Appender：

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）

org.apache.log4j.jdbc.JDBCAppender（把日志用JDBC记录到数据库中）

前三种常用

Layout

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）

%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“\r\n”，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，

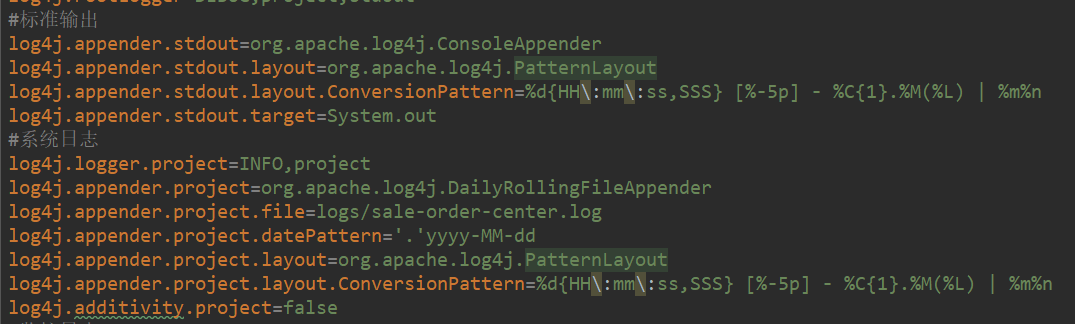
比如：%d{yyyy-MM-dd HH:mm:ss,SSS}，输出类似：2015-12-20 18:35:51,768

Level：

OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL。

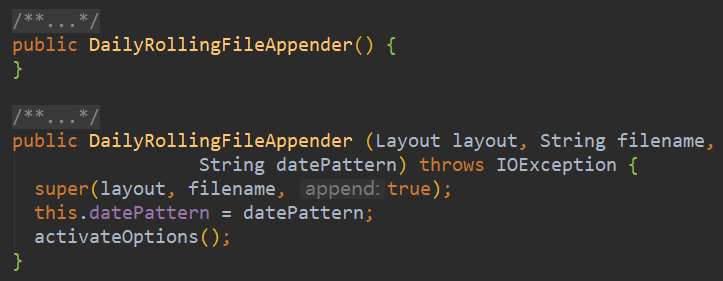
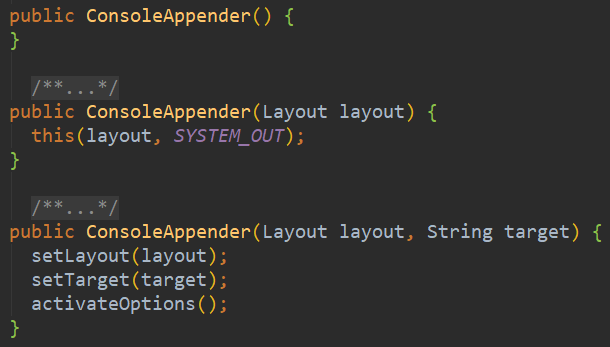
Log4j建议只用四个级别分别是ERROR、WARN、INFO、DEBUG。级别依次递增

Logger：



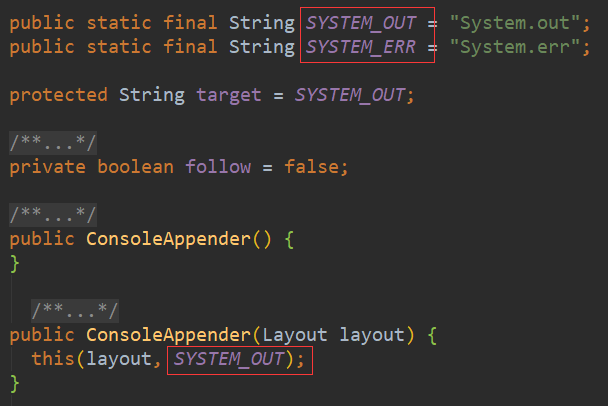
stdout，project都是日志输出源的名称，可以任意，但是必须准守log4j.appender.Xxx

log4j.appender.Xxx = org.apache.log4j.XxxAppender表示以什么样的格式输出，下一行依次为layout，file，dataPattern，target不同的Appender，下一行不同，org.apache.log4j. XxxAppender实际是log4j的一个类

