

# 实现带有maxBackupIndex属性的DailyRollingFileAppender

## 前言

如果你尝试使用Apache Log4J中的DailyRollingFileAppender来打印每天的日志，你可能想对那些日志文件指定一个最大的保存数，就像RollingFileAppender支持maxBackupIndex参数一样。不过遗憾的是，目前版本的Log4j (Apache log4j 1.2.17)无法在使用DailyRollingFileAppender的时候指定保存文件的个数，本文将介绍如何修改DailyRollingFileAppender类，使得它支持maxBackupIndex属性，所以能够删除那些以后不会再使用到的日志文件。

## 涉及到的修改

1、为了不影响自带的DailyRollingFileAppender类，我们首先将DailyRollingFileAppender类重命名为CustomDailyRollingFileAppender，并将它存放到不同的包中，比如com.iteblog.log4j.appenders

2、增加一个新的字段protected int maxBackupIndex = 1，我们将maxBackupIndex字段的默认值设置为1，之所以添加这个字段是因为我们的appender通过maxBackupIndex参数的值来决定需要保存耳朵文件数。

3、我们对maxBackupIndex字段创建setter和getter方法：

```
public int getMaxBackupIndex() {  
    return maxBackupIndex;  
}  
  
public void setMaxBackupIndex(int maxBackups) {  
    this.maxBackupIndex = maxBackups;  
}
```

4、创建一个新的类ModifiedTimeSortableFile，这个类扩展自java.io.File类并且实现了java.lang.Comparable接口，以便根据文件的修改事件来对文件列表进行排序。之所以需要创建这个类是因为我们后面会调用Collections.sort()函数来对已经保存的日志文件按照最后修改时间进行排序，而sort需要使用到public int compareTo(File anotherPathName)方法。具体实现如下：

```
class ModifiedTimeSortableFile extends File implements Serializable, Comparable<File>  
{  
    private static final long serialVersionUID = 1373373728209668895L;  
  
    public ModifiedTimeSortableFile(String parent, String child) {
```

```
super(parent, child);
}

public ModifiedTimeSortableFile(URI uri) {
    super(uri);
}

public ModifiedTimeSortableFile(File parent, String child) {
    super(parent, child);
}

public ModifiedTimeSortableFile(String string) {
    super(string);
}

public int compareTo(File anotherPathName) {
    long thisVal = this.lastModified();
    long anotherVal = anotherPathName.lastModified();
    return (thisVal<anotherVal ? -1 : (thisVal==anotherVal ? 0 : 1));
}
}
```

5、创建完ModifiedTimeSortableFile类之后，我们可以创建名为private List getAllFiles()的方法了，这个方法是根据log4j配置的模式来匹配日志文件列表，并返回list：

```
private List<ModifiedTimeSortableFile> getAllFiles(){
    List<ModifiedTimeSortableFile> files = new ArrayList<ModifiedTimeSortableFile>();
    FilenameFilter filter = new FilenameFilter() {
        public boolean accept(File dir, String name) {
            String directoryName = dir.getPath();
            LogLog.debug("directory name: " + directoryName);
            File file = new File(fileName);
            String parentDirectory = file.getParent();
            if(parentDirectory != null)
            {
                String localFile = fileName.substring(directoryName.length());
                return name.startsWith(localFile);
            }
            return name.startsWith(fileName);
        }
    };
    File file = new File(fileName);
    String parentDirectory = file.getParent();
```

```
if(file.exists())
{
    if(file.getParent() == null){
        String absolutePath = file.getAbsolutePath();
        perentDirectory = absolutePath.substring(0,
        absolutePath.lastIndexOf(fileName));
    }
}
File dir = new File(perentDirectory);
String[] names = dir.list(filter);

for (int i = 0 ; i < names.length ; i++) {
    files.add(new ModifiedTimeSortableFile
(dir + System.getProperty("file.separator") + names[i]));
}
return files;
}
```

6、void rollOver() throws IOException方法的职责是将当前的文件切割成一个新的文件。我们可以在rollOver方法最开始加入一些其他的逻辑。我们的逻辑只是简单地获取已经保存的日志文件列表，并按照这些日志文件的最后修改事件进行排序，然后根据maxBackupIndex属性决定哪些最久的日志文件是需要删除的：

```
List<ModifiedTimeSortableFile> files = getAllFiles();
Collections.sort(files)
if(files.size() >= maxBackupIndex)
{
    int index = 0;
    int diff = files.size() - (maxBackupIndex - 1);
    for(ModifiedTimeSortableFile file : files)
    {
        if(index >= diff)
            break;

        file.delete();
        index++;
    }
}
```

## 如何使用

现在我们已经创建好CustomDailyRollingFileAppender类了，现在是时候来测试我们的Appender了。仅仅需要下面简单的两步即可。

1、创建一个log4j配置文件，并设置好datePattern, maxBackupSize等相关参数，如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd" >
<log4j:configuration>
    <appender name="FILE" class="com.iteblog.log4j.appenders.CustomDailyRollingFileAppender">
        <param name="file" value="test-agent.log" />
        <param name="datePattern" value="_dd-yyyy-MM.log" />
        <param name="maxBackupIndex" value="4" />
        <param name="append" value="true" />
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%d [%t] %p - %m%n" />
        </layout>
    </appender>
    <root>
        <priority value="info" />
        <appender-ref ref="FILE" />
    </root>
</log4j:configuration>
```

可以看到我们已经使用了前面定义的com.iteblog.log4j.appenders.CustomDailyRollingFileAppender类，并且将maxBackupIndex设置成4，也就是说我们最多只保存4个日志文件，

2、创建一个测试类，并将之前创建的CustomDailyRollingFileAppender类加入到这个测试类的环境变量中：

```
public class Main {
    private static org.apache.log4j.Logger log = Logger.getLogger(Main.class);
    public static void main(String[] args) {
        String configFile;
        final File file = new File("log4j.xml");
        if (file.exists()) {
            final URL url = Main.class.getClassLoader().getResource("log4j.xml");
            configFile = url.getPath();
            PropertyConfigurator.configure("log4j.xml");
        }
    }
}
```

```
}

log.trace("Trace");
log.debug("Debug");
log.info("Info");
log.warn("Warn");
log.error("Error");
log.fatal("Fatal");

}
```

然后我们就可以测试我们的系统是不是就只保存最近4个日志文件了。

本博客文章除特别声明，全部都是原创！  
原创文章版权归过往记忆大数据（过往记忆）所有，未经许可不得转载。  
本文链接: 【】()