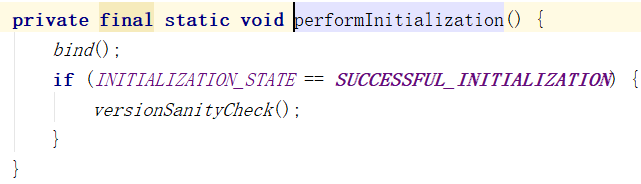
调用getILoggerFactory()函数的逻辑：

1. 执行初始化操作，这个初始化操作主要是查找**org.slf4j.impl.StaticLoggerBinder.class**类；
2. 如果初始化成功，则通过**StaticLoggerBinder**类获取到实际的LoggerFactory类；

分析一下调用getILoggerFactory()函数的时候的过程：



而performInitialization()函数主要做了两件事，第一是执行绑定操作，第二就是在绑定操作完成后执行版本校验操作：



绑定操作的具体逻辑如下：

1. 通过LoggerFactory的classLoader来查找**org.slf4j.impl.StaticLoggerBinder.class**类，如果找到了多个该类则抛出错误，否则查找步骤通过。另外如果是Android平台则会跳过此步查找。
2. 调用StaticLoggerBinder.*getSingleton*();这个调用能够帮助具体实现完成一些初始化工作。实际上在第1步的时候有人会有个疑问，如果找不到**org.slf4j.impl.StaticLoggerBinder.class**类呢，实际上这一步就会抛出异常，因为找不到StaticLoggerBinder类，实际上这个异常会被捕获，然后会将初始化的结果置位NOP\_FALLBACK\_INITIALIZATION 。如果这一步正常执行完毕，就会将初始化的结果置为SUCCESSFUL\_INITIALIZATION

3、执行下面两个函数：

SubstituteLogger*fixSubstituteLoggers*();  
*replayEvents*();

前者主要是对代理Logger工厂（SubstituteLoggerFactory）的代理Logger设置需要代理的对象。后者则是对SubstituteLoggerFactory 持有的LinkedBlockingQueue<SubstituteLoggingEvent> eventQueue进行对应操作。

上面的解释可能有点抽象，这两个操作都涉及SubstituteLoggerFactory类，后面再对SubstituteLoggerFactory类详述。