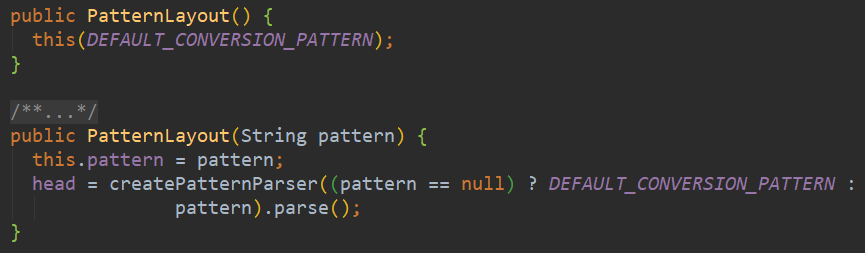


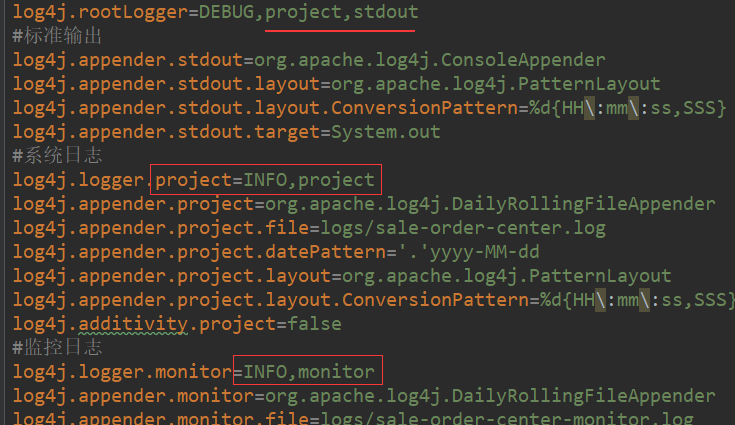
而这些appender都其构造函数，而layout，file，dataPattern，target为其对于的属性，但是ConversionPattern就不是appender的属性，而是layout的属性，而file，target，dataPattern的格式在其构造器中都有说明

Target是ConsoleAppeer的属性，在向控制台输出的时候，以哪种方式



Layout为输出样式





这里的log4j.logger.project=INFO,project表示一个logger，当前的日志输出的级别是INFO

log4j.rootLogger=DEBUG,project,stdout：root表示项目中有日志输入的默认采用debug，输出地方有project，stdout两个，而stdout采用rootLogger的级别Debug，而project采用info

log4j.rootLogger可以引用logger(project)，也可以应用appender(stdout)

log4j.logger.com.xgimi.sale.order.center=DEBUG,stdout,project：

可以为指定的路劲设置日志级别和输出源，这里的project是输出源，而不是logger(只有rootLogger才能应用logger，logger只能应用输出源)，但是此时center下就有个logger，rootLogger/centerLogger，在控制台就执行两次stdout输出源，但是级别又采用谁的呢？

级别取就近，输出看各自：center下有两个logger，级别采用最近的，就是centerLogger，输出两次，输出看各自

但是这样会导致在控制台输出重复(输出看各自)，可以在输出源上添加log4j.additivity.monitor=false停止传播，即一个输出源只会执行一次



也支持xml方式的配置文件，xml方式也执行additivity精确匹配方式，而xml方式还支持filter

当日志级别在levelMin和levelMax之间时，才会执行当前输出源

当properties和xml共存时，优先采用xml方式，一般都是采用xml