Логгирование в java

System.out.println()

Log4J

java.util.logging

Apache Commons Logging

Simple Logging Facade for Java

Logback

Краткое резюме

Apache Log4J – хороший фреймворк для логирования, практически лишенный недостатков. Широко используется. Разработка находится, фактически, в замороженном5 состоянии, производится только исправление ошибок.

java.util.logging – фреймворк, являющийся частью JavaSE. По возможностям уступает Log4J. Тем не менее используется хотя бы потому, что всегда под рукой и не требует дополнительных библиотек.

Apache Commons Logging – фреймворк, предназначенный для абстрагирования конкретного фреймворка ("под" ним может работать как Log4J, так и java.util.logging, а также несколько других). Имеет определенные проблемы с загрузчиком классов, что в определенных ситуациях затрудняет его использование.

SLF4J – еще один абстрагирующий фреймворк, существенно более удачный, чем Commons Logging. Может работать в двух ипостасях – как общий интерфейс к лежащим ниже фреймворкам и как приемник соответствующего типа для фреймворков, расположенных "над" ним.

Logback – молодой, но весьма интересный фреймворк, выросший из Log4J. Взял от родителя все преимущества, плюс еще добавил своих. Возможно, в будущем станет даже более привлекательным, чем Log4J.

Использование Log4J

Логгер представляет собой объект класса org.apache.log4j.Logger, который используется для вывода данных и управления уровнем (детализацией) вывода. В текущей версии – 1.2.16 – Log4J поддерживает следующие уровни вывода, в порядке возрастания:

TRACE

DEBUG

INFO

WARN

ERROR

FATAL

OFF

Аппендер

- Консоль
- Файлы (несколько различных типов)
- JDBC
- Темы (topics) JMS
- NT Event Log
- SMTP
- Сокет
- Syslog
- Telnet
- Любой java.io.Writer или java.io.OutputStream

Файловые аппендеры

org.apache.log4j.FileAppender

org.apache.log4j.RollingFileAppender

org.apache.log4j.DailyRollingFileAppender

Компоновка

org.apache.log4j.SimpleLayout

org.apache.log4j.HTMLLayout

org.apache.log4j.xml.XMLLayout

org.apache.log4j.TTCCLayout

org.apache.log4j.PatternLayout /
org.apache.log4j.EnhancedPatternLayout

org.apache.log4j.SimpleLayout

Наиболее простой вариант. На выходе дает уровень вывода и, собственно, сообщение. Т.е. следующий код –

```
logger.info("Some message");
```

– на выходе даст вот так отформатированную строку:

```
INFO - Some message
```

org.apache.log4j.PatternLayout

%d{ISO8601} [%-5p][%-16.16t][%32.32c] - %m%n

11:31:32,342 Thread-1 ERROR audit.LoadTest - Check in 344ms: GlobalID=2

11:31:32,358 Thread-17 WARN ServiceLoadTest - Check in 156ms: GlobalID=8

11:31:32,378 Thread-2 INFO trace.ServiceLoadTrace - Check in 328ms: GlobalID=3

11:31:35,358 Thread-44 DEBUG ext.ServiceParallelLoadTest - Check in 250ms: GlobalID=5

11:31:36,637 Thread-503 INFO ServiceLoadTest - Check in 219ms: GlobalID=6

11:31:37,846 Thread-59 INFO extract.Extractor - Check in 94ms: GlobalID=10

11:31:39,072 Thread-86 DEBUG ServiceLoadTest - Check in 188ms: GlobalID=7

11:31:41,309 Thread-10 INFO back.BackLoaderInfo - Check in 47ms: GlobalID=11

Конфигурирование

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration debug="false" xmlns:log4j="http://jakarta.apache.org/log4j/">
   <appender name="ConsoleAppender" class="org.apache.log4j.ConsoleAppender">
      <param name="Encoding" value="Cp866"/>
      <layout class="org.apache.log4j.PatternLayout">
         <param name="ConversionPattern" value="%d{ISO8601}
     [%-5p][%-16.16t][%32.32c] - %m%n" />
      </layout>
   </appender>
   <root>
      <priority value="ERROR"/>
      <appender-ref ref="ConsoleAppender" />
   </root>
</log4j:configuration>
```

Конфигурирование

```
log4j.debug = false
log4j.rootLogger = ERROR, ConsoleAppender
log4j.appender.ConsoleAppender = org.apache.log4j.ConsoleAppender
log4j.appender.ConsoleAppender.encoding = Cp866
log4j.appender.ConsoleAppender.layout = org.apache.log4j.PatternLayout
log4j.appender.ConsoleAppender.layout.ConversionPattern = %d{ISO8601}
    [%-5p][%-16.16t][%32.32c] - %m%n
```

Использование логгеров

```
private static final Logger logger =
    Logger.getLogger(MyClass.class);
private static final Logger dbLogger =
    Logger.getLogger("ru.skipy.DB");

logger.info("Starting mass rate charge
    calculation...");
```

Debug

```
logger.debug("Starting rate charge calculation for
   account " + accountNum + " with parameters: rest=" +
   rest +
            ", percent=" + percent + " period=" + period);


if (logger.isDebugEnabled()){
    logger.debug("Starting rate charge calculation for
   account " + accountNum + " with parameters: rest=" +
   rest +
               ", percent=" + percent + ", period=" + period);
}
```