log4j2使用手册（中文）第九章 Appenders（二）

在下喵星人 关注
2019.10.12 16:13:23 字数 5,134 阅读 535

Apache CouchDB的NoSQLAppender

本节详细介绍了CouchDB的NoSQLAppender提供程序的特殊化。 NoSQLAppender使用内部轻量级提供程序接口将日志事件写入NoSQL数据库。

CouchDB参数

参数名称	Type	Description
factoryClassName	Class	要提供与CouchDB数据库的连接，可以使用此属性和factoryMethodName指定要从中获取连接的类和静态方法。 该方法必须返回org.lightcouch.CouchDbClient或org.lightcouch.CouchDbProperties。 如果使用工厂方法提供连接，则不得指定databaseName，protocol，server，port，username或password属性。
factoryMethodName	Method	请参阅文档以获取属性factoryClassName。
databaseName	String	如果未指定factoryClassName和factoryMethodName以提供CouchDB连接，则必须使用此属性指定CouchDB数据库名称。 您还必须指定用户名和密码。 您也可以选择指定协议（默认为http），服务器（默认为localhost）和aport（http默认为80，https默认为443）。
protocol	String	必须是“http”或“https”。 请参阅属性databaseName的文档。
server	String	请参阅属性databaseName的文档。
port	int	请参阅属性databaseName的文档。
username	String	请参阅属性databaseName的文档。

推荐阅读
前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946
基于 Nginx+lua+Memcache 实现灰度发布
阅读 793
你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453
Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542
性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

名称	pe	
password	String	请参阅属性databaseName的文档。

以下是NoSQLAppender和CouchDB提供程序的一些示例配置：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="error">
  <Appenders>
    <NoSql name="databaseAppender">
      <CouchDb databaseName="applicationDb" protocol="https" server="couch.example.org"
               username="loggingUser" password="abc123" />
    </NoSql>
  </Appenders>
  <Loggers>
    <Root level="warn">
      <AppenderRef ref="databaseAppender"/>
    </Root>
  </Loggers>
</Configuration>
```

OutputStreamAppender

OutputStreamAppender为许多其他Appender提供了基础，例如将事件写入输出流的File和Socket appender。 它无法直接配置。 OutputStreamAppender提供对immediateFlush和缓冲的支持。 OutputStreamAppender使用OutputStreamManager来处理实际的I / O，允许Appenders在多个配置中共享流。

RandomAccessFileAppender

RandomAccessFileAppender类似于标准的FileAppender，除了它总是被缓冲（这不能被关闭），并且在内部它使用ByteBuffer + RandomAccessFile而不是BufferedOutputStream。 与FileAppender相比，我们在测量中看到“bufferedIO = true”，性能提升了20-200%。 与FileAppender类似，RandomAccessFileAppender使用RandomAccessFileManager实际执行文件I / O. 虽然无法共享来自不同Configuration的RandomAccessFileAppender，但如果Manager可访问，则RandomAccessFileManagers可以是。 例如，servlet容器中的两个Web应用程序可以拥有自己的配置，并且如果Log4j位于两个共用的ClassLoader中，则可以安全地写入同一文件。

RandomAccessFileAppender参数

参数名称	类型	描述
append	boolean	如果为true - 默认值，则记录将附加到文件末尾。 设置为false时，将在写入新记录之前清除该文件。
fileNa	Stri	要写入的文件的名称。 如果该文件或其任何父目录不存在，则将创建它们

名称	类型	描述
filters	Filter	一个过滤器，用于确定此Appender是否应该处理该事件。 使用CompositeFilter可以使用多个Filter。
immediateFlush	boolean	设置为true时 - 默认值，每次写入后都会进行刷新。 这将保证数据写入磁盘，但可能会影响性能。 每次写入后刷新仅在将此appender与同步记录器一起使用时才有用。 即使immediateFlush设置为false，异步记录器和追加器也会在一批事件结束时自动刷新。 这也保证了数据被写入磁盘但效率更高。
bufferSize	int	缓冲区大小，默认为262,144字节（256 * 1024）。
layout	Layout	用于格式化LogEvent的布局。 如果未提供布局，则将使用默认的模式布局“%m%n”。
name	String	Appender的名字。
ignoreExceptions	boolean	默认值为true，导致在将事件附加到内部记录然后被忽略时遇到异常。 设置为false时，异常将传播给调用者。 将此Appender包装在FailoverAppender中时，必须将此参数设置为false。

以下是RandomAccessFile配置示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <RandomAccessFile name="MyFile" fileName="logs/app.log">
      <PatternLayout>
        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
      </PatternLayout>
    </RandomAccessFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="MyFile"/>
    </Root>
  </Loggers>
</Configuration>
```

RewriteAppender

RewriteAppender允许LogEvent在被另一个Appender处理之前进行操作。 这可用于屏蔽敏感信息（如密码）或将信息注入每个事件。 必须使用RewritePolicy配置RewriteAppender。 应该在引用的任何Appender之后配置RewriteAppender以允许它正确关闭。

RewriteAppender参数

参数名称	类型	描述

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

在下喵星人

AppenderRef	String	操作LogEvent后要调用的Appender的名称。 可以配置多个AppenderRef元素。
filter	Filter	一个过滤器，用于确定此Appender是否应该处理该事件。 使用CompositeFilter可以使用多个Filter。
name	String	Appender的名字。
rewritePolicy	RewritePolicy	RewritePolicy将操纵LogEvent。
ignoreExceptions	boolean	默认值为true，导致在将事件附加到内部记录然后被忽略时遇到异常。 设置为false时，异常将传播给调用者。 将此Appender包装在FailoverAppender中时，必须将此参数设置为false。

RewritePolicy

RewritePolicy是一个接口，允许实现在将LogEvent传递给Appender之前检查并可能修改它们。 RewritePolicy声明了一个必须实现的名为rewrite的方法。 该方法传递给LogEvent，可以返回相同的事件或创建一个新事件。

MapRewritePolicy

MapRewritePolicy将评估包含MapMessage的LogEvent，并将添加或更新Map的元素。

参数名称	类型	描述
mode	String	“Add” 或 “Update”
keyValuePair	KeyValuePair[]	一组键和它们的值。

以下配置显示RewriteAppender，其配置为将产品密钥及其值添加到MapMessage。 ：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <Console name="STDOUT" target="SYSTEM_OUT">
      <PatternLayout pattern="%m%n"/>
    </Console>
    <Rewrite name="rewrite">
      <AppenderRef ref="STDOUT"/>
      <MapRewritePolicy mode="Add">
        <KeyValuePair key="product" value="TestProduct"/>
      </MapRewritePolicy>
    </Rewrite>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="Rewrite"/>
    </Root>
  </Loggers>
</Configuration>
```

PropertiesRewritePolicy

PropertiesRewritePolicy会将策略上配置的属性添加到正在记录的ThreadContext Map中。 这些属性不会添加到实际的ThreadContext Map中。 属性值可能包含将在处理配置时以及记录事件时评估的变量

properties	Property[]	一个或多个Property元素，用于定义要添加到ThreadContext Map的键和值。

以下配置显示了RewriteAppender，其配置为将产品密钥及其值添加到MapMessage：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <Console name="STDOUT" target="SYSTEM_OUT">
      <PatternLayout pattern="%m%n"/>
    </Console>
    <Rewrite name="rewrite">
      <AppenderRef ref="STDOUT"/>
      <PropertiesRewritePolicy>
        <Property name="user">${sys:user.name}</Property>
        <Property name="env">${sys:environment}</Property>
      </PropertiesRewritePolicy>
    </Rewrite>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="Rewrite"/>
    </Root>
  </Loggers>
</Configuration>
```

LoggerNameLevelRewritePolicy

您可以使用此策略通过更改事件级别来减少第三方代码中的记录器。
LoggerNameLevelRewritePolicy将为给定的记录器名称前缀重写日志事件级别。 您可以使用记录器名称前缀和一对级别配置LoggerNameLevelRewritePolicy，其中一对定义源级别和目标级别。

参数名称	类型	描述
logger	String	记录器名称，用作测试每个事件的记录器名称的前缀。
LevelPair	KeyValuePair[]	一组键及其值，每个键是一个源级别，每个值都是一个目标级别。

以下配置显示了RewriteAppender，其配置为将级别INFO映射到DEBUG，并将所有以com.foo.bar开头的记录器的级别WARN级别设置为INFO。

