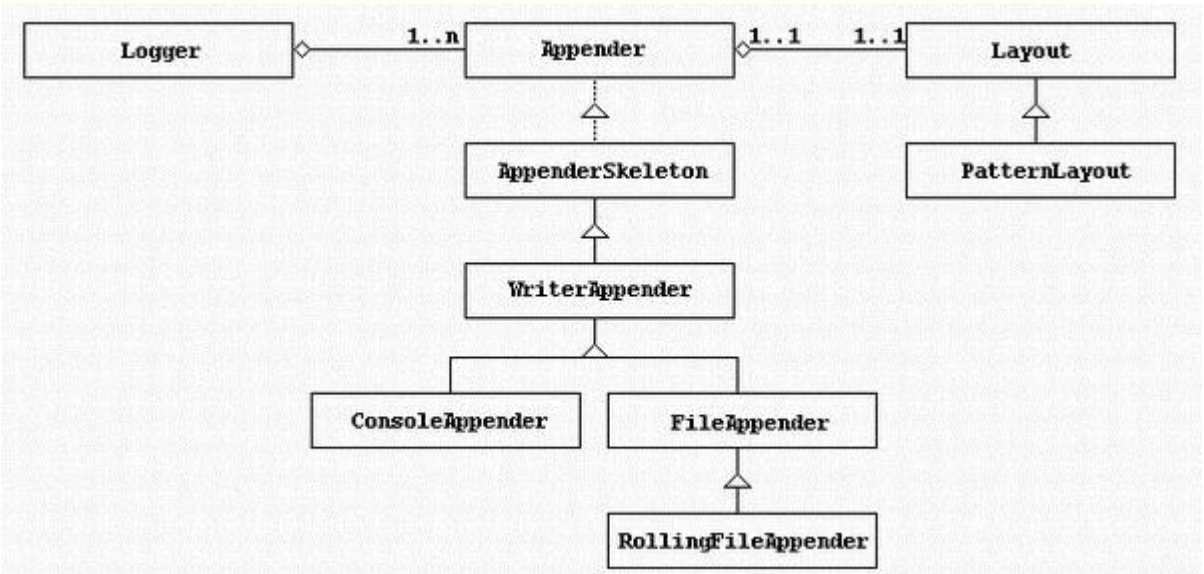
log4j 详解及简易搭建

log4j 是一个非常强大的 log 记录软件.
首先当然是得到 log4j 的 jar 档，推荐使用 1.2.X 版，下载地址：
http://logging.apache.org/log4j/1.2/download.html

下面先来看 Log4J 的类图



Logger - 日志写出器，供程序员输出日志信息
Appender - 日志目的地，把格式化好的日志信息输出到指定的地方去
ConsoleAppender - 目的地为控制台的 Appender
FileAppender - 目的地为文件的 Appender
RollingFileAppender - 目的地为大小受限的文件的 Appender
Layout - 日志格式化器，用来把程序员的 logging request 格式化成字符串
PatternLayout - 用指定的 pattern 格式化 logging request 的 Layout

Log4j 基本使用方法

Log4j 由三个重要的组件构成：日志信息的优先级，日志信息的输出目的地，日志信息的输出格式。日志信息的优先级从高到低有 ERROR、WARN、 INFO、DEBUG，分别用来指定这条日志信息的重要程度；日志信息的输出目的地指定了日志将打印到控制台还是文件中；而输出格式则控制了日志信息的显 示内容。

一、定义配置文件(文件名一般用 log4j.properties,置于 src 目录下)

其实您也可以完全不使用配置文件，而是在代码中配置 Log4j 环境。但是，使用配置文件将使您的应用程序更加灵活。Log4j 支持两种配置文件格式，一种是 XML 格式的文件，一种是 Java 特性文件（键=值）。下面我们介绍使用 Java 特性文件做为配置文件的方法：

1.配置根 Logger，其语法为：

log4j.rootLogger = [level] , appenderName, appenderName, …

其中，level 是日志记录的优先级，分为 OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL 或者您定义的级别。Log4j 建议只使用四个级别，优 先级从高到低分别是 ERROR、WARN、INFO、DEBUG。通过在这里定义的级别，您可以控制到应用程序中相应级别的日志信息的开关。比如在这里定 义了 INFO 级别，则应用程序中所有 DEBUG 级别的日志信息将不被打印出来。appenderName 就是指 B 日志信息输出到哪个地方。您可以同时指定多个输出目的地。

2.配置日志信息输出目的地 Appender，其语法为：

log4j.appender.appenderName = fully.qualified.name.of.appender.class
log4j.appender.appenderName.option1 = value1
…
log4j.appender.appenderName.option = valueN

其中，Log4j 提供的 appender 有以下几种：
org.apache.log4j.ConsoleAppender（控制台），
org.apache.log4j.FileAppender（文件），
org.apache.log4j.DailyRollingFileAppender（每天产生一个日志文件），
org.apache.log4j.RollingFileAppender（文件大小到达指定尺寸的时候产生一个新的文件），
org.apache.log4j.WriterAppender（将日志信息以流格式发送到任意指定的地方）

3.配置日志信息的格式（布局），其语法为：

log4j.appender.appenderName.layout = fully.qualified.name.of.layout.class
log4j.appender.appenderName.layout.option1 = value1
…
log4j.appender.appenderName.layout.option = valueN

其中，Log4j 提供的 layout 有以 e 几种：
org.apache.log4j.HTMLLayout（以 HTML 表格形式布局），

org.apache.log4j.PatternLayout（可以灵活地指定布局模式），
org.apache.log4j.SimpleLayout（包含日志信息的级别和信息字符串），
org.apache.log4j.TTCCLayout（包含日志产生的时间、线程、类别等等信息）

