Java 日志的 10 个小细节

Java 日志或者说用 JAVA 记录日志是一门学问也是一门艺术。知道日志记录工具和 API 是一门技术，但是选择日志格式，消息格式，记录内容，日志级别更像来自经验的总结。因为 Java 日志会非常影响性能，我见证了线上股票交易系统因为 DEBUG 模式比 WARN 或者更高的日志级别带来的延迟，在线交易系统中，延迟和响应速度是非常重要的因素，尤其是对于股票交易系统。正因为此，系统的了解 Java 日志是非常有必要的，不仅仅是对于金融和银行投资系统，而且对于任何的追求响应速度和完备的日志的 java 服务器或者客户端应用。

在这一篇 Java 日志总览中，我会分享我在 Java 日志方面的经验，我会回答一些基础的问题“为什么需要 Java 日志”，“不同的 Java 日志级别以及如何选择正确的日志级别”，“不正确的 Java 日志如何影响了性能”。 我会讨论一些 Java 日志工具和 API， 例如 log4j.jar 和 Java.util.logging.

为什么需要 JAVA 日志

这是一个很基础的关于 Java 日志的问题，大家都说反正已经有 System.out.println()来打印记录了，为什么还需要 logging。我们都是从 System.out.println()打印消息(”Hello World!”)到终端上，开始的 Java 学习之旅。但是这个远不能和功能强大的 Java 日志 API（比如 log4j 和 java.util.logging）相比。如果你开发一个 Java 服务器端应用，如果你不记录任何运行信息，你就不可能知道你的服务器正在干什么。这对于需要很多上传信息和下载信息（信息交换）的应用来说（例如股票交易和在线交易系统），记录日志显得更为重要。如果不记录日志，你压根不知道哪里出了问题。这就是为什么 Java 日志服务在 Java 服务端应用中很重要，而且是必须的。

Java 日志的不同级别

用过 Java 日志的一定知道 Java 日志级别， 例如 DEBUG, INFO, WARN 和 ERROR.

DEBUG 是最低限制级的日志级别，并且我们需要记录所有的日志去调试代码，这个模式只能在开发和测试环境中使用，不能在生产环境中使用。

INFO 是比 DEGUG 更限制的日志级别，我们只应该记录一些重要的信息内容，例如服务器启动，有请求，输入输出数据等等。

WARM 是比 INFO 更限制的日志级别，主要是用来记录警告信息，包括服务器和客户端断开连接，数据库断开连接，socket 到达限制。 这些信息是很有用的，你可以在此基础上监控你的服务器运行状态并对这些警告信息做出反应。

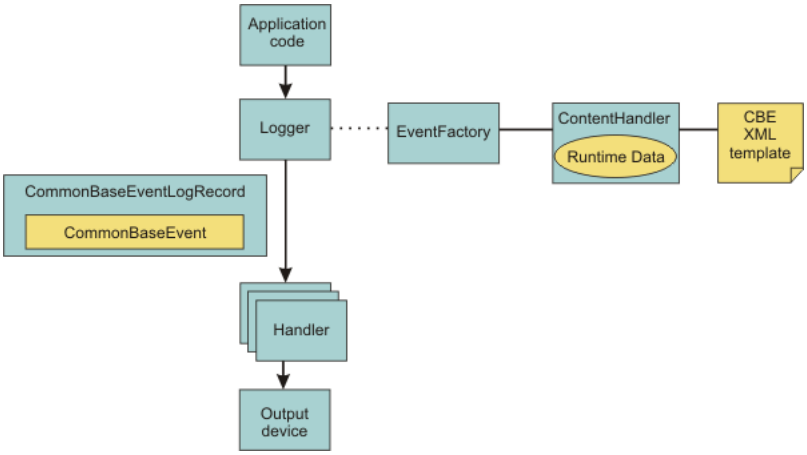
ERROR 是比 WARN 更限制的日志级别，它是用来记录错误和 Exception 的，你可以 alert（弹出）这些错误信息，你必须牢记打印出 ERROR 的错误信息。

FATAL 是用来记录很严重的错误信息，这些错误一般会导致服务器奔溃或者宕机。

OFF 级别代表了要关闭 Java Logging 功能。

这些日志级别是定义在 log4j 包的基础上，跟 java.util.logging API 区别也不大，java.util.logging API 还额外提供例如 SEVERE,FINER,FINEST,FATAL 等其他级别。上述的那些级别是最常用的。

使用 log4j 或者 java.util.loggin API 来做日志记录



我会推荐使用 log4j 来做 java 日志记录，你可能会有疑问，为什么不使用 java.util.logging. 我同意 java.util.logging 功能非常强大，但是 log4j 更直观和易于使用，你已经看到了基于 log4j 的日志级别非常的清晰，另外在 log4j 中你不需要为了改变日志级别而重启服务，当然你用 java.util.logging 也可以做到，前提是你已经使用 JMX 实现这个功能了。

Log4j 可以在 log4j.xml 的基础上轻松的设置每个类的日志级别，你可以选择使用 log4j.properties file 或者 log4j.xml 来配置日志信息，而且 log4j 是线程安全的，log4j 是被设计用于繁忙的多线程系统中的，在另一方面，我发现 java.util.logging 中的 Formatter 和 Appender 非常有用，因为它可以格式化你要打印的东西。