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp">
  <Appenders>
    <Console name="STDOUT" target="SYSTEM_OUT">
      <PatternLayout pattern="%m%n"/>
    </Console>
    <Rewrite name="rewrite">
      <AppenderRef ref="STDOUT"/>
      <LoggerNameLevelRewritePolicy logger="com.foo.bar">
        <KeyValuePair key="INFO" value="DEBUG"/>
        <KeyValuePair key="WARN" value="INFO"/>
      </LoggerNameLevelRewritePolicy>
    </Rewrite>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="Rewrite"/>
    </Root>
  </Loggers>
</Configuration>
```

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

在下喵星人

RollingFileManager（扩展OutputStreamManager）来实际执行文件I / O并执行翻转。 虽然无法共享来自不同Configuration的RolloverFileAppenders，但RollingFileManagers可以是Manager可访问的。 例如，servlet容器中的两个Web应用程序可以拥有自己的配置，并且如果Log4j位于两个共用的ClassLoader中，则可以安全地写入同一文件。

RollingFileAppender需要TriggeringPolicy和RolloverStrategy。 触发策略确定在RolloverStrategy定义应如何完成翻转时是否应执行翻转。 如果未配置RolloverStrategy，则RollingFileAppender将使用DefaultRolloverStrategy。 从log4j-2.5开始，可以在DefaultRolloverStrategy中配置自定义删除操作以在翻转时运行。 从2.8开始，如果没有配置文件名，那么将使用DirectWriteRolloverStrategy而不是DefaultRolloverStrategy。 从log4j-2.9开始，可以在DefaultRolloverStrategy中配置自定义POSIX文件属性视图操作以在翻转时运行，如果未定义，将应用来自RollingFileAppender的继承POSIX文件属性视图。

RollingFileAppender不支持文件锁定。

RollingFileAppender参数

参数名称	类型	描述
append	boolean	如果为true - 默认值，则记录将附加到文件末尾。 设置为false时，将在写入新记录之前清除该文件。
bufferedIO	boolean	如果为true - 默认情况下，记录将写入缓冲区，当缓冲区已满时，数据将写入磁盘;如果设置了immediateFlush，则写入记录时。 文件锁定不能与bufferedIO一起使用。 性能测试表明，即使启用了immediateFlush，使用缓冲I / O也可以显着提高性能。
bufferSize	int	当bufferedIO为true时，这是缓冲区大小，默认为8192字节。
createOnDemand	boolean	appender按需创建文件。 appender仅在日志事件通过所有过滤器并将其路由到此appender时创建该文件。 默认为false。
filter	Filter	一个过滤器，用于确定此Appender是否应该处理该事件。 使用CompositeFilter可以使用多个Filter。
fileName	String	要写入的文件的名称。 如果该文件或其任何父目录不存在，则将创建它们。
filePattern	String	存档日志文件的文件名模式。 模式的格式取决于使用的RolloverPolicy。 DefaultRolloverPolicy将接受与SimpleDateFormatand /兼容的日期/时间模式和表示整数计数器的%i。 该模式还支持在运行时进行插值，因此任何Lookups（例如DateLookup）都可以包含在模式中。
immediateFlush	boolean	设置为true时 - 默认值，每次写入后都会进行刷新。 这将保证数据写入磁盘但可能影响性能。每次写入后刷新仅在将此appender与同步记录器一起使用时才有用。 即使immediateFlush设置为false，异步记录器和追加器也会在一批事件结束时自动刷新。 这也保证了数据被写入磁盘但效率更高。
lay	La	用于格式化LogEvent的布局。 如果未提供布局，则将使用默认的模式布

名称	型	
name	String	Appender的名字。
policy	TriggeringPolicy	用于确定是否应发生翻转的策略。
strategy	RolloverStrategy	用于确定存档文件的名称和位置的策略。
ignoreExceptions	boolean	默认值为true，导致在将事件附加到内部记录然后被忽略时遇到异常。 设置为false时，异常将传播给调用者。 将此Appender包装在FailoverAppender中时，必须将此参数设置为false。
filePermissions	String	POSIX格式的文件属性权限，以便在创建文件时应用。 底层文件系统应支持POSIX文件属性视图。 示例：rw ——-或rw-rw-rw-等...
fileOwner	String	文件所有者定义何时创建文件。 出于安全原因，可能会限制更改文件的所有者，并且不允许操作IOException。 只有具有有效用户ID等于文件用户ID或具有适当权限的进程才可以更改文件的所有权if_POSIX_CHOWN_RESTRICTED对路径有效。 底层文件系统应支持文件所有者属性视图。
fileGroup	String	文件组，用于定义创建文件的时间。 底层文件系统应支持POSIX文件属性视图。

Triggering Policies

Composite Triggering Policy

CompositeTriggeringPolicy组合了多个触发策略，如果任何已配置的策略返回true，则返回true。 CompositeTriggeringPolicy只需通过将其他策略包装在Policies元素中即可配置。

例如，以下XML片段定义了在JVM启动时，当日志大小达到20兆字节时以及当前日期不再与日志的开始日期匹配时翻转日志的策略。

