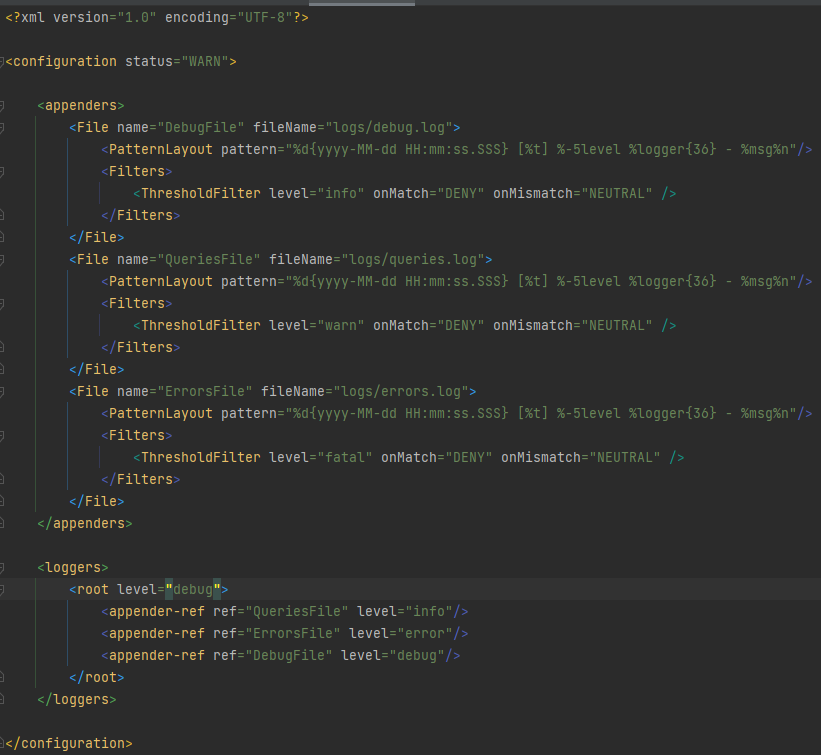
**Заметки по логированию с log4j2**



Уровни логирования:

* **OFF:** никакие логи не записываются, все будут проигнорированы;
* **FATAL:** ошибка, после которой приложение уже не сможет работать и будет остановлено, например, JVM out of memory error;
* **ERROR:** уровень ошибок, когда есть проблемы, которые нужно решить. Ошибка не останавливает работу приложения в целом. Остальные запросы могут работать корректно;
* **WARN:** обозначаются логи, которые содержат предостережение. Произошло неожиданное действие, несмотря на это система устояла и выполнила запрос;
* **INFO:** лог, который записывает важные действия в приложении. Это не ошибки, это не предостережение, это ожидаемые действия системы;
* **DEBUG:** логи, необходимые для отладки приложения. Для уверенности в том, что система делает именно то, что от нее ожидают, или описания действия системы: “method1 начал работу”;
* **TRACE:** менее приоритетные логи для отладки, с наименьшим уровнем логирования;
* **ALL:** уровень, при котором будут записаны все логи из системы.

В пороговом значении фильтра (**ThresholdFilter**) мы указываем на один уровень логирования выше необходимого уровня, т. к. отсекается, все то, что соответствует указанному уровню и выше этого уровня. Например, если будем создавать логи уровня «debug» указываем пороговым значением фильтра уровень «info», для логов уровня «info» пороговое значение фильтра «warn», а для логов уровня «error» ставим порог «fatal» и т. п.

Значение «onMatch» устанавливаем «DENY». Это значение указывает на то, что при выполнении условия мы рассматриваем только уровни выше указанного в фильтре порогового значения, и запишем лог с нужным уровнем (этот уровень, как я понимаю, указывается уже в самом logger, в нашем случае «root»), а остальные отклоняем, пропускаем в следующие фильтры.

А если бы мы установили для «onMatch» значение «ACCEPT» и пороговое значение фильтра уровень записываемого лога, например, собираемся записывать «info», и установили ThresholdFilter level=«info», то лог бы записался, но остальные уровни логов уже не попали бы в другие фильтры.

Важно не забыть указать уровень корневого логгера ниже или равным самому нижнему из уровней лога, которые мы будем записывать. В противном случае, все искомые логи, уровень которых окажется ниже уровня логгера, не будут отслеживаться, и не будут записаны.

