# 概念

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。

# 官网

https://logback.qos.ch/

# maven依赖

<!-- logback+slf4j -->

<dependency>

    <groupId>org.slf4j</groupId>

    <artifactId>slf4j-api</artifactId>

    <version>1.7.21</version>

    <type>jar</type>

    <scope>compile</scope>

</dependency>

<dependency>

    <groupId>ch.qos.logback</groupId>

    <artifactId>logback-core</artifactId>

    <version>1.1.7</version>

    <type>jar</type>

</dependency>

<dependency>

    <groupId>ch.qos.logback</groupId>

    <artifactId>logback-classic</artifactId>

    <version>1.1.7</version>

    <type>jar</type>

</dependency>

# 配置和使用

logback的配置文件放在/src/main/resource/文件夹下的logback.xml文件中。其中logback.xml文件就是logback的配置文件。只要将这个文件放置好了之后，系统会自动找到这个配置文件。

## 配置

<?xml version="1.0" encoding="UTF-8"?>

<!--

scan：当此属性设置为true时，配置文件如果发生改变，将会被重新加载，默认值为true。

scanPeriod：设置监测配置文件是否有修改的时间间隔，如果没有给出时间单位，默认单位是毫秒当scan为true时，此属性生效。默认的时间间隔为1分钟。

debug：当此属性设置为true时，将打印出logback内部日志信息，实时查看logback运行状态。默认值为false。

-->

<configuration scan="false" scanPeriod="60 seconds" debug="false">

    <!-- 定义日志的根目录 (此路径会生成eclipse根目录的相对路径，生产环境建议配合环境变量使用绝对路径)-->

    <property name="LOG\_HOME" value="E:\\Tomcat\\logback" />

    <!-- 定义日志文件名称 -->

    <property name="appName" value="ideal\_system"></property>

    <!-- 控制台输出 -->

    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

        <Encoding>UTF-8</Encoding>

        <!--

        日志输出格式：%d表示日期时间，%thread表示线程名，%-5level：级别从左显示5个字符宽度

        %logger{50} 表示logger名字最长50个字符，否则按照句点分割。 %msg：日志消息，%n是换行符

        -->

        <layout class="ch.qos.logback.classic.PatternLayout">

            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>

        </layout>

    </appender>

    <!-- 滚动记录文件，先将日志记录到指定文件，当符合某个条件时，将日志记录到其他文件（按照每天生成日志文件） -->

    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">

        <Encoding>UTF-8</Encoding>

        <!-- 指定日志文件的名称 -->

        <file>${LOG\_HOME}/${appName}/sys.log</file>

        <!--拒绝ERROR日志-->

        <filter class="ch.qos.logback.classic.filter.LevelFilter">

            <level>ERROR</level>

            <onMatch>DENY</onMatch>

            <onMisMatch>NEUTRAL</onMisMatch>

        </filter>

        <filter class="ch.qos.logback.classic.filter.ThresholdFilter">

            <level>DEBUG</level>

            <onMatch>ACCEPT</onMatch>

            <onMismatch>DENY</onMismatch>

        </filter>

        <!--

        当发生滚动时，决定 RollingFileAppender 的行为，涉及文件移动和重命名

        TimeBasedRollingPolicy： 最常用的滚动策略，它根据时间来制定滚动策略，既负责滚动也负责出发滚动。

        -->

        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

            <!--

            滚动时产生的文件的存放位置及文件名称 %d{yyyy-MM-dd}：按天进行日志滚动

            %i：当文件大小超过maxFileSize时，按照i进行文件滚动

            -->

            <fileNamePattern>${LOG\_HOME}/${appName}/sys-%d{yyyy-MM-dd}-%i.log</fileNamePattern>

            <!--

            可选节点，控制保留的归档文件的最大数量，超出数量就删除旧文件。假设设置每天滚动，

            且maxHistory是180，则只保存最近180天的文件，删除之前的旧文件。注意，删除旧文件是，

            那些为了归档而创建的目录也会被删除。

            -->

            <MaxHistory>180</MaxHistory>

            <!--

            当日志文件超过maxFileSize指定的大小时，根据上面提到的%i进行日志文件滚动

            注意此处配置SizeBasedTriggeringPolicy是无法实现按文件大小进行滚动的，必须配置timeBasedFileNamingAndTriggeringPolicy

            -->

            <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">

                <maxFileSize>100MB</maxFileSize>

            </timeBasedFileNamingAndTriggeringPolicy>

        </rollingPolicy>

        <!--

        日志输出格式：%d表示日期时间，%thread表示线程名，%-5level：级别从左显示5个字符宽度

        %logger{50} 表示logger名字最长50个字符，否则按照句点分割。 %msg：日志消息，%n是换行符

        -->

        <layout class="ch.qos.logback.classic.PatternLayout">

            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [ %thread ] - [ %-5level ] [ %logger{50} : %line ] - %msg%n</pattern>

        </layout>

    </appender>

    <appender name="FILE-ERROR" class="ch.qos.logback.core.rolling.RollingFileAppender">

        <Encoding>UTF-8</Encoding>

        <file>${LOG\_HOME}/${appName}/sys-error.log</file>

        <filter class="ch.qos.logback.classic.filter.LevelFilter">

            <level>ERROR</level>

            <onMatch>ACCEPT</onMatch>

            <onMismatch>DENY</onMismatch>

        </filter>

        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

            <fileNamePattern>${LOG\_HOME}/${appName}/sys-error-%d{yyyy-MM-dd}-%i.log</fileNamePattern>

            <MaxHistory>180</MaxHistory>

            <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">

                <maxFileSize>100MB</maxFileSize>

            </timeBasedFileNamingAndTriggeringPolicy>

        </rollingPolicy>

        <layout class="ch.qos.logback.classic.PatternLayout">

            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [ %thread ] - [ %-5level ] [ %logger{50} : %line ] - %msg%n</pattern>

        </layout>

    </appender>

    <!--

    logger主要用于存放日志对象，也可以定义日志类型、级别

    name：表示匹配的logger类型前缀，也就是包的前半部分

    level：要记录的日志级别，包括 TRACE < DEBUG < INFO < WARN < ERROR

    additivity：作用在于children-logger是否使用 rootLogger配置的appender进行输出，

        false：表示只用当前logger的appender-ref，

        true：表示当前logger的appender-ref和rootLogger的appender-ref都有效

    -->

    <!-- mybatis logger -->

    <logger name="org.mybatis" level="error" />

    <!-- Spring framework logger -->

    <logger name="org.springframework" level="error" additivity="false"></logger>

   <!--  <logger name="com.ideal" level="info" additivity="true">

        <appender-ref ref="appLogAppender" />

    </logger> -->

    <!--

    root与logger是父子关系，没有特别定义则默认为root，任何一个类只会和一个logger对应，

    要么是定义的logger，要么是root，判断的关键在于找到这个logger，然后判断这个logger的appender和level。

    -->

    <root level="debug">

        <appender-ref ref="STDOUT"/>

        <appender-ref ref="FILE"/>

        <appender-ref ref="FILE-ERROR"/>

    </root>

</configuration>

## 完整的示例文件



## 使用