```
<Policies>
  <OnStartupTriggeringPolicy />
  <SizeBasedTriggeringPolicy size="20 MB" />
  <TimeBasedTriggeringPolicy />
</Policies>
```

Cron Triggering Policy

推荐阅读
前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946
基于 Nginx+lua+Memcache 实现灰度发布
阅读 793
你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453
Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542
性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

在下喵星人

filePattern属性应该包含一个时间戳，否则每次翻转都会覆盖目标文件。

CronTriggeringPolicy参数

参数名称	类型	描述
schedule	String	cron表达式。 表达式与Quartz调度程序中允许的表达式相同。 有关表达式的完整说明，请参阅CronExpression。
evaluateOnStartup	boolean	在启动时，将根据文件的最后修改时间戳评估cron表达式。 如果cron表达式指示应该在该时间与当前时间之间发生翻转，则文件将立即翻转。

OnStartup Triggering Policy

如果日志文件早于当前JVM的开始时间且达到或超过最小文件大小，则OnStartupTriggeringPolicy策略将导致翻转。

OnStartupTriggeringPolicy参数

参数名称	类型	描述
minSize	long	文件必须翻转的最小大小。 无论文件大小是多少，大小为零都会导致翻转。 默认值为1，这将阻止滚动空文件。

Google App Engine note:

在Google App Engine中运行时，如果日志文件早于Log4J初始化的时间，则OnStartup策略会导致翻转。 （Google App Engine限制对某些类的访问，因此Log4J无法使用java.lang.management.ManagementFactory.getRuntimeMXBean（）.getStartTime（）确定JVM启动时间，而是回退到Log4J初始化时间。）

SizeBased Triggering Policy

一旦文件达到指定大小，SizeBasedTriggeringPolicy将导致翻转。 大小可以以字节为单位指定，后缀为KB，MB或GB，例如20MB。 文件模式必须包含%i，否则目标文件将在每次翻转时被覆盖，因为SizeBased触发策略不会导致文件名中的时间戳值发生更改。

TimeBased Triggering Policy

一旦日期/时间模式不再适用于活动文件，TimeBasedTriggeringPolicy将导致翻转。 此策略接受interval属性，该属性指示基于时间模式和modulate布尔属性进行翻转的频率。

TimeBasedTriggeringPolicy参数

参数名称	类型	描述
interval	integer	根据日期模式中最具体的时间单位进行翻转的频率。 例如，使用以小时为最具体项目的日期模式，并且每4小时增加4次翻转。 默认值为1。

在下喵星人

modulate	boolean	指示是否应调整间隔以使间隔边界上发生下一次翻转。 例如，如果项目是小时，当前小时是凌晨3点，间隔是4，那么第一次翻转将在凌晨4点发生，然后下一次翻转将发生在上午8点，中午，下午4点等。
maxRandomDelay	integer	表示随机延迟翻转的最大秒数。 默认情况下，该值为0表示没有延迟。 此设置在将多个应用程序配置为同时翻转日志文件的服务器上非常有用，并且可以跨时间分担这样做的负担。

Rollover Strategies

Default Rollover Strategy

默认翻转策略接受日期/时间模式和RollingFileAppender本身上指定的filePattern属性中的整数。 如果存在日期/时间模式，则将替换为当前日期和时间值。 如果模式包含整数，则每次翻转时都会递增。 如果模式在模式中包含日期/时间和整数，则整数将递增，直到日期/时间模式的结果发生变化。 如果文件模式以“.gz”，“.zip”，“。bz2”，“。deflate”，“。pack200”或“.xz”结尾，则使用与后缀匹配的压缩方案压缩生成的存档。 bzip2，Deflate，Pack200和XZ格式需要Apache Commons Compress。 此外，XZ需要XZ for Java。 该模式还可以包含可以在运行时解析的查找引用，如下面的示例所示。

默认翻转策略支持三种用于递增计数器的变体。 首先是“固定窗口”策略。 为了说明它是如何工作的，假设min属性设置为1，max属性设置为3，文件名为“foo.log”，文件名模式为“foo-%i.log”。

rollover次数	有效的输出目标	存档的日志文件	描述
0	foo.log	-	所有日志记录都将转到初始文件。
1	foo.log	foo-1.log	在第一次翻转期间，foo.log被重命名为foo-1.log。 创建一个新的foo.log文件并开始写入。
2	foo.log	foo-1.log, foo-2.log	在第二次翻转期间，foo-1.log被重命名为foo-2.log，foo.log被重命名为foo-1.log。 创建一个新的foo.log文件并开始写入。
3	foo.log	foo-1.log, foo-2.log, foo-3.log	在第三次翻转期间，foo-2.log被重命名为foo-3.log，foo-1.log被重命名为foo-2.log，foo.log被重命名为foo-1.log。 创建一个新的foo.log文件并开始写入。
4	foo.log	foo-1.log, foo-2.log, foo-3.log	在第四次及以后的翻转中，删除foo-3.log，将foo-2.log重命名为foo-3.log，将foo-1.log重命名为foo-2.log，将foo.log重命名为foo-1.log。 创建一个新的foo.log文件并开始写入。

通过对比，当fileIndex属性设置为“max”但所有其他设置相同时，将执行以下操作。

rollover次数	有效的输出目标	存档的日志文件	描述
0	foo.log	-	所有日志记录都将转到初始文件。

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

er次数	出目标	志文件	描述
2	foo.log	foo-1.log, foo-2.log	在第二次翻转期间，foo.log被重命名为foo-2.log。 创建一个新的foo.log文件并开始写入。
3	foo.log	foo-1.log, foo-2.log, foo-3.log	在第三次翻转期间，foo.log被重命名为foo-3.log。 创建一个新的foo.log文件并开始写入。
4	foo.log	foo-1.log, foo-2.log, foo-3.log	在第四次及以后的翻转中，删除foo-1.log，将foo-2.log重命名为foo-1.log，将foo-3.log重命名为foo-2.log，将foo.log重命名为foo-3.log。 创建一个新的foo.log文件并开始写入。

最后，从版本2.8开始，如果fileIndex属性设置为“nomax”，则将忽略最小值和最大值，文件编号将增加1，每个翻转将具有递增的更高值，没有最大文件数。

DefaultRolloverStrategy参数

参数名称	类型	描述
fileIndex	String	如果设置为“max”（默认值），索引较高的文件将比索引较小的文件更新。 如果设置为“min”，文件重命名和计数器将遵循上述固定窗口策略。
min	integer	计数器的最小值。 默认值为1。
max	integer	计数器的最大值。 达到此值后，将在后续翻转时删除旧档案。 默认值为7。
compressionLevel	integer	设置压缩级别0-9，其中0 =无，1 =最佳速度，9 =最佳压缩。 仅适用于ZIP文件。
tempCompressedFilePattern	String	压缩期间存档日志文件的文件名模式。

DirectWrite Rollover Strategy

DirectWriteRolloverStrategy使日志事件直接写入由文件模式表示的文件。 使用此策略文件不会执行重命名。 如果基于大小的触发策略导致在指定的时间段内写入多个文件，则它们将从1开始编号并连续递增，直到发生基于时间的翻转。

警告：如果文件模式具有后缀，指示应该进行压缩，则在关闭应用程序时不会压缩当前文件。此外，如果时间改变使得文件模式不再与当前文件匹配，则它也不会在启动时被压缩。

DirectWriteRolloverStrategy参数

参数名称	类型	描述
fileIndex	Strin	如果设置为“max”（默认值），索引较高的文件将比索引较小的文件更新。 如果设置为“min”，文件重命名和计数器将遵循上述固定窗口策略

min	integer	计数器的最小值。 默认值为1。
max	integer	计数器的最大值。 达到此值后，将在后续翻转时删除旧档案。 默认值为7。
compressionLevel	integer	设置压缩级别0-9，其中0 =无，1 =最佳速度，9 =最佳压缩。 仅适用于ZIP文件。
tempCompressedFilePattern	String	压缩期间存档日志文件的文件名模式。

下面是使用RollingFileAppender同时具有基于时间和大小的触发策略的示例配置，将在同一天（1-7）创建最多7个存档，这些存档基于当前年份和月份存储在目录中，并且将 使用gzip压缩每个存档：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <RollingFile name="RollingFile" fileName="logs/app.log"
                 filePattern="logs/$${date:yyyy-MM}/app-%d{MM-dd-yyyy}-%i.log.gz">
      <PatternLayout>
        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
      </PatternLayout>
      <Policies>
        <TimeBasedTriggeringPolicy />
        <SizeBasedTriggeringPolicy size="250 MB"/>
      </Policies>
    </RollingFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingFile"/>
    </Root>
  </Loggers>
</Configuration>
```

第二个示例显示了一个翻转策略，在删除之前最多可保留20个文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <RollingFile name="RollingFile" fileName="logs/app.log"
                 filePattern="logs/$${date:yyyy-MM}/app-%d{MM-dd-yyyy}-%i.log.gz">
      <PatternLayout>
        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
      </PatternLayout>
      <Policies>
        <TimeBasedTriggeringPolicy />
        <SizeBasedTriggeringPolicy size="250 MB"/>
      </Policies>
      <DefaultRolloverStrategy max="20"/>
    </RollingFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingFile"/>
    </Root>
  </Loggers>
</Configuration>
```

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

在下喵星人

每个存档，当小时可以被6整除时，每6个小时滚动一次：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <RollingFile name="RollingFile" fileName="logs/app.log"
                 filePattern="logs/$${date:yyyy-MM}/app-%d{yyyy-MM-dd-HH}-%i.log.gz">
      <PatternLayout>
        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
      </PatternLayout>
      <Policies>
        <TimeBasedTriggeringPolicy interval="6" modulate="true"/>
        <SizeBasedTriggeringPolicy size="250 MB"/>
      </Policies>
    </RollingFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingFile"/>
    </Root>
  </Loggers>
</Configuration>
```

此示例配置使用RollingFileAppender同时具有基于cron和size的触发策略，并直接写入无限数量的归档文件。 cron触发器每小时导致翻转，而文件大小限制为250MB：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <RollingFile name="RollingFile" filePattern="logs/app-%d{yyyy-MM-dd-HH}-%i.log.gz">
      <PatternLayout>
        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
      </PatternLayout>
      <Policies>
        <CronTriggeringPolicy schedule="0 0 * * * ?"/>
        <SizeBasedTriggeringPolicy size="250 MB"/>
      </Policies>
    </RollingFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingFile"/>
    </Root>
  </Loggers>
</Configuration>
```

此示例配置与上一个相同，但将每小时保存的文件数限制为10：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <RollingFile name="RollingFile" filePattern="logs/app-%d{yyyy-MM-dd-HH}-%i.log.gz">
      <PatternLayout>
        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
      </PatternLayout>
      <Policies>
        <CronTriggeringPolicy schedule="0 0 * * * ?"/>
        <SizeBasedTriggeringPolicy size="250 MB"/>
      </Policies>
      <DirectWriteRolloverStrategy maxFiles="10"/>
    </RollingFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingFile"/>
    </Root>
  </Loggers>
</Configuration>
```

在下喵星人

Log4j-2.5引入了一个Delete操作，使用户可以更好地控制在翻转时删除哪些文件，而不是使用DefaultRolloverStrategy max属性。 “删除”操作允许用户配置一个或多个条件，以选择要相对于基目录删除的文件。

请注意，可以删除任何文件，而不仅仅是滚动日志文件，因此请谨慎使用此操作！ 使用testMode参数，您可以测试配置，而不会意外删除错误的文件。

Delete参数

参数名称	类型	描述
basePath	String	必填。 从哪里开始扫描要删除的文件的基本路径。
maxDepth	int	要访问的目录的最大级别数。 值0表示仅访问起始文件（基本路径本身），除非安全管理器拒绝。 值Integer.MAX_VALUE表示应访问所有级别。 默认值为1，表示仅指定基目录中的文件。
followLinks	boolean	是否遵循符号链接。 默认值为false。
testMode	boolean	如果为true，则不删除文件，而是在INFO级别向状态记录器打印消息。 使用此选项进行干运行以测试配置是否按预期工作。 默认值为false。
pathSorter	PathSorter	一个实现PathSorter的插件 在选择要删除的文件之前对文件进行排序的界面。 默认设置是首先对最近修改的文件进行排序。
pathConditions	PathCondition[]	如果未指定ScriptCondition，则为必需。一个或多个PathCondition元素。如果指定了多个条件，则它们都需要在删除之前接受路径。条件可以嵌套，在这种情况下，仅当外部条件接受路径时才评估内部条件。如果条件没有嵌套，则可以按任何顺序进行评估。条件也可以与逻辑运算符AND，OR和NOT结合使用IfAll，IfAny和IfNot复合条件。用户可以创建自定义条件或使用内置条件: -IfFileName-接受其路径（相对于基本路径）与正则表达式匹配的文件或者glob). -IfLastModified-接受与指定持续时间一样长或过早的文件。 -IfAccumulatedFileCount-在文件树遍历期间超过某个计数阈值后接受路径. -IfAccumulatedFileSize-在文件树遍历期间超过累积文件大小阈值后接受路径。 - IfAll - 如果所有嵌套条件都接受它（逻辑AND），则接受路径。可以按任何顺序评估嵌套条件。- IfAny - 如果其中一个嵌套条件接受它（逻辑OR），则接受路径。可以按任何顺序评估嵌套条件。- IfNot - 如果嵌套条件不接受它，则接受路径（逻辑NOT）。