В нашем случае мы планируем записывать логи уровней «debug», «info» и «error». Каждый в уровень в отдельный файл. В данном случае у нас один логгер, это корневой логгер (root). Т. к. мы уже определились, что собираемся записывать логи уровней «debug», «info» и «error», а самым нижним уровнем в иерархии уровней из них будет «debug», то и указываем его уровнем нашего логгера:

<root level="debug">

Даем осмысленные названия аппендерам, и указываем адреса файлов, в которые будем записывать логи:

<File name="DebugFile" fileName="logs/debug.log">

<File name="QueriesFile" fileName="logs/queries.log">

<File name="ErrorsFile" fileName="logs/errors.log">

Указываем шаблоны для записи логов:

<PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>

Устанавливаем пороговые значения для каждого фильтра и их значения «onMatch» и «onMismatch»:

<ThresholdFilter level="info" onMatch="DENY" onMismatch="NEUTRAL" />

<ThresholdFilter level="warn" onMatch="DENY" onMismatch="NEUTRAL" />

<ThresholdFilter level="fatal" onMatch="DENY" onMismatch="NEUTRAL" />

Указываем аппендеры с указанием их уровней для нашего логгера:

<appender-ref ref="QueriesFile" level="info"/>  
<appender-ref ref="ErrorsFile" level="error"/>  
<appender-ref ref="DebugFile" level="debug"/>

Попробуем прочитать файл с настройками.

Создаем три аппендера: «DebugFile» с файлом назначения «logs/debug.log» и пороговым значением «info», «QueriesFile» с файлом назначения «logs/queries.log» и пороговым значением «warn» и «ErrorsFile» с файлом назначения «logs/errors.log» и пороговым значением «fatal». Уровень корневого логгера установлен «debug», поэтому будут записываться только логи уровня «debug» и выше. Сначала отработает фильтр аппендера «DebugFile». Т. к. пороговое значение этого фильтра «info», то когда выполняется условие этого фильтра, который имеет уровень логов «debug», то лог «debug» будет записан в файл «logs/debug.log», а логи с уровнем «info» и выше, благодаря значению «onMatch» равному «DENY» будут отклонены (отправлены дальше, в следующие фильтры). Следующим отработает фильтр аппендера «QueriesFile», он отправит в следующий фильтр все логи с уровнем «warn» и выше, а себе оставит и запишет логи своего уровня - «info» в файл «logs/queries.log». И последний фильтр аппендера «ErrorsFile» пропускает дальше логи уровня «fatal» и выше, и записывает логи уровня «error» в файл «logs/errors.log». Хотя в последнем фильтре можно установить пороговое значение «error», а для «onMatch» можно указать значение «ACCEPT», т. к. больше у нас нет фильтров, куда дальше передавались бы логи. Так тоже отработает.

**ALL => TRACE => DEBUG => INFO => WARN => ERROR => FATAL => OFF**

Ситуацию с логирование можно представить, как если бы существовало несколько команд, например по баскетболу, разделенных по росту участников. Допустим, команд будет шесть. Самые высокие будут соответствовать уровню логирования **«TRACE»** (уровни **«ALL»** и **«OFF»** не учитываются), а самая младшая детская группа будет представлять собой уровень **«FATAL»**. Перед началом очередного сезона игр владелец команд решил изменить дизайн формы. Сшили новую форму для всех команд. Новую форму привезли для раздачи командам. Форма не была отсортирована по размерам. Команда **«TRACE»** размещается в другом корпусе и на раздачу формы не попала. Команда **«DEBUG»** просматривала только большие размеры форм **TRACE** и **DEBUG**, и из них отбирала форму своего размера – **DEBUG**, а форму размера **TRACE** просто оставляла в стороне. Форму размера **INFO** (пороговое значение фильтра)и меньше команда **«DEBUG»** передавала на сортировку командам младше. В нашем случае следующей командой, которая получала оставшиеся комплекты формы для отбора своего размера, стала команда **«INFO»**. Команда **«INFO»** забрала форму своего размера – **INFO**, а форму размера **WARN** и меньше передала следующей команде. Команда **WARN** также находится в другом корпусе спортивного комплекса. А вот команда **«ERROR»** из нашего корпуса. Она и принимает оставшиеся экземпляры формы для поиска своего размера. Отобрав все комплекты своего размера – **ERROR**, она оставляет неподошедшую по размеру форму на месте, если у нас для **onMatch** установлено значение **«ACCEPT»** или передает её куда-то дальше, если значение у **onMatch** будет **«DENY»**.