配置log4j涉及分配级别，定义追加程序，并在配置文件中指定布局的对象。
log4j.properties文件是一个键 – 值对保存 log4j 配置属性文件。默认情况下，日志管理在CLASSPATH 查找一个名为 log4j.properties 的文件。

- 根日志记录器的级别定义为DEBUG并连接附加器命名为X到它
- 设置名为X的附加目的地是一个有效的appender
- 设置布局的附加器X

log4j.properties 语法:
以下是 log4j.properties 文件的一个appender X的语法：

```
# Define the root logger with appender X
log4j.rootLogger = DEBUG, X

# Set the appender named X to be a File appender
log4j.appender.X=org.apache.log4j.FileAppender

# Define the layout for X appender
log4j.appender.X.layout=org.apache.log4j.PatternLayout
log4j.appender.X.layout.conversionPattern=%m%n
```

log4j.properties 示例:
使用上面的语法，我们定义 log4j.properties 文件如下：

- 根日志记录器(logger)的级别定义为DEBUG并连接附加器命名为FILE
- 附加器(appender)File是定义为org.apache.log4j.FileAppender并写入到一个名为“log.out”位于日志log目录下
- 定义的布局模式是%m%n，这意味着每打印日志消息之后，将加上一个换行符

```
# Define the root logger with appender file
log4j.rootLogger = DEBUG, FILE

# Define the file appender
log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=${log}/log.out

# Define the layout for file appender
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.conversionPattern=%m%n
```

需要注意的是log4j支持UNIX风格的变量替换，如 ${variableName}.
调试级别：
使用DEBUG两个追加程序。所有可能的选项有：

- TRACE
- DEBUG
- INFO
- WARN

- ERROR
- FATAL
- ALL

Appenders:

Apache的log4j提供Appender对象主要负责打印日志消息到不同的目的地，如控制台，文件，sockets，NT事件日志等等。

每个Appender对象具有与之相关联的不同的属性，并且这些属性表明对象的行为

属性	描述
layout	Appender使用布局Layout 对象和与之相关的格式化的日志记录信息转换模式
target	目标可以是一个控制台，一个文件，或根据附加器的另一个项目
level	级别是必需的，以控制日志消息的过滤
threshold	Appender可以有与之独立的记录器级别相关联的级别阈值水平。Appender忽略具有级别低于阈级别的任何日志消息
filter	Filter 对象可以分析超出级别的匹配记录信息，并决定是否记录的请求应该由一个特定 Appender 或忽略处理

可以通过包括以下方法的配置文件中的下面设置一个 Appender 对象添加到记录器：

```
log4j.logger.[logger-name]=level, appender1,appender..n
```

可以编写以XML格式相同的结构如下：

```
<logger name="com.apress.logging.log4j" additivity="false">
   <appender-ref ref="appender1"/>
   <appender-ref ref="appender2"/>
</logger>
```

如果想要添加Appender对象到程序，那么可以使用下面的方法：

```
public void addAppender(Appender appender);
```

addAppender()方法添加一个appender到Logger对象。作为示例配置演示，可以添加很多Appender对象到记录器在逗号分隔的列表，每个打印日志信息分离目的地。

我们仅使用一个附加目的地FileAppender在我们上面的例子。所有可能的附加目的地选项有：

- AppenderSkeleton
- AsyncAppender
- ConsoleAppender
- DailyRollingFileAppender
- ExternallyRolledFileAppender
- FileAppender
- JDBCAppender
- JMSAppender
- LF5Appender
- NTEventLogAppender
- NullAppender

- RollingFileAppender
- SMTPAppender
- SocketAppender
- SocketHubAppender
- SyslogAppender
- TelnetAppender
- WriterAppender

Layout:

我们使用的PatternLayout 使用 appender。所有可能的选项有：

- DateLayout
- HTMLLayout
- PatternLayout
- SimpleLayout
- XMLLayout

使用HTMLLayout和XMLLayout，可以在HTML和XML格式和生成日志。

下面创建一个简单的配置文件：

- 下载最新的Log4j库：http://logging.apache.org/log4j/2.x/download.html
- 根记录器的级别定义为DEBUG并连接appender命名为FILE。
- appender FILE文件被定义为 org.apache.log4j.FileAppender 并写入到一个名为“log.out”位于 log 目录下。
- 定义的布局模式是 %m%n，这意味着打印日志消息之后自动加上一个换行符。

所以 log4j.properties 文件的内容如下：

```
# Define the root logger with appender file
log = D:/
log4j.rootLogger = DEBUG, FILE

# Define the file appender
log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=${log}/log.out

# Define the layout for file appender
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.conversionPattern=%m%n
```

在Java程序中使用log4j：

下面的Java类是一个非常简单的例子，Java应用程序初始化，然后使用Log4J日志库。

```
import org.apache.log4j.Logger;

import java.io.*;
import java.sql.SQLException;
import java.util.*;
```

```
public class log4jExample{
  /* Get actual class name to be printed on */
  static Logger log = Logger.getLogger(
                      log4jExample.class.getName());

  public static void main(String[] args)
                throws IOException,SQLException{

     log.debug("Hello this is an debug message");
     log.info("Hello this is an info message");
  }
}
```

编译和运行：

下面是步骤编译并运行上述程序。确保在进行编译和执行之前，适当地设置PATH和CLASSPATH。

所有的库应该在 CLASSPATH 和 log4j.properties 文件应该在PATH可用。所以，做到以下几点：

- 创建log4j.properties如上图所示。
- 创建log4jExample.java如上图所示，并对其进行编译。
- 执行log4jExample二进制运行程序。

在里面 /usr/home/log4j/log.out 文件会得到下面的结果：

```
Hello this is an debug message
Hello this is an info message
```