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

名称	型	
scriptCondition	ScriptCondition	如果未指定PathConditions，则为必需。 指定脚本的ScriptCondition元素。 ScriptCondition应包含Script，ScriptRef或ScriptFile元素，用于指定要执行的逻辑。 （有关配置ScriptFiles和ScriptRef的更多示例，另请参阅ScriptFilter文档。 ）该脚本传递了许多参数，包括在基本路径下找到的路径列表（最多为maxDepth），并且必须返回包含要删除的路径的列表。

IfFileName条件参数

参数名	类型	描述
glob	String	如果未指定正则表达式，则为必需。 使用类似于正则表达式但具有更简单语法)的有限模式语言匹配相对路径（相对于基本路径）。
regex	String	如果未指定glob，则为必需。 使用Pattern类定义的正则表达式匹配相对路径（相对于基本路径）。
nestedConditions	PathCondition[]	一组可选的嵌套PathConditions。 如果存在任何嵌套条件，则它们都需要在删除之前接受该文件。 仅当外部条件接受文件（如果路径名称匹配）时，才会评估嵌套条件。

IfLastModified条件参数

参数名称	类型	描述
age	String	必填。 指定持续时间。 该条件接受与指定持续时间相同的旧文件或更旧文件。
nestedConditions	PathCondition[]	一组可选的嵌套PathConditions。 如果存在任何嵌套条件，则它们都需要在删除之前接受该文件。 仅当外部条件接受文件时（如果文件足够大），才会评估嵌套条件。

IfAccumulatedFileCount条件参数

参数名称	类型	描述
exceeds	int	必填。 将删除文件的阈值计数。
nestedConditions	PathCondition[]	一组可选的嵌套PathConditions。 如果存在任何嵌套条件，则它们都需要在删除之前接受该文件。 仅当外部条件接受文件时（如果已超过阈值计数），才会评估嵌套条件。

IfAccumulatedFileSize条件参数

参数名称	类型	描述
exceeds	String	必填。 阈值累计文件大小，将从中删除文件。 大小可以以字节为单位指定，后缀为KB，MB或GB，例如20MB。
nestedCon	PathCond	一组可选的嵌套PathConditions。 如果存在任何嵌套条件，则它们都需要在删除之前接受该文件。 仅当外部条件接受文件时才会评估嵌套条件（如果已

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

在下喵星人

60天或更早的所有文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Properties>
    <Property name="baseDir">logs</Property>
  </Properties>
  <Appenders>
    <RollingFile name="RollingFile" fileName="${baseDir}/app.log"
          filePattern="${baseDir}/$${date:yyyy-MM}/app-%d{yyyy-MM-dd}.log.gz">
      <PatternLayout pattern="%d %p %c{1.} [%t] %m%n" />
      <CronTriggeringPolicy schedule="0 0 0 * * ?"/>
      <DefaultRolloverStrategy>
        <Delete basePath="${baseDir}" maxDepth="2">
          <IfFileName glob="*/app-*.log.gz" />
          <IfLastModified age="60d" />
        </Delete>
      </DefaultRolloverStrategy>
    </RollingFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingFile"/>
    </Root>
  </Loggers>
</Configuration>
```

下面是使用RollingFileAppender同时具有基于时间和大小的触发策略的示例配置，将在同一天（1-100）创建最多100个存档，这些存档存储在基于当前年份和月份的目录中，并且将 使用gzip压缩每个存档，并且每小时滚动一次。 在每次翻转期间，此配置将删除与“/ app - .log.gz”匹配且30天或更早的文件，但保留最新的100 GB或最新的10个文件，以先到者为准。

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Properties>
    <Property name="baseDir">logs</Property>
  </Properties>
  <Appenders>
    <RollingFile name="RollingFile" fileName="${baseDir}/app.log"
          filePattern="${baseDir}/$${date:yyyy-MM}/app-%d{yyyy-MM-dd-HH}-%i.log.gz">
      <PatternLayout pattern="%d %p %c{1.} [%t] %m%n" />
      <Policies>
        <TimeBasedTriggeringPolicy />
        <SizeBasedTriggeringPolicy size="250 MB"/>
      </Policies>
      <DefaultRolloverStrategy max="100">
        <!--
        Nested conditions: the inner condition is only evaluated on files
        for which the outer conditions are true.
        -->
        <Delete basePath="${baseDir}" maxDepth="2">
          <IfFileName glob="*/app-*.log.gz">
            <IfLastModified age="30d">
              <IfAny>
                <IfAccumulatedFileSize exceeds="100 GB" />
                <IfAccumulatedFileCount exceeds="10" />
              </IfAny>
            </IfLastModified>
          </IfFileName>
        </Delete>
      </DefaultRolloverStrategy>
    </RollingFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingFile"/>
    </Root>
  </Loggers>
</Configuration>
```

名称		
script	Script, ScriptFile or ScriptRef	Script元素，指定要执行的逻辑。 该脚本将传递在基本路径下找到的路径列表，并且必须将要删除的路径作为java.util.List <PathWithAttributes>返回。 有关如何配置ScriptFiles和ScriptRefs的示例，另请参阅ScriptFilter文档。

Script 参数

参数名称	类型	描述
basePath	java.nio.file.Path	“删除”操作开始扫描要删除的文件的目录。 可用于对pathList中的路径进行相对化。
pathList	java.util.List<PathWithAttributes>	在基本路径下找到的路径列表，直到指定的最大深度，首先对最近修改的文件进行排序。 该脚本可以自由修改并返回此列表。
statusLogger	StatusLogger	StatusLogger，可用于在脚本执行期间记录内部事件。
configuration	Configuration	拥有此ScriptCondition的配置。
substitutor	StrSubstitutor	StrSubstitutor用于替换查找变量。
?	String	配置中声明的任何属性。

下面是一个示例配置，它使用RollingFileAppender和cron触发策略配置为每天午夜触发。 档案存储在基于当前年份和月份的目录中。 该脚本返回13日星期五基本目录下的翻转文件列表。 Delete操作将删除脚本返回的所有文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="trace" name="MyApp" packages="">
  <Properties>
    <Property name="baseDir">logs</Property>
  </Properties>
  <Appenders>
    <RollingFile name="RollingFile" fileName="${baseDir}/app.log"
          filePattern="${baseDir}/$${date:yyyy-MM}/app-%d{yyyyMMdd}.log.gz">
      <PatternLayout pattern="%d %p %c{1.} [%t] %m%n" />
      <CronTriggeringPolicy schedule="0 0 0 * * ?"/>
      <DefaultRolloverStrategy>
        <Delete basePath="${baseDir}" maxDepth="2">
          <ScriptCondition>
            <Script name="superstitious" language="groovy"><![CDATA[
                import java.nio.file.*;

                def result = [];
                def pattern = ~/\d*\/app-(\d*)\.log\.gz/;

                pathList.each { pathWithAttributes ->
                  def relative = basePath.relativize pathWithAttributes.path
                  statusLogger.trace 'SCRIPT: relative path=' + relative + " (base=$basePath)"

                  // remove files dated Friday the 13th

                  def matcher = pattern.matcher(relative.toString());
                  if (matcher.find()) {
                    def dateString = matcher.group(1);
                    def calendar = Date.parse("yyyyMMdd", dateString).toCalendar();
                    def friday13th = calendar.get(Calendar.DAY_OF_MONTH) == 13 \
                                && calendar.get(Calendar.DAY_OF_WEEK) == Calendar.FRIDAY;
                    if (friday13th) {
                      result.add pathWithAttributes;
```

在下喵星人 关注 赞赏支持

