自：https://www.cnblogs.com/morvenhuang/p/3958086.html

[**Log4j2常见使用示例及Syslog/Syslog-ng**](https://www.cnblogs.com/morvenhuang/p/3958086.html)

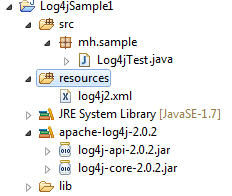
[**Log4j2常见使用示例及Syslog/Syslog-ng**](https://www.cnblogs.com/morvenhuang/p/3958086.html)

**准备工作**

打开<http://logging.apache.org/log4j/>，点击左侧Download，我下载的是Apache Log4j 2 binary (zip)，目前是2.0.2版本。解压后有30几个jar包，大部分是跟兼容性及移植性相关的可选组件，我们要用的是：

log4j-api-2.0.2.jar

log4j-core-2.0.2.jar

**第一个示例程序**

**log4j2.xml**

[复制代码](javascript:void(0);)

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

2 <Configuration status="warn">

3 <Appenders>

4 <Console name="Console" target="SYSTEM\_OUT">

5 <PatternLayout pattern="%m%n" />

6 </Console>

7 </Appenders>

8 <Loggers>

9 <Root level="INFO">

10 <AppenderRef ref="Console" />

11 </Root>

12 </Loggers>

13 </Configuration>

[复制代码](javascript:void(0);)

解释一下Configuration后面的status，这个用于设置log4j2自身内部的信息输出，可以不设置，当设置成trace时，你会看到log4j2内部各种详细输出。

log4j2配置文件可以使用XML或JSON，似乎 不再支持properties文件了。默认的文件名也有所不同，log4j2.xml，不再是log4j.xml。

log4j2.xml可以放在任意的地方，只要你最后把它放到了classpath里，上面的项目中新建一个resources目录用于放置log4j2.xml，如果在未加入classpath时尝试运行时会报如下错误：

ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.

以Eclipse环境为例，可以在Run—Run Configurations对应项目的Classpath中选择User Entries，点击Advanced，选择Add Folder，把resources文件夹添加进来。

**Log4jTest.java**

[复制代码](javascript:void(0);)

1 package mh.sample;

2

3 import org.apache.logging.log4j.LogManager;

4 import org.apache.logging.log4j.Logger;

5

6 public class Log4jTest {

7 static Logger log = LogManager.getLogger( Log4jTest.class.getName());

8

9 public static void main(String[] args) {

10 log.info("This is an info log.");

11 log.warn("This is a warn log.");

12 }

13 }

[复制代码](javascript:void(0);)

需要注意的是，log4j2中要用LogManager.get logger，而不是以前的Logger.getLogger

运行此类，可以看到控制台输出两条日志信息：

This is an info log.

This is a warn log.

 通过将配置文件中的level改为”WARN”，可以看到只有一条日志了：

This is a warn log.

关于level的事就这么简单，不多解释了。

关于pattern，详见：http://logging.apache.org/log4j/2.x/manual/layouts.html#PatternLayout

比如常用的是

[%-5p] %d %c - %m%n

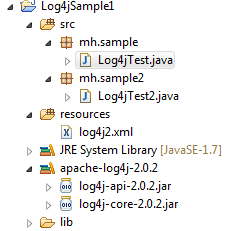
如果使用此pattern，输出将类似如下：

[INFO ] 2014-08-29 16:18:15,066 mh.sample.Log4jTest - This is an info log.

[WARN ] 2014-08-29 16:18:15,069 mh.sample.Log4jTest - This is a warn log.

