日志组件 logback 的介绍及配置使用方法

一、logback 的介绍

Logback 是由 log4j 创始人设计的又一个开源日志组件。logback 当前分成三个模块：logback-core,logback- classic 和 logback-access。logback-core 是其它两个模块的基础模块。logback-classic 是 log4j 的一个 改良版本。此外 logback-classic 完整实现 SLF4J API 使你可以很方便地更换成其它日志系统如 log4j 或 JDK14 Logging。logback-access 访问模块与 Servlet 容器集成提供通过 Http 来访问日志的功能。 Logback 是要与 SLF4J 结合起来用两个组件的官方网站如下：

logback 的官方网站： http://logback.qos.ch

SLF4J 的官方网站：http://www.slf4j.org

本文章用到的组件如下：请自行到官方网站下载！

logback-access-1.0.0.jar

logback-classic-1.0.0.jar

logback-core-1.0.0.jar

slf4j-api-1.6.0.jar

二、logback 取代 log4j 的理由：

Logback 和 log4j 是非常相似的，如果你对 log4j 很熟悉，那对 logback 很快就会得心应手。下面列了 logback 相对于 log4j 的一些优点：

1、更快的实现 Logback 的内核重写了，在一些关键执行路径上性能提升10倍以上。而且 logback 不仅性能提升了，初始化内存加载也更小了。

2、非常充分的测试 Logback 经过了几年，数不清小时的测试。Logback 的测试完全不同级别的。在作者的观点，这是简单重要的原因选择 logback 而不是 log4j。

3、Logback-classic 非常自然实现了 SLF4j Logback-classic 实现了 SLF4j。在使用 SLF4j 中，你都感觉不到 logback-classic。而且因为 logback-classic 非常自然地实现了 SLF4J，所 以切换到 log4j 或者其他，非常容易，只需要提供成另一个 jar 包就 OK，根本不需要去动那些通过 SLF4JAPI 实现的代码。

4、非常充分的文档 官方网站有两百多页的文档。

5、自动重新加载配置文件 当配置文件修改了，Logback-classic 能自动重新加载配置文件。扫描过程快且安全，它并不需要另外创建一个扫描线程。这个技术充分保证了应用程序能跑得很欢在 JEE 环境里面。

6、Lilith Lilith 是 log 事件的观察者，和 log4j 的 chainsaw 类似。而 lilith 还能处理大数量的 log 数据 。

7、谨慎的模式和非常友好的恢复 在谨慎模式下，多个 FileAppender 实例跑在多个 JVM 下， 能 够安全地写道同一个日志文件。RollingFileAppender 会有些限制。Logback 的 FileAppender 和它的子类包括 RollingFileAppender 能够非常友好地从 I/O 异常中恢复。

8、配置文件可以处理不同的情况 开发人员经常需要判断不同的 Logback 配置文件在不同的环境下（开发，测试，生产）。而这些配置文件仅仅只有一些很小的不同，可以通过,和来实现，这样一个配置文件就可以适应多个环境。

9、Filters（过滤器） 有些时候，需要诊断一个问题，需要打出日志。在 log4j，只有降低日志级别，不过这样会打出大量的日志，会影响应用性能。在 Logback，你可以继续 保持那个日志级别而除掉某种特殊情况，如 alice 这个用户登录，她的日志将打在 DEBUG 级别而其他用户可以继续打在 WARN 级别。要实现这个功能只需 加4行 XML 配置。可以参

考 MDCFIlter 。

10、SiftingAppender（一个非常多功能的 Appender） 它可以用来分割日志文件根据任何一个给定的运行参数。如，SiftingAppender 能够区别日志事件跟进用户的 Session，然后每个用户会有一个日志文件。

11、自动压缩已经打出来的 log RollingFileAppender 在产生新文件的时候，会自动压缩已经打出来的日志文件。压缩是个异步过程，所以甚至对于大的日志文件，在压缩过程中应用不会受任何影响。

12、堆栈树带有包版本 Logback 在打出堆栈树日志时，会带上包的数据。

13、自动去除旧的日志文件 通过设置 TimeBasedRollingPolicy 或者 SizeAndTimeBasedFNATP 的 maxHistory 属性，你可以控制已经产生日志文件的最大数量。如果设置 maxHistory 12，那那些 log 文件超过12个月的都会被自动移除。

总之，logback 比 log4j 太优秀了，让我们的应用全部建立 logback 上吧 ！

三、Logback 的配置介绍

1、Logger、appender 及 layout

Logger 作为日志的记录器，把它关联到应用的对应的 context 上后，主要用于存放日志对象，也可以定义日志类型、级别。

Appender 主要用于指定日志输出的目的地，目的地可以是控制台、文件、远程套接字服务器、 MySQL、 PostreSQL、 Oracle 和其他数据库、 JMS 和远程 UNIX Syslog 守护进程等。

Layout 负责把事件转换成字符串，格式化的日志信息的输出。

2、logger context

各个 logger 都被关联到一个 LoggerContext，LoggerContext 负责制造 logger，也负责以树结构排列各 logger。其他所有 logger 也通过 org.slf4j.LoggerFactory 类的静态方法 getLogger 取得。getLogger 方法以 logger 名称为参数。用同一名字调用 LoggerFactory.getLogger 方法所得到的永远都是同一个 logger 对象的引用。