Log4J 采用类似 C 语言中的 printf 函数的打印格式格式化日志信息，打印参数如下： %m 输出代码中指定的消息

%p 输出优先级，即 DEBUG，INFO，WARN，ERROR，FATAL
%r 输出自应用启动到输出该 log 信息耗费的毫秒数
%c 输出所属的类目，通常就是所在类的全名
%t 输出产生该日志事件的线程名
%n 输出一个回车换行符，Windows 平台为“rn”，Unix 平台为“n”
%d 输出日志时间点的日期或时间，默认格式为 ISO8601，也可以在其后指定格式，比如：%d{yyy MMM dd HH:mm:ss,SSS}，输出类似：2002 年 10 月 18 日 22： 10：28，921
%l 输出日志事件的发生位置，包括类目名、发生的线程，以及在代码中的行数。举例：Testlog4.main(TestLog4.java:10)

二、在代码中使用 **Log4j** (后面有实例代码)

1.得到记录器

使用 Log4j，第一步就是获取日志记录器，这个记录器将负责控制日志信息。其语法为：

```
public static Logger getLogger( String name)
```

通过指定的名字获得记录器，如果必要的话，则为这个名字创建一个新的记录器。Name 一般取本类的名字，比如：

```
static Logger logger = Logger.getLogger ( ServerWithLog4j.class.getName () )
```

2.读取配置文件**(当 properties 文件名为: log4j.properties 且置于 src 文件下时,可省略此步骤,系统会自动加载该文件)**

当获得了日志记录器之后，第二步将配置 Log4j 环境，其语法为：
BasicConfigurator.configure ()： 自动快速地使用缺省 Log4j 环境。
PropertyConfigurator.configure (String configFilename) ：读取使用 Java 的特性文件编写的配置文件。

DOMConfigurator.configure (String filename) ：读取 XML 形式的配置文件。

3.插入记录信息（格式化日志信息）

当上两个必要步骤执行完毕，您就可以轻松地使用不同优先级别的日志记录语句插入到您想记录日志的任何地方，其语法如下：

```
logger.trace( Object message) ;
  logger.debug ( Object message ) ;
logger.info ( Object message ) ;
logger.warn ( Object message ) ;
logger.error ( Object message ) ;
logger.fatal( Object message);
```

实例代码

实例 1.FileAppender 的 PatternLayout 样式的配置

```
#设置级别和目的地,即输出到 file 所指向的目的地
log4j.rootLogger = debug , file

#输出到文件
log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.File=project.log
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L-%m%n
```

实例 2.ConsoleAppender 的 SimpleLayout 样式的配置

```
#设置级别和目的地,即 out 所指向的位置
log4j.rootLogger = debug , out

#指向控制台,采用 SimpleLayout
log4j.appender.out = org.apache.log4j.ConsoleAppender
log4j.appender.out.Target=System.out
log4j.appender.out.layout = org.apache.log4j.SimpleLayout
```

实例 3. 组合 appender 的配置(包括 file 跟 out,名字可自定义),加多一个 logger 的配置

```
#设置级别和目的地,即输出到 file 所指向的目的地 (这里设为 Info 级别,表明只有优先级高于 info 的才会输出到目的地.例如在这里 debug 信息不会输出)
log4j.rootLogger = info, file , out

#配置 convention 日志记录 logger 级别为 DEBUG,最后会使用 rootLogger 中设定的 Appender 进行日志输出
log4j.logger.org.apache.struts2.convention=DEBUG

#配置 hibernate 的 hbm2dll 日志记录级别为 DEBUG,输出到 out, 即覆盖了父 logger--rootLogger 的配置
log4j.logger.org.hibernate.tool.hbm2dll=DEBUG , out

#输出到文件
log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.File=project.log
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

#指向控制台,采用 SimpleLayout
log4j.appender.out = org.apache.log4j.ConsoleAppender
log4j.appender.out.Target=System.out
log4j.appender.out.layout = org.apache.log4j.SimpleLayout
```

实例 4. JDBCAppender 输出到数据库的配置

```
 log4j.rootLogger = error , database
#指向 JDBC 数据库 , 使用 PatternLayout
log4j.appender.database = org.apache.log4j.jdbc.JDBCAppender
#ERROR 或者 ERROR 以上级别输出
log4j.appender.database.Threshold=ERROR
#配置数据库连接信息
log4j.appender.database.URL=jdbc:mysql://localhost:3306/log4j
log4j.appender.database.driver=com.mysql.jdbc.Driver
log4j.appender.database.user=root
```

```
log4j.appender.database.password=admin
#配置 sql 语句
log4j.appender.database.sql=INSERT INTO tb_log (date , priority, message , c
lassname ) VALUES ('%d' , '%p' , '%m' , '%c' )
#配置 PatternLayout
log4j.appender.database.layout=org.apache.log4j.PatternLayout
log4j.appender.database.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss:SS
S}  %m
```

实例 5. 代码中使用 Log4j.

```
package com.mai.test;
import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

public class Log4jTest {

    public static void main(String[] args) {

        Logger log = Logger.getLogger(Log4jTest.class);
        //这里的路径相对于工程根目录,但假如log4j.properties是置于src目录下,也可以省略
此句,因为系统会自动加载该文件
        PropertyConfigurator.configure("src/log4j.properties");
        log.debug("yes,debug");
        log.info("yes,info");
        log.error("yes,error");
        log.warn("yes,warn");
    }

}
```

输出结果: (这里只配置了 out 控制台记录器,采用 SimpleLayout)

```
DEBUG - yes,debug
INFO - yes,info
ERROR - yes,error
WARN - yes,warn
```