28_代码基石：引入 logback 框架进行日志打印

1、开篇

上周的课我们介绍 AOP IOC 和 AOP 技术，以及其应用原理。本周正式进入实践课程，顺着上周的 Spring AOP 技术来介绍 Logback 实现日志记录的功能。内容包括：

- Logback 概述
- Logback 配置介绍

2、Logback 介绍

Logback 是一个 Java 领域的日志框架。它被认为是 Log4J 的继承人。 Logback 和 Log4j 属于同一个作者，看上去 Logback 是 Log4j 的升级版。Logback 分为三个模块， logback-core， logback-classic 和 logback-access，其中 logback-core 是核心，其他两个模块依赖 core，这个 logback-classic 是 log4j 的改善版本，并且原生实现了 SLF4J 门面。模块 logback-access 可以集成于 Servlet 容器，比如 Tomact 和 Jetty。你可以基于 logback-core 自己创建其他的模块。

如图 1 所示，在项目的包引用中，需要依赖三个 jar 包分别是：slf4j-api，logback-core，logback-classic。

```
<dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.7.5</version>
</dependency>
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-core</artifactId>
    <version>1.0.11</version>
</dependency>
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.0.11</version>
</dependency>
```

图 1　Logback 的依赖引用

3、Logback 的配置介绍

由于 Logback 针对整个系统的日志进行定义，因此需要有一个文件对这些定义进行配置。这就是 Logback 的配置文件，这里我们以 logback.xml 文件为例给大家介绍。

（1）根节点<configuration>

<configuration>元素包含下面三个属性：

scan: 此属性设置为“true”时，配置文件发生改变，将会被重新加载。

scanPeriod: 配置文件的修改间隔，当 scan 为“true”时此属性生效。“60 seconds”表示配置文件每间隔 60 秒检查一次是否被修改。

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="true" scanPeriod="60 seconds" debug="false">
    <!--其他配置省略-->
</configuration>
```

图 2 configuration 节点

（2）子节点<property>

该节点是 configuration 的子节点，用来定义变量值，它有两个属性 name 和 value，通过<property>定义的值会被插入到 logger 上下文中，可以使“${}”来使用变量。

name：变量的名称

value：的值时变量定义的值

```
<configuration scan="true" scanPeriod="60 seconds" debug="false">
      <property name="APP_Name" value="myAppName" />
      <contextName>${APP_Name}</contextName>
      <!--其他配置省略-->
</configuration>
```

图 3 property 节点

（3）子节点<appender>

同样 appender 节点也是 configuration 的子节点，Logback 将写入日志事件的任务委托给一个名为 appender 的组件。它有两个属性 name 和 class。name 指定 appender 名称，class 指定类的全限定名用于实例化。<appender>元素可以包含 0 或一个 <layout> 元素，0 或多个 <encoder> 元素，0 或多个 <filter> 元素。除了这些公共的元素之外，<appender> 元素可以包含任意与 appender 类的 JavaBean 属性相一致的元素。常见的结构如下：

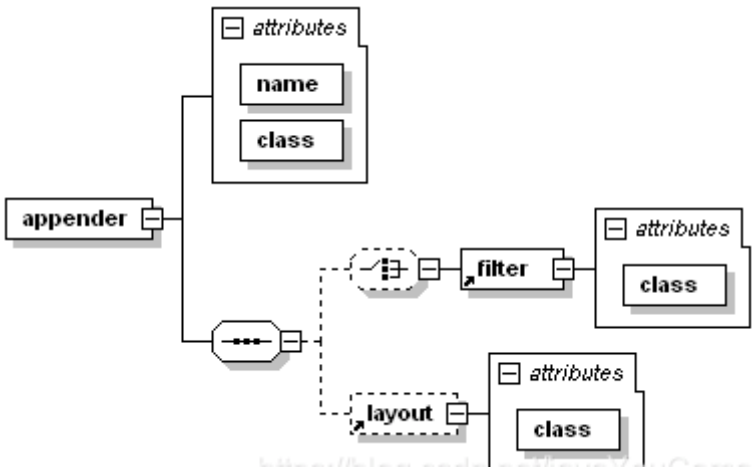

图 3 appender 结构

（4）RollingFileAppender（轮转日志）

RollingFileAppender 是一类 appender 其功能比较强大，因此被广泛应用。它将日志输出到 log.txt 文件，在满足了特定的条件之后，将日志输出到另外一个文件。

与 RollingFileAppender 进行交互的有两个重要的子组件。第一个是 RollingPolicy，它负责日志轮转的记录策略。另一个是 TriggeringPolicy，它负责日志轮转的记录时机。如图 4 所示，

<file>：被写入的文件名，可以是相对目录，也可以是绝对目录，如果上级目录不存在会自动创建，没有默认值。

<append>：如果是 true，日志被追加到文件结尾，如果是 false，清空现存文件，默认是 true。

<encoder>：对记录事件进行格式化。包括两个功能，一是把日志信息转换成字节数组，二是把字节数组写入到输出流。

PatternLayoutEncoder 是唯一有用的且默认的 encoder ，有一个<pattern>节点，用来设置日志的输入格式。使用“%”加“转换符”方式，如果要输出“%”，则必须用“\”对“\%”进行转义。

<prudent>：当为 true 时，不支持 FixedWindowRollingPolicy。支持 TimeBasedRollingPolicy，但是有两个限制，1 不支持也不允许文件压缩，2 不能设置 file 属性，必须留空。

<rollingPolicy>:当发生滚动时，决定 RollingFileAppender 的行为，涉及文件移动和重命名。属性 class 定义具体的滚动策略类。