3、有效级别及级别的继承

Logger 可以被分配级别。级别包括：TRACE、DEBUG、INFO、WARN 和 ERROR，定义于 ch.qos.logback.classic.Level 类。如果 logger 没有被分配级别，那么它将从有被分配级别的最近的祖先那里继承级别。root logger 默认级别是 DEBUG。

4、打印方法与基本的选择规则

打印方法决定记录请求的级别。例如，如果 L 是一个 logger 实例，那么，语句 L.info("..")是一条级别为 INFO 的记录语句。记录请求的级别在高于或等于其 logger 的有效级别时被称为被启用，否则，称为被禁用。记录请求级别为 p，其 logger 的有效级别为 q，只有则当 p>=q 时，该请求才会被执行。

该规则是 logback 的核心。级别排序为： TRACE < DEBUG < INFO < WARN < ERROR。

四、Logback 的默认配置

如果配置文件 logback-test.xml 和 logback.xml 都不存在，那么 logback 默认地会调用 BasicConfigurator ，创建一个最小化配置。最小化配置由一个关联到根 logger 的 ConsoleAppender 组成。输出用模式为 %d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n 的 PatternLayoutEncoder 进行格式化。root logger 默认级别是 DEBUG。

1、Logback 的配置文件

Logback 配置文件的语法非常灵活。正因为灵活，所以无法用 DTD 或 XML schema

进行定义。尽管如此，可以这样描述配置文件的基本结构：以<configuration>开头，后面有零个或多个<appender>元素，有零个或多个<logger>元素，有最多一个<root>元素。

2、Logback 默认配置的步骤

(1). 尝试在 classpath 下查找文件 logback-test.xml；

(2). 如果文件不存在，则查找文件 logback.xml；

(3). 如果两个文件都不存在，logback 用 Bas icConfigurator 自动对自己进行配置，这会导致记录输出到控制台。

3、Logback.xml 文件

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <!--定义日志文件的存储地址 勿在 LogBack 的配置中使用相对路径-->
    <property name="LOG_HOME" value="c:/log" />
    <!-- 控制台输出 -->
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <!-- 日志输出编码 -->
        <Encoding>UTF-8</Encoding>
         <layout class="ch.qos.logback.classic.PatternLayout">
              <!--格式化输出：%d 表示日期，%thread 表示线程名，%-5level：级别从左显示5个字符宽度%msg：日志消息，%n 是换行符-->
              <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n
              </pattern>
        </layout>
    </appender>
    <!-- 按照每天生成日志文件 -->
    <appender name="FILE"  class="ch.qos.logback.core.rolling.RollingFileAppender">
        <Encoding>UTF-8</Encoding>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!--日志文件输出的文件名-->

<FileNamePattern>${LOG_HOME}/myApp.log.%d{yyyy-MM-dd}.log</FileNamePattern>
            <MaxHistory>30</MaxHistory>
        </rollingPolicy>
        <layout class="ch.qos.logback.classic.PatternLayout">
            <!--格式化输出：%d 表示日期，%thread 表示线程名，%-5level：级别从左显示5个字符宽度%msg：日志消息，%n 是换行符-->
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n
            </pattern>
        </layout>
         <!--日志文件最大的大小-->
```

```
            <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
              <MaxFileSize>10MB</MaxFileSize>
            </triggeringPolicy>
        </appender>
       <!-- show parameters for hibernate sql  专为  Hibernate  定制  -->
        <logger name="org.hibernate.type.descriptor.sql.BasicBinder"    level="TRACE" />
        <logger name="org.hibernate.type.descriptor.sql.BasicExtractor"    level="DEBUG" />
        <logger name="org.hibernate.SQL" level="DEBUG" />
        <logger name="org.hibernate.engine.QueryParameters" level="DEBUG" />
        <logger name="org.hibernate.engine.query.HQLQueryPlan" level="DEBUG" />

        <!--  日志输出级别  -->
        <root level="INFO">
            <appender-ref ref="STDOUT" />
            <appender-ref ref="FILE" />
        </root>

         <!--日志异步到数据库  -->
        <appender name="DB" class="ch.qos.logback.classic.db.DBAppender">
            <!--日志异步到数据库  -->
            <connectionSource class="ch.qos.logback.core.db.DriverManagerConnectionSource">
                <!--连接池  -->
                <dataSource class="com.mchange.v2.c3p0.ComboPooledDataSource">
                    <driverClass>com.mysql.jdbc.Driver</driverClass>
                    <url>jdbc:mysql://127.0.0.1:3306/databaseName</url>
                    <user>root</user>
                    <password>root</password>
                 </dataSource>
            </connectionSource>
    </appender> -->
</configuration>
```

五、在程序用引用 Logback

```
package com.stu.system.action;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class BlogAction{
        //定义一个全局的记录器，通过 LoggerFactory 获取
        private final static Logger logger = LoggerFactory.getLogger(BlogAction.class);
         /**
         * @param args
         */
```

```
    public static void main(String[] args) {
        logger.info("logback  成功了");
        logger.error("logback  成功了");
    }
}
```