```
                result;
              ]] >
            </Script>
          </ScriptCondition>
        </Delete>
      </DefaultRolloverStrategy>
    </RollingFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingFile"/>
    </Root>
  </Loggers>
</Configuration>
```

日志存档文件属性视图策略：Rollover上的自定义文件属性

Log4j-2.9引入了PosixViewAttribute操作，使用户可以更好地控制应该应用哪个文件属性权限，所有者和组。 PosixViewAttribute操作允许用户配置一个或多个条件，以选择相对于基目录的符合条件的文件。

PosixViewAttribute参数

参数名称	类型	描述
basePath	String	必填。 从哪里开始扫描文件以应用属性的基本路径。
maxDepth	int	要访问的目录的最大级别数。 值0表示仅访问起始文件（基本路径本身），除非安全管理器拒绝。 值Integer.MAX_VALUE表示应访问所有级别。 默认值为1，表示仅指定基目录中的文件。
followLinks	boolean	是否遵循符号链接。 默认值为false。
pathConditions	PathCondition[]	请参阅DeletePathCondition
filePermissions	String	执行操作时应用POSIX格式的文件属性权限。下层文件系统应支持POSIX文件属性视图。示例：rw ——-或rw-rw-rw- etc ...
fileOwner	String	文件所有者定义何时执行操作。由于安全原因，可能会限制更改文件的所有者，并且不允许操作IOException。 如果_POSIX_CHOWN_RESTRICTED对路径有效，则只有具有等效文件用户ID或具有适当权限的有效用户ID的进程才能更改文件的所有权。 底层文件系统应支持文件所有者属性视图。

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

名称	型	
fileGroup	String	用于定义执行操作的文件组。下层文件系统应支持POSIX文件属性视图。

下面是一个示例配置，它使用RollingFileAppender并为当前和滚动日志文件定义不同的POSIX文件属性视图。

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="trace" name="MyApp" packages="">
  <Properties>
    <Property name="baseDir">logs</Property>
  </Properties>
  <Appenders>
    <RollingFile name="RollingFile" fileName="${baseDir}/app.log"
                 filePattern="${baseDir}/$${date:yyyy-MM}/app-%d{yyyyMMdd}.log.gz"
                 filePermissions="rw-------">
      <PatternLayout pattern="%d %p %c{1.} [%t] %m%n" />
      <CronTriggeringPolicy schedule="0 0 0 * * ?"/>
      <DefaultRolloverStrategy stopCustomActionsOnError="true">
        <PosixViewAttribute basePath="${baseDir}/$${date:yyyy-MM}" filePermissions="r--r--r--"
            <IfFileName glob="*.gz" />
        </PosixViewAttribute>
      </DefaultRolloverStrategy>
    </RollingFile>
  </Appenders>

  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingFile"/>
    </Root>
  </Loggers>

</Configuration>
```

RollingRandomAccessFileAppender

RollingRandomAccessFileAppender类似于标准的RollingFileAppender，除了它总是被缓冲（这不能被关闭），并且在内部它使用ByteBuffer + RandomAccessFile而不是BufferedOutputStream。与RollingFileAppender相比，我们在测量中看到“bufferedIO = true”，性能提升了20-200%。 RollingRandomAccessFileAppender写入fileName参数中指定的文件，并根据TriggeringPolicy和RolloverPolicy滚动文件。 与RollingFileAppender类似，RollingRandomAccessFileAppender使用RollingRandomAccessFileManager实际执行文件I / O并执行翻转。 虽然无法共享来自不同Configuration的RollingRandomAccessFileAppender，但RollingRandomAccessFileManagers可以是Manager可访问的。 例如，servlet容器中的两个Web应用程序可以拥有自己的配置，并且如果Log4j位于两个共用的ClassLoader中，则可以安全地写入同一文件。

RollingRandomAccessFileAppender需要TriggeringPolicy和RolloverStrategy。 触发策略确定在RolloverStrategy定义应如何完成翻转时是否应执行翻转。 如果未配置RolloverStrategy，则RollingRandomAccessFileAppender将使用DefaultRolloverStrategy。 从log4j-2.5开始，可以在DefaultRolloverStrategy中配置自定义删除操作以在翻转时运行。

RollingRandomAccessFileAppender不支持文件锁定。

RollingRandomAccessFileAppender参数

推荐阅读
前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946
基于 Nginx+lua+Memcache 实现灰度发布
阅读 793
你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453
Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542
性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

在下喵星人 关注 赞赏支持

名称	型	
append	boolean	如果为true - 默认值，则记录将附加到文件末尾。 设置为false时，将在写入新记录之前清除该文件。
filter	Filter	一个过滤器，用于确定此Appender是否应该处理该事件。 使用CompositeFilter可以使用多个Filter。
fileName	String	要写入的文件的名称。 如果该文件或其任何父目录不存在，则将创建它们。
filePattern	String	存档日志文件的文件名模式。 模式的格式应取决于使用的RolloverPolicy。 DefaultRolloverPolicy将同时接受与SimpleDateFormat兼容的日期/时间模式和/或表示整数计数器的%i。 该模式还支持在运行时进行插值，因此任何 Lookups （例如DateLookup都可以包含在模式中）。
immediateFlush	boolean	设置为true时 - 默认值，每次写入后都会进行刷新。 这将保证数据写入磁盘但可能影响性能。每次写入后刷新仅在将此appender与同步记录器一起使用时才有用。 即使immediateFlush设置为false，异步记录器和追加器也会在一批事件结束时自动刷新。 这也保证了数据被写入磁盘但效率更高。
bufferSize	int	缓冲区大小，默认为262,144字节（256 * 1024）。
layout	Layout	用于格式化LogEvent的布局。 如果未提供布局，则将使用“%m%n”的默认模式布局。
name	String	Appender的名字。
policy	TriggeringPolicy	用于确定是否应发生翻转的策略。
strategy	RolloverStrategy	用于确定存档文件的名称和位置的策略。
ignoreExceptions	boolean	默认值为true，导致在将事件附加到内部记录然后被忽略时遇到异常。 设置为false时，异常将传播给调用者。 将此Appender包装在FailoverAppender中时，必须将此参数设置为false。
filePermissions	String	POSIX格式的文件属性权限，以便在创建文件时应用。 底层文件系统应支持POSIX文件属性视图。示例：rw ——- orrw-rw-rw-etc ...

名称	型	
fileOwner	String	文件所有者在文件创建时定义。出于安全原因，可能会限制更改文件的所有者，并且不允许操作IOException。 只有具有有效用户ID等于文件用户ID或具有适当权限的进程才可以更改文件的所有权if_POSIX_CHOWN_RESTRICTED对路径生效。下行文件系统应支持文件所有者属性视图。
fileGroup	String	用于定义文件创建时的文件组。下层文件系统应支持POSIX文件属性视图。

Triggering Policies

请参见RollingFileAppender触发策略。

Rollover Strategies

请参阅RollingFileAppender翻转策略。

下面是使用RollingRandomAccessFileAppender同时具有基于时间和大小的触发策略的示例配置，将在同一天（1-7）创建最多7个存档，这些存档基于当前年份和月份存储在目录中，并且将 使用gzip压缩每个存档：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <RollingRandomAccessFile name="RollingRandomAccessFile" fileName="logs/app.log"
                 filePattern="logs/$${date:yyyy-MM}/app-%d{MM-dd-yyyy}-%i.log.gz">
      <PatternLayout>
        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
      </PatternLayout>
      <Policies>
        <TimeBasedTriggeringPolicy />
        <SizeBasedTriggeringPolicy size="250 MB"/>
      </Policies>
    </RollingRandomAccessFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingRandomAccessFile"/>
    </Root>
  </Loggers>
</Configuration>
```

第二个示例显示了一个翻转策略，在删除之前最多可保留20个文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <RollingRandomAccessFile name="RollingRandomAccessFile" fileName="logs/app.log"
                 filePattern="logs/$${date:yyyy-MM}/app-%d{MM-dd-yyyy}-%i.log.gz">
      <PatternLayout>
        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
      </PatternLayout>
      <Policies>
        <TimeBasedTriggeringPolicy />
        <SizeBasedTriggeringPolicy size="250 MB"/>
      </Policies>
      <DefaultRolloverStrategy max="20"/>
    </RollingRandomAccessFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingRandomAccessFile"/>
    </Root>
```

