Opis pliku log4j.properties

W bibliotece log4j wyróżniamy 3 typy obiektów:

- Logger'y
- Appender'y
- Layout'y

Loggery posiadają metody, które tworzą logi i ustawiają im odpowiedni priorytet.
Miejsca, do których mogą trafić logi są definiowane za pomocą **Appender'ów.**
O tym jaką postać mają mieć komunikaty decydują obiekty typu **Layout.**

Jeśli chodzi o ustawienie priorytetów, to wyróżniamy następujące poziomy:

- **FATAL**: Poważne błędy powodujące przedwczesne zakończenie działania aplikacji;
- **ERROR**: Błędy wykonania;
- **WARN**: Przy użyciu przestarzałych komponentów, w sytuacjach niespodziewanych nie wpływających na wadliwe działanie;
- **INFO**: W celu śledzenia wykonania;
- **DEBUG**: Szczegółowe info. dotyczące przepływu w działaniu aplikacji, w celach diagnostycznych;
- **TRACE**: Bardziej szczegółowe info;
- **ALL / OFF**: Wszystkie poziomy / wyłączenie logów.

Przy wyborze danego poziomu musimy wziąć pod uwagę to, że wszystkie logi wypisywane na poziomach niższych od tego który wybraliśmy nie będą wyświetlane. Przykładowo, jeśli poziom ustawimy na DEBUG (domyślny) to wówczas wszystkie logi wypisywane na poziomie TRACE nie ujrzą światła dziennego

W log4j wyróżniamy następujące Logger'y:

- NOPLogger,
- RootCategory,
- RootLogger - najważniejszy, z niego korzystamy

Jest dużo rodzajów Appenderów (czyli miejsc, gdzie mogą być zapisywane logi) - my używamy dwóch:

- ConsoleAppender - wypisuje logi na konsolę
- FileAppender - wpisuje logi do pliku

PatternLayout - obiekt Layout - umożliwia określenie szablonu wpisu

kod pliku log4j.properties

// tworzenie logu, określenie priorytetu (DEBUG) i “deklaracja” dwóch miejsc zapisu logów (Appender1 i Appender2)
log4j.rootLogger=DEBUG, Appender1, Appender2

// dokładne określenie pierwszego miejsca zapisu logów (ConsoleAppender → wyświetli logi na konsoli)
log4j.appender.Appender1=org.apache.log4j.ConsoleAppender

// utworzenie obiektu PatternLayout - który będzie określać, jak ma wyglądać komunikat dotyczący logów
log4j.appender.Appender1.layout=org.apache.log4j.PatternLayout

// dokładne określenie jak ma wyglądać wyświetlany komunikat - ConversionPattern zostaje ustawiony na wartość “%-7p %d [%t] %c %x - %m%n” , gdzie:

The percent sign (%) is required which indicates the start of a specifier.
The specifiers used in the above example:

- **t**: name of the current executing thread.
- **p**: priority
- **c**: category
- **m**: log message.
- **n**: line separator character.

Dzięki ustawieniu takiej wartości, postać wyświetlanych logów wygląda następująco:
DEBUG 2020-05-25 23:04:07,709 [main] MainClass - this is a debug log message

log4j.appender.Appender1.layout.ConversionPattern=%-7p %d [%t] %c %x - %m%n

// dokładnie to samo dla Appendera 2 → określenie miejsca zapisu (FileAppender - zapis do pliku)
log4j.appender.Appender2=org.apache.log4j.FileAppender

// wskazanie nazwy i rozszerzenia pliku, do którego mają być zapisane logi
log4j.appender.Appender2.File=applog.txt

// utworzenie obiektu PatternLayout - który będzie określać, jak ma wyglądać komunikat dotyczący logów
log4j.appender.Appender2.layout=org.apache.log4j.PatternLayout

// dokładne określenie wyświetlanego komunikatu
log4j.appender.Appender2.layout.ConversionPattern=%-7p %d [%t] %c %x - %m%n