5表示自动将level级别补空格至5个字符

**第二个示例程序**



**Log4j2.xml**

[复制代码](javascript:void(0);)

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

2 <Configuration status="warn">

3 <Appenders>

4 <Console name="Console" target="SYSTEM\_OUT">

5 <PatternLayout pattern="[%-5p] %d %c - %m%n" />

6 </Console>

7 <File name="File" fileName="dist/my.log">

8 <PatternLayout pattern="%m%n" />

9 </File>

10 </Appenders>

11

12 <Loggers>

13 <Logger name="mh.sample2.Log4jTest2" level="INFO">

14 <AppenderRef ref="File" />

15 </Logger>

16 <Root level="INFO">

17 <AppenderRef ref="Console" />

18 </Root>

19 </Loggers>

20 </Configuration>

[复制代码](javascript:void(0);)

在配置文件中我新加了一个Appender，

<File name="File" fileName="dist/my.log">

    <PatternLayout pattern="%m%n" />

</File>

另外，新加了一个Logger，

<Logger name="mh.sample2.Log4jTest2" level="INFO">

    <AppenderRef ref="File" />

</Logger>

**Log4jTest.java**

[复制代码](javascript:void(0);)

public class Log4jTest {

static Logger log = LogManager.getLogger( Log4jTest.class.getName());

public static void main(String[] args) {

log.info("This is an info log.");

log.warn("This is a warn log.");

//

Log4jTest2 lt2 = new Log4jTest2();

lt2.printLog();

}

}

[复制代码](javascript:void(0);)

**Log4jTest2.java**

[复制代码](javascript:void(0);)

1 public class Log4jTest2 {

2

3 static Logger log = LogManager.getLogger( Log4jTest2.class.getName());

4

5 public void printLog(){

6 log.info("This is an info log from Log4jTest2.");

7 log.warn("This is a warn log from Log4jTest2.");

8 }

9 }

[复制代码](javascript:void(0);)

运行起来，控制台的输出是，

[INFO ] 2014-08-29 17:16:39,899 mh.sample.Log4jTest - This is an info log.

[WARN ] 2014-08-29 17:16:39,901 mh.sample.Log4jTest - This is a warn log.

[INFO ] 2014-08-29 17:16:39,903 mh.sample2.Log4jTest2 - This is an info log from Log4jTest2.

[WARN ] 2014-08-29 17:16:39,904 mh.sample2.Log4jTest2 - This a is warn log from Log4jTest2.

项目下dist/my.log文件中的内容是，

This is an info log from Log4jTest2.

This is a warn log from Log4jTest2.

可以看到，Root总是输出所有。而新添加的logger负责把Log4jTest2中的日志写到指定文件。这个特性有利于在比较大的项目中对日志进行分类。

需要注意的一点是，log4j存在一个“反直觉”的行为，尝试把Root的level改为“WARN”，重新运行，控制台的输出是：

[WARN ] 2014-08-29 17:24:06,788 mh.sample.Log4jTest - This is a warn log.

[INFO ] 2014-08-29 17:24:06,793 mh.sample2.Log4jTest2 - This is an info log from Log4jTest2.

[WARN ] 2014-08-29 17:24:06,794 mh.sample2.Log4jTest2 - This is a warn log from Log4jTest2.

Log4jTest2中的info级别的log还是被输出了，也就是，只要某个logger接受了该log请求，那为作为父亲的Root就会跟着接受此log请求，而不再考虑它的level

**顺带介绍additivity属性，**

<Logger name="mh.sample2.Log4jTest2" level="INFO" additivity="false">

作了上面的修改，再运行程序，控制台的输出：

[WARN ] 2014-08-29 17:31:17,388 mh.sample.Log4jTest - This is a warn log.