在下喵星人

置，将在同一天（1-7）创建最多7个存档，这些存档基于当前年份和月份存储在目录中，并且将 使用gzip压缩每个存档，当小时可以被6整除时，每6个小时滚动一次：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <RollingRandomAccessFile name="RollingRandomAccessFile" fileName="logs/app.log"
                 filePattern="logs/$${date:yyyy-MM}/app-%d{yyyy-MM-dd-HH}-%i.log.gz">
      <PatternLayout>
        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
      </PatternLayout>
      <Policies>
        <TimeBasedTriggeringPolicy interval="6" modulate="true"/>
        <SizeBasedTriggeringPolicy size="250 MB"/>
      </Policies>
    </RollingRandomAccessFile>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="RollingRandomAccessFile"/>
    </Root>
  </Loggers>
</Configuration>
```

RoutingAppender

RoutingAppender评估LogEvents，然后将它们路由到从属Appender。 目标Appender可以是先前配置的appender，可以通过其名称引用，也可以根据需要动态创建Appender。 应该在它引用的任何Appender之后配置RoutingAppender以允许它正确关闭。

您还可以使用脚本配置RoutingAppender：您可以在appender启动时以及为日志事件选择路由时运行脚本。

RoutingAppender参数

参数名称	类型	描述
Filter	Filter	一个过滤器，用于确定此Appender是否应该处理该事件。 使用CompositeFilter可以使用多个Filter。
name	String	Appender的名字。
RewritePolicy	RewritePolicy	RewritePolicy将操纵LogEvent。
Routes	Routes	包含一个或多个Route声明，以标识选择Appender的条件。
Script	Script	当Log4j启动RoutingAppender,此脚本运行,并返回String Route键以确定默认Route时。此脚本传递以下变量：请看下面 RoutingAppender Script参数
ignoreExceptions	boolean	默认值为true，导致在将事件附加到内部记录然后被忽略时遇到异常。 设置为false时，异常将传播给调用者。 将此Appender包装在FailoverAppender中时，必须将此参数设置为false。

RoutingAppender Script参数

参数名称	类型	描述
configura	Configura	

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

staticVariables	Map	此appender实例的所有脚本调用之间共享的Map。 这是传递给Routes Script的相同map。

在此示例中，脚本使“ServiceWindows”路由成为Windows上的默认路由，并使所有其他操作系统上的“ServiceOther”路由成为默认路由。 请注意，List Appender是我们的测试appender之一，可以使用任何appender，它只用作速记。

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN" name="RoutingTest">
  <Appenders>
    <Routing name="Routing">
      <Script name="RoutingInit" language="JavaScript"><![CDATA[
        importPackage(java.lang);
        System.getProperty("os.name").search("Windows") > -1 ? "ServiceWindows" : "ServiceOthe
      </Script>
      <Routes>
        <Route key="ServiceOther">
          <List name="List1" />
        </Route>
        <Route key="ServiceWindows">
          <List name="List2" />
        </Route>
      </Routes>
    </Routing>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="Routing" />
    </Root>
  </Loggers>
</Configuration>
```

Routes

Routes元素接受名为“pattern”的单个属性。 针对所有已注册的查找评估模式，结果用于选择路由。 每个路由可以配置一个密钥。 如果密钥与评估模式的结果匹配，则将选择该路由。 如果路由上未指定密钥，则该路由是默认路由。 只能将一个Route配置为默认值。

Routes元素可以包含Script子元素。 如果指定，则为每个日志事件运行脚本，并返回要使用的String Route键。

您必须指定pattern属性或Script元素，但不能同时指定两者。

每条路线必须引用一个Appender。 如果Route包含ref属性，则Route将引用在配置中定义的Appender。 如果Route包含Appender定义，则将在RoutingAppender的上下文中创建Appender，并且每次通过Route引用匹配的Appender名称时将重用Appender

此脚本传递以下变量：
RoutingAppender Routes Script参数

参数名称	类型	描述
configuration	Configuration	有效的Configuration
staticVariables	Map	此appender实例的所有脚本调用之间共享的Map。 这是传递给Routes Script的相同map。
logEvent	LogEvent	日志事件。

在此示例中，脚本针对每个日志事件运行，并根据名为“AUDIT”的标记的存在来选择路由。

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

在下喵星人

```
      <Agent host="192.168.10.102" port="8800"/>
      <RFC5424Layout enterpriseNumber="18060" includeMDC="true" appName="MyApp"/>
    </Flume>
    <Routing name="Routing">
      <Routes>
        <Script name="RoutingInit" language="JavaScript"><![CDATA[
          if (logEvent.getMarker() != null && logEvent.getMarker().isInstanceOf("AUDIT")) {
                return "AUDIT";
            } else if (logEvent.getContextMap().containsKey("UserId")) {
                return logEvent.getContextMap().get("UserId");
            }
            return "STDOUT";]]>
        </Script>
        <Route>
          <RollingFile
              name="Rolling-${mdc:UserId}"
              fileName="${mdc:UserId}.log"
              filePattern="${mdc:UserId}.%i.log.gz">
            <PatternLayout>
              <pattern>%d %p %c{1.} [%t] %m%n</pattern>
            </PatternLayout>
            <SizeBasedTriggeringPolicy size="500" />
          </RollingFile>
        </Route>
        <Route ref="AuditLogger" key="AUDIT"/>
        <Route ref="STDOUT" key="STDOUT"/>
      </Routes>
      <IdlePurgePolicy timeToLive="15" timeUnit="minutes"/>
    </Routing>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="Routing" />
    </Root>
  </Loggers>
</Configuration>
```

Purge Policy

可以使用PurgePolicy配置RoutingAppender，其目的是停止和删除由RoutingAppender动态创建的休眠Appender。 Log4j目前提供IdlePurgePolicy作为唯一可用于清理Appender的PurgePolicy。 IdlePurgePolicy接受2个属性; timeToLive，即Appender在没有发送任何事件的情况下应该存活的timeUnit的数量，timeUnit是与timeToLive属性一起使用的java.util.concurrent.TimeUnit的String表示。

下面是一个示例配置，它使用RoutingAppender将所有Audit事件路由到FlumeAppender，所有其他事件将路由到仅捕获特定事件类型的RollingFileAppender。 请注意，在根据需要创建RollingFileAppender时预定义了AuditAppender。

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <Flume name="AuditLogger" compress="true">
      <Agent host="192.168.10.101" port="8800"/>
      <Agent host="192.168.10.102" port="8800"/>
      <RFC5424Layout enterpriseNumber="18060" includeMDC="true" appName="MyApp"/>
    </Flume>
    <Routing name="Routing">
      <Routes pattern="$${sd:type}">
        <Route>
          <RollingFile name="Rolling-${sd:type}" fileName="${sd:type}.log"
                       filePattern="${sd:type}.%i.log.gz">
            <PatternLayout>
              <pattern>%d %p %c{1.} [%t] %m%n</pattern>
            </PatternLayout>
            <SizeBasedTriggeringPolicy size="500" />
          </RollingFile>
        </Route>
        <Route ref="AuditLogger" key="Audit"/>
```

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

```
        <AppenderRef ref="Routing"/>
    </Root>
  </Loggers>
</Configuration>
```

SMTPAppender

发生特定日志记录事件时发送电子邮件，通常发生错误或致命错误。

此电子邮件中传递的日志记录事件数取决于BufferSize选项的值。 SMTPAppender仅在其循环缓冲区中保留最后一个BufferSize日志记录事件。 这使内存需求保持在合理的水平，同时仍然提供有用的应用程序上下 缓冲区中的所有事件都包含在电子邮件中。 缓冲区将包含触发电子邮件的事件之前TRACE到WARN级别的最新事件。

默认行为是在记录ERROR或更高严重性事件时触发发送电子邮件，并将其格式化为HTML。 可以通过在Appender上设置一个或多个过滤器来控制发送电子邮件的情况。 与其他Appender一样，可以通过为Appender指定布局来控制格式。

SMTPAppender参数

参数名称	类型	描述
name	String	Appender的名字。
from	String	发件人的电子邮件地址。
replyTo	String	以逗号分隔的回复电子邮件地址列表。
to	String	逗号分隔的收件人电子邮件地址列表。
cc	String	以逗号分隔的CC电子邮件地址列表。
bcc	String	以逗号分隔的BCC电子邮件地址列表。
subject	String	电子邮件的主题。
bufferSize	integer	要缓冲以包含在消息中的最大日志事件数。 默认为512。
layou	Ly	

filter	Filter	一个过滤器，用于确定此Appender是否应该处理该事件。 使用CompositeFilter可以使用多个Filter。
smtpDebug	boolean	设置为true时，在STDOUT上启用会话调试。 默认为false。
smtpHost	String	要发送到的SMTP主机名。 此参数是必需的。
smtpPassword	String	对SMTP服务器进行身份验证所需的密码。
smtpPort	integer	要发送到的SMTP端口。
smtpProtocol	String	SMTP传输协议（例如“smtps”，默认为“smtp”）。
smtpUsername	String	对SMTP服务器进行身份验证所需的用户名。
ignoreExceptions	boolean	默认值为true，导致在将事件附加到内部记录然后被忽略时遇到异常。 设置为false时，异常将传播给调用者。 将此Appender包装在FailoverAppender中时，必须将此参数设置为false。

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <SMTP name="Mail" subject="Error Log" to="errors@logging.apache.org" from="test@logging.ap
          smtpHost="localhost" smtpPort="25" bufferSize="50">
    </SMTP>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="Mail"/>
    </Root>
  </Loggers>
</Configuration>
```

