# 日志

## 日志框架

在Java中，输出日志需要使用一个或者多个日志框架，这些框架提供了必要的对象、方法和配置来传输消息。Java在java.util.logging包中提供了一个默认的框架。除此之外，还有很多其它第三方框架，包括Log4j、Logback以及tinylog。还有其它一些开发包，例如SLF4J和Apache Commons Logging，它们提供了一些抽象层，对你的代码和日志框架进行解耦，从而允许你在不同的日志框架中进行切换。

如何选择一个日志解决方案，这取决于你的日志需求的复杂度、和其它日志解决方案的兼容性、易用性以及个人喜好。Logback基于Log4j之前的版本开发（版本1），因此它们的功能集合都非常类似。然而，Log4j在最新版本（版本2）中引用了一些改进，例如支持多API，并提升了在用Disruptor库的性能。而tinylog，由于缺少了一些功能，运行特别快，非常适合小项目。

另外一个考虑因素是框架在基于Java的各种不同项目上的支持程度。例如Android程序只能使用Log4j、Logback或者第三方包来记录日志， Apache Tomcat可以使用Log4j来记录内部消息，但只能使用版本1的Log4j。

## 抽象层

诸如SLF4J这样的抽象层，会将你的应用程序从日志框架中解耦。应用程序可以在运行时选择绑定到一个特定的日志框架（例如java.util.logging、Log4j或者Logback），这通过在应用程序的类路径中添加对应的日志框架来实现。如果在类路径中配置的日志框架不可用，抽象层就会立刻取消调用日志的相应逻辑。抽象层可以让我们更加容易地改变项目现有的日志框架，或者集成那些使用了不同日志框架的项目。

## 配置

尽管所有的Java日志框架都可以通过代码进行配置，但是大部分配置还是通过外部配置文件完成的。这些文件决定了日志消息在何时通过什么方式进行处理，日志框架可以在运行时加载这些文件。

## 重要性

对于任何一个成熟的系统来说，日志的作用不仅仅来用记录系统操作，日常运行，也为系统监控、系统异常分析、问题定位等提供了准确的依据。而在Java项目（也包括其他系统和应用）中，上述提到的是日志的生产，在实际操作中，日志文件是非常庞大的，靠人工进行分析和处理是很难的，所以需要采用其他的方式去做。

# 分析和监控

常用的日志分析和监控，我们一般都是采用第三方（相对于生产日志的系统）系统，因为日志分析和监控属于系统内部管理的一个环节，无需对外，且与系统避免耦合，所以采用第三方系统。

## 常见的分析和监控

商业工具：

这里一般都是Saas工具，比如国内包括：日志易、（阿里、腾讯、百度等）日志分析。

国外：算了不提了。

开源工具：logstash（就是我们平时提到的ELK工具）、takipi for logs。

当然对于分布式的系统中，可以采用flume等大数据日志监控与分析框架

一般来说对于开发周期短的项目可以使用商业工具，对于已经成熟的项目可以使用开源的框架。

现在使用比较多的是ELK工具。参考“ELK工具教程.doc”