所以additivity实际上中止了log请求向上级传播，这导致了Log4jTest2中的日志被Logger name="mh.sample2.Log4jTest2" “截留”。

**常用Appender之RollingFile**

**Log4j2.xml**

[复制代码](javascript:void(0);)

<Configuration status="warn">

<Appenders>

<Console name="Console" target="SYSTEM\_OUT">

<PatternLayout pattern="[%-5p] %d %c - %m%n" />

</Console>

<File name="File" fileName="dist/my.log">

<PatternLayout pattern="%m%n" />

</File>

<RollingFile name="RollingFile" fileName="dist/my2.log"

filePattern="dist/$${date:yyyy-MM}/app-%d{MM-dd-yyyy}-%i.log.gz">

<PatternLayout pattern="[%-5p] %d %c - %m%n" />

<Policies>

<TimeBasedTriggeringPolicy />

<SizeBasedTriggeringPolicy size="25 KB" />

</Policies>

<DefaultRolloverStrategy max="20"/>

</RollingFile>

</Appenders>

<Loggers>

<Logger name="mh.sample2.Log4jTest2" level="INFO" additivity="false">

<AppenderRef ref="File" />

<AppenderRef ref="RollingFile" />

</Logger>

<Root level="WARN">

<AppenderRef ref="Console" />

</Root>

</Loggers>

</Configuration>

[复制代码](javascript:void(0);)

上述RollingFile的配置将把日志内容**追加**到dist/my2.log文件中，每当大小达到设定的25KB，就会按filePattern的规则备份到dist目录下的子目录中，由于当前是2014年9月3号，该子目录名称是2014-09，里面的文件则是app-09-03-2014-1.log.gz，app-09-03-2014-2.log.gz，app-09-03-2014-3.log.gz，……，从这还可以看出，log4j2自动按你配置的文件名进行gzip压缩。

**DefaultRolloverStrategy**属性如不设置，则默认为最多同一文件夹下7个文件，这里设置了20.

**常用Appender之Syslog**

log4j2中对syslog的简单配置，这里就不重复展示log4j2.xml了：

<Syslog name="SYSLOG" host="localhost" port="514" protocol="UDP" facility="LOCAL3"/>

host是指你将要把日志写到的目标机器，可以是ip（本地ip或远程ip，远程ip在实际项目中很常见，有专门的日志服务器来存储日志），也可以使用主机名，如果是本地，还可以使用localhost或127.0.0.1

Port指定端口，默认514，参见/etc/rsyslog.conf（以Fedora系统为例，下同）。protocol指定传输协议，这里是UDP，facility是可选项，后面可以看到用法，具体关于facility的规则可以点击：<http://logging.apache.org/log4j/2.x/manual/appenders.html#SyslogAppender>

**Syslog及Syslog-ng相关配置（Fedora）**

在运行程序之前，需要修改

/etc/rsyslog.conf

把这两行前的#去掉，即取消注释：

#$ModLoad imudp

#$UDPServerRun 514

这里启用udp监听，514是默认监听端口，重启syslog：

service syslog restart

大部分日志会默认写到/var/log/messages中，如果不想写到这个文件里，可以按下面修改，这样local3的日志就会写到app.log中。这里的local3即 log4j2.xml中facility的配置。

\*.info;mail.none;authpriv.none;cron.none;local3.none                /var/log/messages

新增一行

local3.\*                                                                                 /var/log/app.log

除了使用自带的syslog，我们也可以使用syslog的替代品，比如syslog-ng，这对于log4j2.xml配置没有影响。

安装：

yum install syslog-ng

启动：

service syslog-ng start

其配置文件为

/etc/syslog-ng/syslog-ng.conf

启动前把source一节中这一行取消注释即可：

#udp(ip(0.0.0.0) port(514));

这个端口会和syslog冲突，可以使用别的端口比如50014，同时修改log4j2.xml中的port属性。另外提一下，使用非默认端口，要求log4j版本在1.2.15或以上。

syslog-ng本身也可以设置把日志送到远程机器上，

在源机器上的syslog-ng.conf中添加：

destination d\_remote1 { udp(153.65.171.73 port(514));};

这表示源机器上的syslog-ng会把接收到的日志送到远程主机153.65.171.73的514端口上，只要保证153.65.171.73上的syslog-ng正常运行并监听对应的端口即可。