ScriptAppenderSelector

构建配置时，ScriptAppenderSelector appender调用脚本来计算appender名称。 然后，Log4j使用ScriptAppenderSelector的名称创建AppenderSet下列出的一个appender。 配置完成后，Log4j会忽略ScriptAppenderSelector。 Log4j仅从配置树构建一个选定的appender，并忽略其他AppenderSet子节点。

在以下示例中，脚本返回名称“List2”。 appender名称记录在ScriptAppenderSelector的名称下，而不是所选appender的名称。 在本例中为“SelectIt”

```
    <ScriptAppenderSelector name="SelectIt">
      <Script language="JavaScript"><![CDATA[
        importPackage(java.lang);
        System.getProperty("os.name").search("Windows") > -1 ? "MyCustomWindowsAppender" : "My
      </Script>
      <AppenderSet>
        <MyCustomWindowsAppender name="MyAppender" ... />
        <SyslogAppender name="MySyslog" ... />
      </AppenderSet>
    </ScriptAppenderSelector>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="SelectIt" />
    </Root>
  </Loggers>
</Configuration>
```

SocketAppender

SocketAppender是一个OutputStreamAppender，它将其输出写入由主机和端口指定的远程目标。 数据可以通过TCP或UDP发送，并可以任何格式发送。 您可以选择使用SSL保护通信。

参数名称	类型	描述
name	String	Appender的名字。
host	String	正在侦听日志事件的系统的名称或地址。 此参数是必需的。
port	integer	正在侦听日志事件的主机上的端口。 必须指定此参数。
protocol	String	“TCP”（默认），“SSL”或“UDP”。
SSL	SslConfiguration	包含KeyStore和TrustStore的配置。 请参阅SSL。
filter	Filter	一个过滤器，用于确定此Appender是否应该处理该事件。 使用CompositeFilter可以使用多个Filter。
immediateFail	boolean	设置为true时，日志事件将不会等待尝试重新连接，如果套接字不可用，则会立即失败。
immediateFlush	boolean	设置为true时 - 默认值，每次写入后都会进行刷新。 这将保证数据写入磁盘，但可能会影响性能。
bufferedIO	boolean	如果为true - 默认情况下，事件将写入缓冲区，当缓冲区已满时，数据将写入套接字;如果设置了immediateFlush，则写入记录时。
bufferSize	int	当bufferedIO为true时，这是缓冲区大小，默认为8192字节。
layout	Layout	用于格式化LogEvent的布局。 必需，没有默认值。 自2.9以来的新版本，在以前的版本中，SerializedLayout是默认的。
reconnectionDelayMillis	integer	如果设置为大于0的值，则在发生错误后，SocketManager将在等待指定的毫秒数后尝试重新连接到服务器。 如果重新连接失败，则抛出异常（如果ignoreExceptions设置为false，则应用程序可以捕获该异常）。
connectTimeo	inte	连接超时（以毫秒为单位）。 默认值为0（无限超时，如

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

ignoreExceptions	boolean	默认的istrue，导致在将事件附加到内部记录然后被忽略时遇到异常。 设置为false时，异常将传播给调用者。 将此Appender包装在FailoverAppender中时，必须将此参数设置为false。

这是一个不安全的TCP配置：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <Socket name="socket" host="localhost" port="9500">
      <JsonLayout properties="true"/>
    </Socket>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="socket"/>
    </Root>
  </Loggers>
</Configuration>
```

这是一个安全的SSL配置：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <Socket name="socket" host="localhost" port="9500">
      <JsonLayout properties="true"/>
      <SSL>
        <KeyStore   location="log4j2-keystore.jks" passwordEnvironmentVariable="KEYSTORE_PASSW
        <TrustStore location="truststore.jks"      passwordFile="${sys:user.home}/truststore.p
      </SSL>
    </Socket>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="socket"/>
    </Root>
  </Loggers>
</Configuration>
```

SSL Configuration

可以将多个appender配置为使用普通网络连接或安全套接字层（SSL）连接。 本节介绍可用于SSL配置的参数。

SSL配置参数

参数名称	类型	描述
protocol	String	SSL如果省略。 另请参见标准名称。
KeyStore	KeyStore	包含您的私钥和证书，并确定要发送到远程主机的身份验证凭据。
TrustStore	TrustStore	包含远程交易对手的CA证书。 确定是否应该信任远程身份验证凭据（以及连接）。

KeyStore

KeyStore旨在包含您的私钥和证书，并确定要发送到远程主机的身份验证凭据。

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

location	String	keystore 文件的路径。
password	char[]	用于访问keystore的纯文本密码。 不能与passwordEnvironmentVariable或passwordFile结合使用。
passwordEnvironmentVariable	String	保存密码的环境变量的名称。 不能与密码或密码文件组合使用。
passwordFile	String	保存密码的文件的路径。 不能与密码或passwordEnvironmentVariable结合使用。
type	String	可选的KeyStore类型，例如JKS，PKCS12，PKCS11，BKS，Windows-MY / Windows-ROOT，KeychainStore等。默认为JKS。 另请参见标准类型。
keyManagerFactoryAlgorithm	String	可选的KeyManagerFactory算法。 默认为SunX509。 另请参见标准算法。

TrustStore

信任存储区旨在包含您在远程方提供其证书时愿意信任的CA证书。 确定是否应该信任远程身份验证凭据（以及连接）。

在某些情况下，它们可以是同一个商店，尽管使用不同的商店通常是更好的做法（特别是当它们是基于文件的时候）。

TrustStore配置参数

参数名称	类型	描述
location	String	keystore 文件的路径。
password	char[]	用于访问keystore的纯文本密码。 不能与passwordEnvironmentVariable或passwordFile结合使用。
passwordEnvironmentVariable	String	保存密码的环境变量的名称。 不能与密码或密码文件组合使用。
passwordFile	String	保存密码的文件的路径。 不能与密码或passwordEnvironmentVariable结合使用。
type	String	可选的KeyStore类型，例如JKS，PKCS12，PKCS11，BKS，Windows-MY / Windows-ROOT，KeychainStore等。默认为JKS。 另请参见标准类型。

trustManagerFactoryAlgorithm	String	可选的TrustManagerFactory算法。 默认为SunX509。 另请参见标准算法。

例如:

```
...
      <SSL>
        <KeyStore   location="log4j2-keystore.jks" passwordEnvironmentVariable="KEYSTORE_PASSW
        <TrustStore location="truststore.jks"      passwordFile="${sys:user.home}/truststore.p
      </SSL>
  ...
```

SyslogAppender

SyslogAppender是一个SocketAppender，它将其输出写入主机和端口指定的远程目标，格式符合BSD Syslog格式或RFC 5424格式。 数据可以通过TCP或UDP发送。

SyslogAppender参数

参数名称	类型	描述
advertise	boolean	指示是否应该通告appender。
appName	String	在RFC 5424 syslog记录中用作APP-NAME的值。
charset	String	将syslog String转换为字节数组时使用的字符集。 String必须是有效的Charset。 如果未指定，将使用默认系统Charset。
connectTimeoutMillis	integer	连接超时（以毫秒为单位）。 默认值为0（无限超时，如Socket.connect（ ）方法）。
enterpriseNumber	integer	RFC 5424中描述的IANA企业编号
filter	Filter	一个过滤器，用于确定此Appender是否应该处理该事件。 使用CompositeFilter可以使用多个Filter。
facility	String	该工具用于尝试对消息进行分类。 设施选项必须设置为“KERN”，“USER”，“MAIL”，“DAEMON”，“AUTH”，“SYSLOG”，“LPR”，“NEWS”，“UUCP”，“CRON”，“AUTHPRIV”，“FTP”，“NTP”，“AUDIT”，“ALERT”，“CLOCK”，“LOCAL0”，“LOCAL1”，“LOCAL2”，“LOCAL3”，“LOCAL4”，“LOCAL5”，“LOCAL6”，或“LOCAL7”。 这些值可以指定为大写或小写字符。

推荐阅读
前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946
基于 Nginx+lua+Memcache 实现灰度发布
阅读 793
你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453
Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542
性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