日志记录怎么影响 java 性能的

Java 日志服务对你的应用性能能产生很严重的影响，你打印的日志越多，IO 消耗就越多，对你的性能影响就越大。这就是为什么选择正确的日志级别是很重要的。你要做的就是控制日志的级别和信息。所以把 DEBUG 信息放在 isDebugEnabled()块内，下面的代码就是一个很好的例子。

```
if(logger.isDebugEnabled()){
  logger.debug("java logging level is DEBUG Enabled");
}
```

在正式环境中使用 WARN ERROR 或者 FINER, FINEST 这样的日志级别，绝对不要使用 DEBUG 在正式环境中，我见过一些应用运行起来非常慢的原因就是因为有 DEBUG 级别的日志在打印。

在 JAVA 日志服务中的 10 个小细节

1) 使用 isDebugEnabled()将来输出 DEBUG 日志，它会节省很多的字符串连接操作当你转换到正式环境中。{这里是因为正式环境中，都不需要执行这段代码了，所以不会有日志输出，当然节省了很多的字符串连接操作}。

2) 仔细的选择你的日志级别和打印日志的位置，日志不宜过多，因为会影响性能，也不宜过少，因为过少的日志信息很难让你定位到错误位置和你需要的关键信息。

3) 建议使用 log4j.xml, 因为我用过它很多次，我发现它很灵活，它能让你改变日志级别而不需要重启服务，这在环境转换中是非常有用的(从调试环境到正式环境)，它是通过 log4j watchdog（看门狗）程序一直在监视 log4j.xml,如果找到它就加载然后重新设置日志级别。

4) 通过使用 log4j.xml,你可以对不同的 java 类设置不同的日志级别，这是非常灵活的。

5) 另一个重要的点是，记住日志格式，在日志中最好包括进程名字和类名，方法名，这会帮你很方便的定位。

6) 仔细选择日志格式，你可以通过它生成报表。Balabala 一堆。

7) 在你生成日志的时候，可以在日志的开头使用一些特殊的标记，例如方便定位到 client side, database side, session side 等，因为接下来你可以在 linux 环境下使用 grep 或者 find 命令很方便的找到相关的信息。例如你可以加一个标记“DB_LOG”针对于有关数据库的日志，“SESSION_LOG”针对有关于缓存的信息。

8) 假如你没用定义一个日志级别，那么它会自动的是使用离他最近的一个锚，这就是为什么我们经常定义 root 日志级别为 log4j.rootLogger = DEBUG.

9) 没有日志和过多的日志都是坏主意，所以仔细选择你想要打印的内容并选好日志级别，这样能让你在正式环境中快速的运行应用，同时让你在开发环境中很快的找到问题。

10) 优化你的日志是很重要的，日志要保持简单和明了，不单单是为了自己，也为了方便别人查看和更改。

11) 如果你是使用 SJFS 来记录 java 日志的，那么使用参数模式可以让你性能变得更快，

```
logger.debug("No of Orders " + noOfOrder + " for client : " + client); // slower

logger.debug("No of Executions {} for clients:{}", noOfOrder , client); // faster
```

这里作者也不知道为什么，其实是因为第二种模式少了字符串拼接过程，所以变的更快了。后面的链接中有给出，大家可以看看。

更新：

记录日志的好坏可以用来区分好的程序员和伟大的程序员。不仅仅是对于 JAVA,对于其他编程语言也是一样的。你需要关注的是：

1）你需要打印什么样的信息？

2）这个信息是属于哪个日志级别的？

这是两个最重要的问题，下面是我编程的一些宝贵经验：

1）不要打印敏感信息，包括密码，社会人口号，信用卡账号等等，在你完成这部分开发的时候就应该把这些可能打印出来的关键信息删除了。

2）打印一些决策性的信息，比如你在 java 中加载一些配置信息，如果没找到就加载默认选项，那你可以打印以下内容：

logger.info("Not able to load personal settings, default Setting selected for user : {user});

这里就包括决策性信息，而且也包括了哪个用户信息你没法打印，因为你可能拥有很多个用户。

3）保持一致性，日志格式需要保持一致。

4）打印关键信息，尤其是你在调试的时候。

我们经常用到日期转换成字符串的操作，这个方法经常会抛出一些异常，以下就是一个日志，这个打印信息包含的内容不全，它没有告诉我们到底是什么问题，是输入的日期格式不对，日期空白，还是什么。

logger.info("failed to convert String to date");

这个打印日志语句就好了很多，它告诉了我们输入的到底是什么东西，我们可以很容易的定位出问题位置和问题原因。

logger.info("invalid startDate : {startDate});

感谢你阅读本篇文章，如果你觉得不错，欢迎分享给你的朋友。

——

最后面是我找到的几篇 JAVA 日志级别的文章

http://developer.51cto.com/art/201507/484646.htm Java 日志终极指南

http://www.ibm.com/developerworks/cn/java/j-lo-practicelog/index.html#ibm-pcon Java 日志管理最佳实践

http://www.oschina.net/translate/why-use-sl4j-over-log4j-for-logging?cmp 为什么使用 SLF4J 而不是 Log4J 来做 Java 日志（这是原作者写的关于 java 日志的另一篇文章）