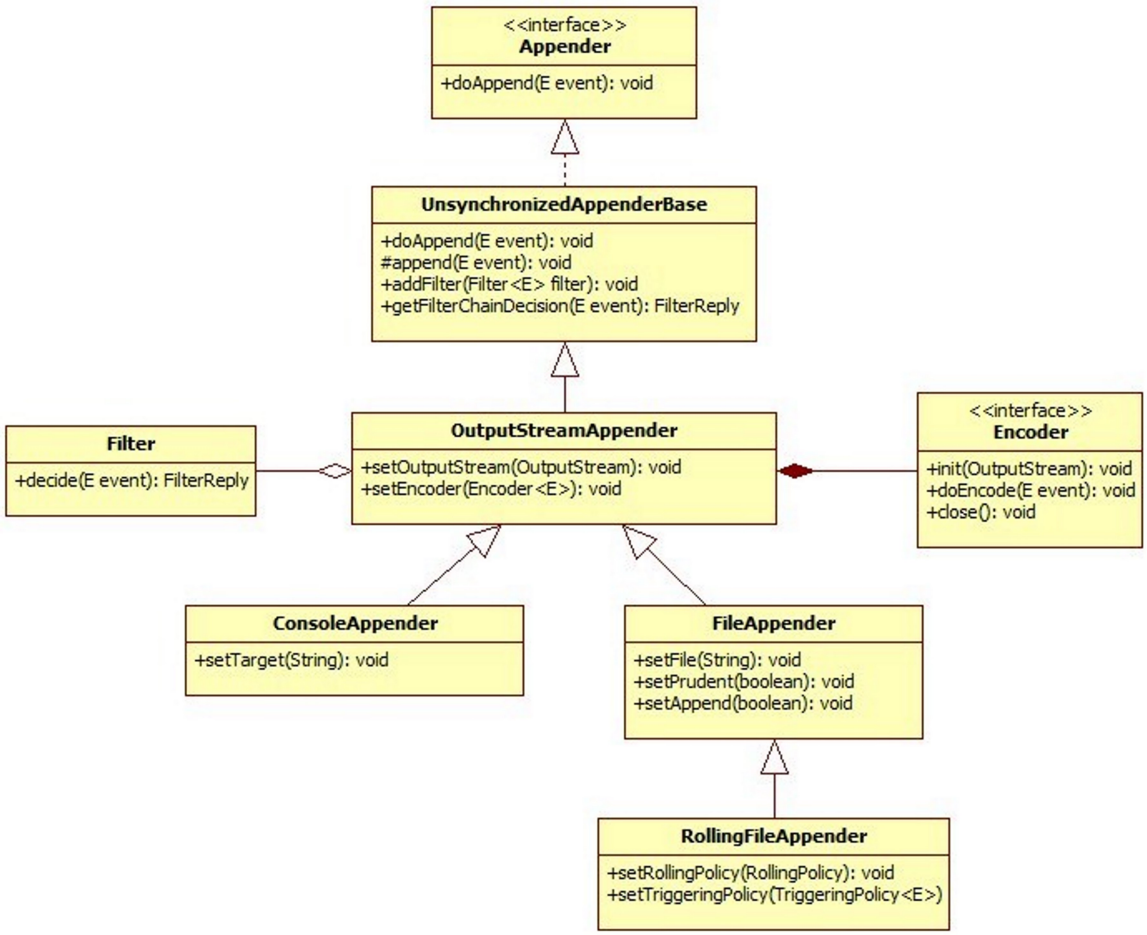
Logback

2021年6月7日 10:16

依赖

```
<dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
        <version>1.2.3</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-core</artifactId>
        <version>1.2.3</version>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.7.26</version>
    </dependency>
```

Appender 日志输出



1. 控制台

```
<appender name="console_out" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger - %msg%
n</pattern>
        </encoder>
        <charset>UTF-8</charset>
    </appender>
```

2. 文件

```
<appender name="file_out" class="ch.qos.logback.core.FileAppender">
        <file>./log/1.log</file>
        <append>true</append>
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger - %msg%
n</pattern>
        </encoder>
    </appender>
```

3. 滚动文件

```
<appender name="rolling_out" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <!-- 滚动规则 -->
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>rolling-file-%d{yyyy-MM-dd}.log</fileNamePattern>
            <!-- 最大历史文件数 -->
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger - %msg%
n</pattern>
        </encoder>
    </appender>
```

★ **滚动文件复杂操作**

文件大小滚动

```
<!-- 设置为按照索引的方式滚动，定义文件名称的时候使用%i作为占位符，滚动后会会用角标替换 -->
<rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
    <fileNamePattern>/logback/log/test-%i.log</fileNamePattern>
    <minIndex>1</minIndex>
    <maxIndex>3</maxIndex>
</rollingPolicy>
<!-- 指定文件最大尺寸，达到该尺寸，就触发rollingPolicy对应的策略，maxFileSize属性指定文件大
小 -->
<triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
        <maxFileSize>1MB</maxFileSize>
</triggeringPolicy>
```

时间配置文件大小滚动

从class名就可以看出是时间文件命名和条件触发的混合逻辑

```
<!-- 使用按照时间滚动策略，内嵌按照文件大小来分隔日志的触发器策略 -->
<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <fileNamePattern>/logback/log/test-%d{yyyy-MM-dd}-%i.log</fileNamePattern>
    <!-- 使用SizeAndTimeBasedFNATP实现，可以看一下TimeBasedRollingPolicy源码中对应
timeBasedFileNamingAndTriggeringPolicy的类型，根据类型确定需要使用的class类 -->
    <timeBasedFileNamingAndTriggeringPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
        <maxFileSize>1MB</maxFileSize>
    </timeBasedFileNamingAndTriggeringPolicy>
```

```
</rollingPolicy>
```

属性的占位符管理

配置：<property name="file_url" value="./log/1.log" />
使用：${file_url}

注意点:

```
<!-- 更细粒度的level会覆盖上层level -->
<logger name="org.example.LogDemo" level="debug" additivity="true" />
<logger name="org.example" level="debug" additivity="false">
    <appender-ref ref="console_out" />
</logger>

<!-- 同名的logger会被xml配置按顺序覆盖 -->
<logger name="org.example" level="debug" additivity="false">
    <appender-ref ref="console_out" />
</logger>
<logger name="org.example" level="info" additivity="false">
    <appender-ref ref="console_out" />
</logger>
```

filter 过滤
可用于对日志不同级别拆分输出

```
<!-- 过滤器，过滤掉不是指定日志水平的日志 -->
<filter class="ch.qos.logback.classic.filter.LevelFilter">
    <!-- 设置日志级别 -->
    <level>DEBUG</level>
    <!-- 如果跟该日志水平相匹配，则接受 -->
    <onMatch>ACCEPT</onMatch>
    <!-- 如果跟该日志水平不匹配，则过滤掉 -->
    <onMismatch>DENY</onMismatch>
</filter>
```

文件时间戳命名

```
<!--获取时间戳字符串 key为名称  datePattern为将解析配置文件的时间作为当前时间格式化的标准-->
<timestamp key="bySecond" datePattern="yyyyMMdd'T'HHmmss"/>
```

使用：<file>log-${bySecond}.txt</file>

参考资料：

https://blog.csdn.net/xintonghanchuang/article/details/91348257

https://www.cnblogs.com/yw0219/p/9361040.html

https://blog.csdn.net/millery22/article/details/86672284