在java项目中使用log4j2

作者：梁志艳
email：liangzhiyan@kyee.com.cn

前言

在java项目中我们经常要使用Log记录日志信息，做调试用。当然有些情况我们直接使用System.out.println()来做同样的事情。但是这样如果产品发布的时候我们就要一个个手动删除这些记录日志的语句。 来自Apache的Log4j项目就解决了这个问题。目前Log4j最新版到了2.x版本，但是网上很多教程还在介绍如何使用Log4j。本文主要是介绍一下如何在项目中使用log4j2。

依赖

在项目的pom文件中添加以下依赖：

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.1</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.1</version>
</dependency>
```

Logger

在类中使用private static final的修饰符生命Logger对象。

```
private static final Logger logger = LogManager.getLogg
```

然后就可以使用 `logger.debug()` 、 `logger.error()` 等方法记录log了。 详细的logger api请参考：官方文档

XML

在项目的classpath目录中新建log4j2.xml文件，maven项目中是src/main/resources目录。
请如果项目中存在其他log4j的xml配置文件，log4j将会按照一定的顺序覆盖原来的配置。加载的顺序如下：

> 1. Log4j will inspect the "log4j.configurationFile" system property and, if set, will attempt to load the configuration using the ConfigurationFactory that matches the file extension.
> 2. If no system property is set the YAML ConfigurationFactory will look for log4j2-test.yaml or log4j2-test.yml in the classpath.
> 3. If no such file is found the JSON ConfigurationFactory will look for log4j2-test.json or log4j2-test.jsn in the classpath.
> 4. If no such file is found the XML ConfigurationFactory will look for log4j2-test.xml in the classpath.
> 5. If a test file cannot be located the YAML ConfigurationFactory will look for log4j2.yaml or log4j2.yml on the classpath.
> 6. If a YAML file cannot be located the JSON ConfigurationFactory will look for log4j2.json or

> log4j2.jsn on the classpath.
> 7. If a JSON file cannot be located the XML ConfigurationFactory will try to locate log4j2.xml on the classpath.
> 8. If no configuration file could be located the DefaultConfiguration will be used. This will cause logging output to go to the console.

一个简单的配置文件如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--status 代表log4j中什么类型的日志会被记录-->
<Configuration status="WARN">
    <!--Appenders 配置记录到哪里-->
    <Appenders>
        <!--命令行输入-->
        <Console name="Console" target="SYSTEM_OUT">
            <!--记录格式-->
            <PatternLayout pattern="%d{HH:mm:ss.SSS} [%
        </Console>
    </Appenders>
    <Loggers>
        <!--默认root的logger是error才会记录-->
        <Root level="error">
            <AppenderRef ref="Console"/>
        </Root>
    </Loggers>
</Configuration>
```

如果想对类中声明的Logger单独配置，只需要在Loggers中添加对应的Logger标签就行了。

```
<Loggers>
    <Logger name="com.foo.Bar" level="trace" additivity
        <AppenderRef ref="Console"/>
    </Logger>
```

```
        <Root level="error">
            <AppenderRef ref="Console"/>
        </Root>
    </Loggers>
```

其中 `additivity="false"` 表示Root将不会记录Logger命中的日志信息。

更多信息

xml配置文档
Appenders配置文档