名称	型	
format	String	如果设置为“RFC5424”，则数据将根据RFC 5424进行格式化。否则，它将被格式化为BSD Syslog记录。 请注意，虽然BSD Syslog记录要求为1024字节或更短，但SyslogLayout不会截断它们。 RFC5424Layout也不会截断记录，因为接收方必须接受最多2048字节的记录，并且可以接受更长的记录。
host	String	正在侦听日志事件的系统的名称或地址。 此参数是必需的。
id	String	根据RFC 5424进行格式化时使用的默认结构化数据ID。如果LogEvent包含StructuredDataMessage，将使用Message中的id而不是此值。
ignoreExceptions	boolean	默认值为true，导致在将事件附加到内部记录然后被忽略时遇到异常。 设置为false时，异常将传播给调用者。 将此Appender包装在FailoverAppender中时，必须将此参数设置为false。
immediateFail	boolean	设置为true时，日志事件将不会等待尝试重新连接，如果套接字不可用，则会立即失败
immediateFlush	boolean	设置为true时 - 默认值，每次写入后都会进行刷新。 这将保证数据写入磁盘，但可能会影响性能。
includeMDC	boolean	指示来自ThreadContextMap的数据是否将包含在RFC 5424 Syslog记录中。 默认为true。
Layout	Layout	自定义布局，覆盖格式设置。
loggerFields	List of KeyValuePairs	允许任意PatternLayout模式包含在指定的ThreadContext字段中; 没有指定默认值。 要使用，请包含> LoggerFields <嵌套元素，其中包含一个或多个> KeyValuePair <元素。 每个> KeyValuePair <必须具有key属性，该属性指定将用于标识MDC Structured Data元素中的字段的键名称，以及value属性，该属性指定要用作值的PatternLayout模式。
mdcExcludes	String	以逗号分隔的mdc键列表，应从LogEvent中排除。 这与mdcIncludes属性互斥。 此属性仅适用于RFC 5424 syslog记录。
mdcIncludes	String	以逗号分隔的mdc键列表，应包含在FlumeEvent中。 将排除列表中未找到的MDC中的任何键。 此选项与mdcExcludes属性互斥。 此属性仅适用于RFC 5424 syslog记录。
mdcRequired	String	以逗号分隔的mdc键列表，必须存在于MDC中。 如果没有键，则抛出LoggingException。 此属性仅适用于RFC 5424 syslog记录。
mdcPrefix	String	应添加到每个MDC密钥的字符串，以便将其与事件属性区分开来。 默认字符串是“mdc：”。 此属性仅适用于RFC 5424 syslog记录。

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

名称	型	
messageId	String	RFC 5424 syslog记录的MSGID字段中使用的默认值。
name	String	Appender的名字。
newLine	boolean	如果为true，则会将新行附加到syslog记录的末尾。 默认值为false。
port	integer	正在侦听日志事件的主机上的端口。 必须指定此参数。
protocol	String	“TCP”或“UDP”。 此参数是必需的。
SSL	SslConfiguration	包含KeyStore和TrustStore的配置。 请参阅SSL。
reconnectionDelayMillis	integer	如果设置为大于0的值，则在发生错误后，SocketManager将在等待指定的毫秒数后尝试重新连接到服务器。 如果重新连接失败，则抛出异常（如果ignoreExceptions设置为false，则应用程序可以捕获该异常）。

示例syslogAppender配置，配置有两个SyslogAppender，一个使用BSD格式，另一个使用RFC 5424。

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
  <Appenders>
    <Syslog name="bsd" host="localhost" port="514" protocol="TCP"/>
    <Syslog name="RFC5424" format="RFC5424" host="localhost" port="8514"
            protocol="TCP" appName="MyApp" includeMDC="true"
            facility="LOCAL0" enterpriseNumber="18060" newLine="true"
            messageId="Audit" id="App"/>
  </Appenders>
  <Loggers>
    <Logger name="com.mycorp" level="error">
      <AppenderRef ref="RFC5424"/>
    </Logger>
    <Root level="error">
      <AppenderRef ref="bsd"/>
    </Root>
  </Loggers>
</Configuration>
```

对于SSL，此appender将其输出写入由主机指定的远程目标和SSL上的端口，格式符合BSD Syslog格式或RFC 5424格式。

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

在下喵星人

```
      </SSL>
    </TLSSyslog>
  </Appenders>
  <Loggers>
    <Root level="error">
      <AppenderRef ref="bsd"/>
    </Root>
  </Loggers>
</Configuration>
```

ZeroMQ/JeroMQ Appender

ZeroMQ appender使用JeroMQ库将日志事件发送到一个或多个ZeroMQ端点。

这是一个简单的JeroMQ配置

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration name="JeroMQAppenderTest" status="TRACE">
  <Appenders>
    <JeroMQ name="JeroMQAppender">
      <Property name="endpoint">tcp://*:5556</Property>
      <Property name="endpoint">ipc://info-topic</Property>
    </JeroMQ>
  </Appenders>
  <Loggers>
    <Root level="info">
      <AppenderRef ref="JeroMQAppender"/>
    </Root>
  </Loggers>
</Configuration>
```

下表描述了所有选项。 有关详细信息，请参阅JeroMQ和ZeroMQ文档。

JeroMQ参数

参数名称	类型	描述
name	String	Appender的名字。 必填。
Layout	layout	用于格式化LogEvent的布局。 如果未提供布局，则将使用默认的模式布局“%m%n”。
Filters	Filter	Appender的过滤器。
Properties	Property[]	一个或多个Property元素，名为endpoint。
ignoreExceptions	boolean	如果为true，则将记录并禁止异常。 如果将记录错误错误，然后将其传递给应用程序。
affinity	long	ZMQ_AFFINITY选项。 默认为0。
backlog	long	ZMQ_BACKLOG选项。 默认为100。
delayAttachOnConnect	boolean	ZMQ_DELAY_ATTACH_ON_CONNECT选项。 默认为false。
identity	byte[]	ZMQ_IDENTITY选项。 默认为none。
ipv4Only	boolean	ZMQ_IPV4ONLY选项。 默认为true。
linger	long	ZMQ_LINGER选项。 默认为-1。
maxMsgSize	long	ZMQ_MAXMSGSIZE选项。 默认为-1。
rcvHwm	long	ZMQ_RCVHWM选项。 默认为1000。

推荐阅读

前端vue中防止用户在短时间内频繁多次点击按钮
阅读 2,946

基于 Nginx+lua+Memcache 实现灰度发布
阅读 793

你居然还去服务器上捞日志，搭个日志收集系统难道不香么
阅读 5,453

Nginx 实现 10万+ 并发，Linux 内核优化
阅读 1,542

性能优越的轻量级日志收集工具，微软、亚马逊都在用！
阅读 1,485

receiveBufferSize	long	ZMQ_RCVBUF选项。 默认为0。
receiveTimeOut	int	ZMQ_RCVTIMEO选项。 默认为-1。
reconnectIVL	long	ZMQ_RECONNECT_IVL选项。 默认为100。
reconnectIVLMax	long	ZMQ_RECONNECT_IVL_MAX选项。 默认为0。
sendBufferSize	long	ZMQ_SNDBUF选项。 默认为0。
sendTimeOut	int	ZMQ_SNDTIMEO选项。 默认为-1。
sndHwm	long	ZMQ_SNDHWM选项。 默认为1000。
tcpKeepAlive	int	ZMQ_TCP_KEEPALIVE选项。 默认为-1。
tcpKeepAliveCount	long	ZMQ_TCP_KEEPALIVE_CNT选项。 默认为-1。
tcpKeepAliveIdle	long	ZMQ_TCP_KEEPALIVE_IDLE选项。 默认为-1。
tcpKeepAliveInterval	long	ZMQ_TCP_KEEPALIVE_INTVL选项。 默认为-1。
xpubVerbose	boolean	ZMQ_XPUB_VERBOSE选项。 默认为false。

log4j2使用手册（中文）第九章 Appenders （二）

在下喵星人 关注 赞赏支持

log4j2使用手册（中文）第九章 Appenders（一）

Appender负责将LogEvents传递到目的地。 每个Appender都必须实现Appender接口。 大多...

在下喵星人 阅读 219 评论 0 赞 1

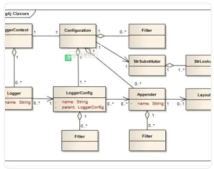
Log4J日志配置详解（转）

from：https://www.cnblogs.com/ITtangtang/p/3926665.html一、L...

enshunyan 阅读 1,601 评论 0 赞 0

log4j2使用手册（中文）第二章 架构

架构 架构 主要组件 Log4J类结构如下图所示: 使用Log4j 2 API的应用程序将从LogManager请...



在下喵星人 阅读 401 评论 0 赞 0