FRAMEWORK LOG4J

PLAN DU COURS

- Introduction

- MISE EN OEUVRE
- LOG4J.XML
- INSTRUCTION JAVA
- LES NIVEAUX DE LOG
- LES APPENDERS
- LES LAYOUTS

- Place à la Pratique

INTRODUCTION

- Log4j est une API de **journalisation** très répandue dans le monde Java.

- la bibliothèque log4j met trois sortes de composants à disposition du programmeur : les **loggers**, les **appenders** et les **layouts**.

- Les premiers permettent d'écrire les messages, les deuxièmes servent à sélectionner la destination des messages, et les derniers à mettre en forme les messages.

MISE EN OEUVRE

- Pour utiliser Log4j, il suffit d'ajouter le fichier **log4j-1.2.17.jar** dans le pom.xml de notre application.

- Ensuite, Il faut créer un fichier **log4j.xml** dans **src/main/resources**) : ce fichier contient la configuration de Log4j pour l'application.

- Dans le code source des classes, il faut :
  - Obtenir une instance du logger relative à la classe
  - Utiliser l'API pour émettre un message associé à un niveau de gravité

LOG4J.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration PUBLIC
"http://logging.apache.org/log4j/docs/api/org/apache/log4j/xml/log4j.dtd"
"http://logging.apache.org/log4j/docs/api/org/apache/log4j/xml/log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

<appender name="console" class="org.apache.log4j.ConsoleAppender">
    <layout class="org.apache.log4j.PatternLayout"> <param name="ConversionPattern" value="%d{yyyy-
   MM-dd HH:mm:ss} %l:%L - %m%n" /> </layout>
</appender>

<appender name="file" class="org.apache.log4j.RollingFileAppender">
    <param name="append" value="false" />     <param name="maxFileSize" value="10MB" />
    <param name="maxBackupIndex" value="10" />     <param name="file" value="C:/logs/formation.log"
   />
    <layout class="org.apache.log4j.PatternLayout"> <param name="ConversionPattern" value="%d{yyyy-
   MM-dd HH:mm:ss} %l:%L - %m%n" /></layout>
</appender>

<root> <level value="DEBUG" /> <appender-ref ref="console" /> <appender-ref ref="file" /> </root>

</log4j:configuration>
```

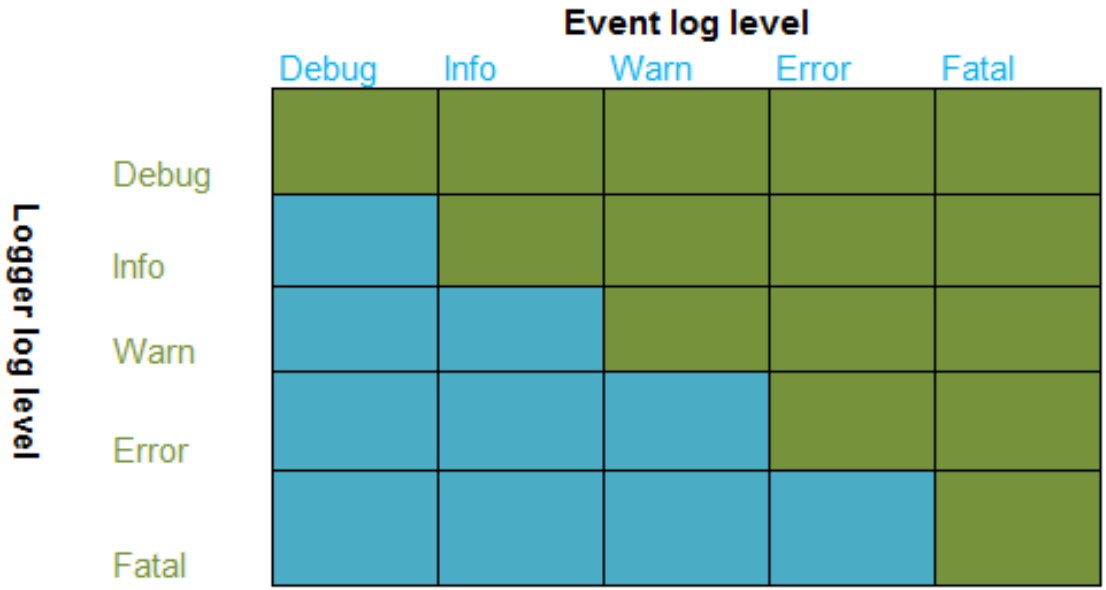
INSTRUCTION JAVA

```
package com.m2i.formation;
import org.apache.log4j.Logger;

public class HelloLog4J {

private static final Logger logger = Logger.getLogger(HelloLog4J.class);

public static void main(String a[]) {

if (logger.isTraceEnabled()) logger.trace("Test log TRACE");
if (logger.isDebugEnabled()) logger.debug("Test log DEBUG");
if (logger.isInfoEnabled()) logger.info("Test log INFO");
logger.warn("Test log WARN");
logger.error("Test log ERROR");
logger.fatal("Test log FATAL");

try {
if (logger.isInfoEnabled()) logger.info("Entrée dans Methode MAIN : ");
if (logger.isDebugEnabled()) logger.debug("Test log DEBUG");
if (logger.isInfoEnabled()) logger.info("Sortie de Methode MAIN : ");
} catch (Exception e) {
logger.error("Erreur dans la Methode  : " + e);
}

}
}
```

LES NIVEAUX DE LOG

- Log4j gère des priorités, ou Level, pour permettre au logger de déterminer si le message sera envoyé dans le fichier de log (ou la console). Il existe six priorités qui possèdent un ordre hiérarchique croissant :

  TRACE, DEBUG, INFO, WARN, ERROR, FATAL



LES APPENDERS

- L'interface org.apache.log4j.Appender désigne un flux qui représente le fichier de log et se charge de l'envoie de message formaté à ce flux.

- Un logger peut posséder plusieurs appenders. Si le logger décide de traiter la demande de message, le message est envoyés à chacun des appenders.

- Pour ajouter un appender à un logger, il suffit de le rajouter dans le fichier configuration log4j.properties :

```
<root>
<level value="WARN" />
<appender-ref ref="console" />
<appender-ref ref="file" />
</root>
```

LES LAYOUTS

- Ces composants représentés par la classe org.apache.log4j.Layout permettent de définir le format du fichier de log. Un layout est associé à un appender lors de son instanciation.

```
<layout class="org.apache.log4j.PatternLayout"> <param name="ConversionPattern"
  value="%d{yyyy-MM-dd HH:mm:ss} %l:%L - %m%n" /> </layout>
```

Motif	Role
%C	le nom de la classe qui a émis le message
%d	le timestamp de l'émission du message
%m	le message
%n	un retour chariot
%L	Le numéro de ligne dans le code émettant le message : l'utilisation de ce motif est coûteuse en ressources
%l	Des informations sur l'origine du message dans le code source (package, classe, méthode)

PLACE A LA PRATIQUE

Développement d'un Projet qui implémente LOG4J

Ajout de LOG4J dans un Projet déjà existant