```xml
<configuration>
    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>logFile.%d{yyyy-MM-dd}.log</fileNamePattern>
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <encoder>
            <pattern>%-4relative [%thread] %-5level %logger{35} - %msg%n</pattern>
        </encoder>
    </appender>

    <root level="DEBUG">
        <appender-ref ref="FILE" />
    </root>
</configuration>
```

图 4 RollingFileAppender 结构

（5）Filter 日志过滤器

Filter 作为 appender 的子节点，如图所示，

<filter>将日志信息过滤：常用 LevelFilter-等级过滤器来过滤日志信息。有三个字元素

<level>INFO</level>：指定特定等级

<onMatch>ACCEPT</onMatch>：如果匹配则接受输出

<onMismatch>DENY</onMismatch>：如果不匹配则过滤掉

```
<configuration>
  <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
    <filter class="ch.qos.logback.classic.filter.LevelFilter">
      <level>INFO</level>
      <onMatch>ACCEPT</onMatch>
      <onMismatch>DENY</onMismatch>
    </filter>
    <encoder>
      <pattern>
        %-4relative [%thread] %-5level %logger{30} - %msg%n
      </pattern>
    </encoder>
  </appender>
  <root level="DEBUG">
    <appender-ref ref="CONSOLE" />
  </root>
</configuration>
```

图 5 filter 日志过滤器

（6）Logger 子节点

其用来设置某一个包或具体的某一个类的日志打印级别、以及指定<appender>。

<logger>仅有一个 name 属性，一个可选的 level 和一个可选的 addtivity 属性。

可以包含零个或多个<appender-ref>元素，标识这个 appender 将会添加到这个 logger

name: 用来指定受此 logger 约束的某一个包或者具体的某一个类。

level: 用来设置打印级别，大小写无关：TRACE, DEBUG, INFO, WARN, ERROR, ALL 和 OFF，还有一个特殊值 INHERITED 或者同义词 NULL，代表强制执行上级的级别。 如果未设置此属性，那么当前 logger 将会继承上级的级别。

addtivity: 是否向上级 logger 传递打印信息。默认是 true。

logger 节点可以包含零个或多个<appender-ref>元素，标识这个 appender 将会添加到这个 logger。

（7）Root 节点

它也是一个 logger 节点元素，不过它是一个根 logger，也就是是所有 logger 的上级节点。只有一个 level 属性，因为 name 已经被命名为"root",且已经是最上级了。

level: 用来设置打印级别，大小写无关：TRACE, DEBUG, INFO, WARN, ERROR, ALL 和 OFF，不能设置为 INHERITED 或者同义词 NULL。 默认是 DEBUG。

```
<!-- 日志输出级别 -->
<logger name="io.netty" level="ERROR"/>
<logger name="org.apache" level="ERROR"/>
<logger name="org.mybatis" level="ERROR"/>
<logger name="org.springframework" level="ERROR"/>
<logger name="org.jboss" level="ERROR"/>
<logger name="javax" level="ERROR"/>
<logger name="com.fasterxml" level="ERROR"/>
<logger name="com.alibaba" level="WARN"/>

<root>
    <appender-ref ref="STDOUT"/>
    <appender-ref ref="FILE"/>
    <appender-ref ref="FILE_DEBUG"/>
    <appender-ref ref="FILE_ERROR"/>
</root>
```

图 6 logger 和 root 节点

如图 7 所示，我们将互联网教育系统的 logback.xml 文件的截图放到这里，供大家参考。后面我们的团队也会提供相应的代码给大家使用。

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration debug="false" scan="true" scanPeriod="30 minutes">
    <statusListener class="ch.qos.logback.core.status.NopStatusListener"/>
    <!-- 引入外部配置文件的地址 -->
    <property resource="logback.properties"/>

    <!-- 控制台输出 -->
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
            <level>${STDOUT_LEVEL}</level>
        </filter>
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <pattern>[%date{yyyy-MM-dd HH:mm:ss.SSS}] %X{logthreadId} %-5level %logger{80} %line - %msg%n</pattern>
        </encoder>
    </appender>

    <!-- INFO级别的日志 -->
    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${LOG_HOME}/${FILE_NAME}.log</file>
        <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
            <level>${FILE_LEVEL}</level>
        </filter>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!--日志文件输出的文件名-->
            <!-- 文件扩展名设置为.zip/.gz后在文件滚动时会自动对旧日志进行压缩 -->
            <FileNamePattern>${LOG_HOME}/${FILE_NAME}.log.%d{yyyyMMdd}.%i.zip</FileNamePattern>
            <!-- 除按日志记录之外，还配置了日志文件不能超过512MB，若超过512MBM，日志文件会以索引0开始，命名日志文件，例如log-error-201
            <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>${FILE_MAX_SIZE}</maxFileSize>
```

图 7 互联网教育系统的 logback.xml

4、总结

本节课作为 Logback 框架的第一堂课，从 Logback 框架的基本概念入手，谈到了它的作用和依赖。然后，通过配置文件的节点给大家讲解了其需要配置的节点，包括：configuration、property、appender、logger 以及 root 节点。这里先对这些节点建立一个概念在代码实战的时候，我们会针对具体项目的配置文件给大家进行讲解。

下节课会延续 Logback 的讲解，在了解 Logback 之后会来看看代码中是如何使用 Logback 的。下期见，拜拜。