**JAVA логгирование**

Изначально в Java не было своего логгера, что привело к написанию нескольких независимых логгеров. Самым распространенным из них стал log4j. Спустя несколько лет, в Java все же был добавлен свой логгер, но его функциональность была гораздо беднее и большого распространения он не получил. Факт, как говорится, на лицо – в Java есть официальный логгер, но все сообщество Java-программистов предпочитает пользоваться другим. На основе log4j потом было написано еще несколько логгеров.

А затем для них всех был написан специальный универсальный логгер slf4j, который сейчас повсеместно используют. Он очень похож на log4j, поэтому я расскажу тебе логирование на его примере.

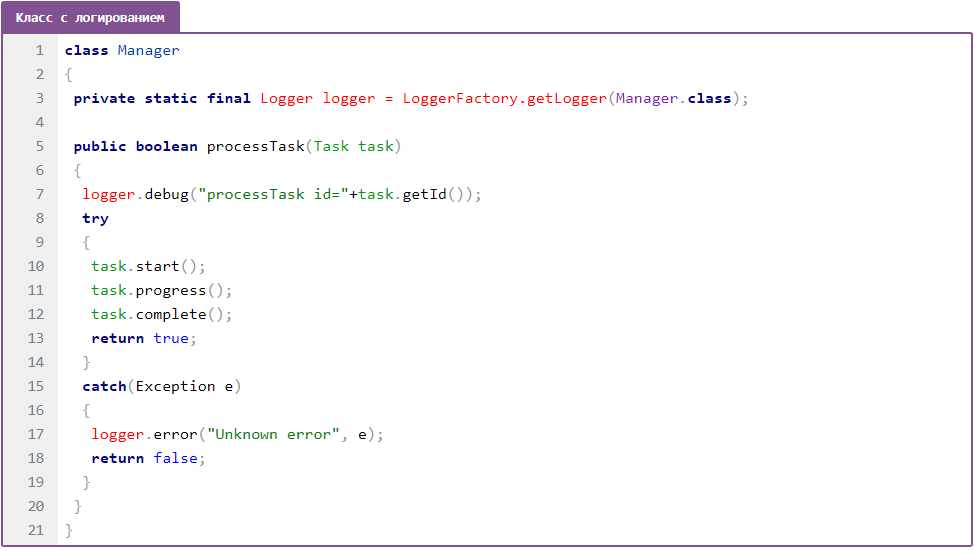
Весь процесс логирования состоит из трех частей.

**Первая часть** – это сбор информации.

**Вторая часть** – это фильтрование собранной информации.

**Третья часть** – это запись отобранной информации.

Начнем со сбора. Вот типичный пример класса, который ведет лог:



Обрати внимание на слова, выделенные красным.

**Строка 3** – создание объекта logger. Такой статический объект создают практически в каждом классе! Ну, разве что кроме классов, которые ничего не делают, а только хранят данные.

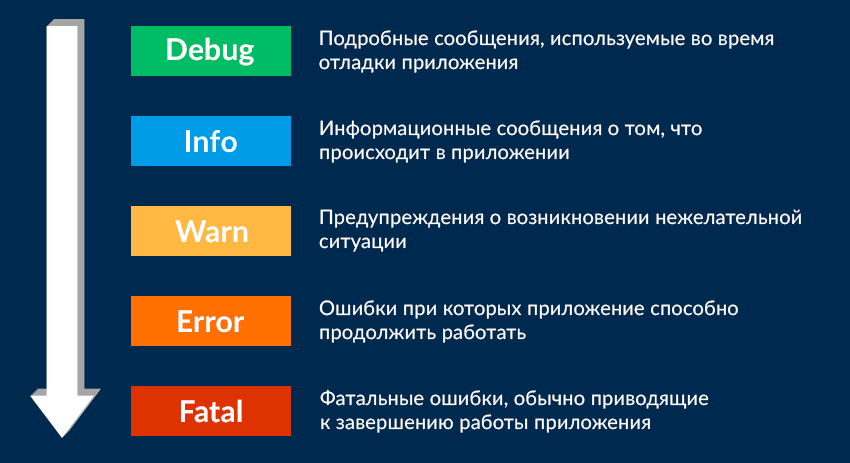
**LoggerFactory** – это специальный класс для создания логгеров, а getLogger – это его статический метод. В него обычно передают текущий класс, хотя возможны различные варианты.

**Строка 7** – в логгер пишется информация о вызове метода. Обрати внимание – это первая строчка метода. Только метод вызвался – сразу пишем информацию в лог.

Мы вызываем метод debug, это значит, что важность информации «уровня DEBUG». Этот факт используется на уровне фильтрации. Об этом я расскажу через пару минут.

**Строка 17** – мы перехватили исключение и… сразу же записали его в лог! Именно так и нужно делать.

На этот раз мы вызываем метод error, что сразу придает информации статус «ERROR»

[](https://javarush.ru/api/1.0/rest/images/739040/a33f0707-5dbf-4a43-9745-d17c04e6d7ef)

— Пока вроде все ясно. Ну, насколько это может быть ясно в середине разговора.

— Отлично, тогда перейдем к записи фильтрации.

Обычно, у каждого лог-сообщения есть своя степень важности, и, используя ее можно часть этих сообщений отбрасывать. Вот эти степени важности:

|  |  |
| --- | --- |
| **Степень важности** | **Описание** |
| ALL | Все сообщения |
| TRACE | Мелкое сообщение при отладке |
| **DEBUG** | **Сообщения важные при отладке** |
| **INFO** | **Просто сообщение** |
| **WARN** | **Предупреждение** |
| **ERROR** | **Ошибка** |
| **FATAL** | **Фатальная ошибка** |
| OFF | Нет сообщения |

Эти уровни используются еще и при отсеве сообщений.

Скажем, если выставить уровень логирования в WARN, то все сообщения, менее важные, чем WARN будут отброшены: TRACE, DEBUG, INFO.

Если выставить уровень фильтрации в FATAL, то будут отброшены даже ERROR’ы.

Есть еще два уровня важности, которые используются при фильтрации – это OFF – отбросить все сообщения и ALL – показать все сообщения (не отбрасывать ничего).

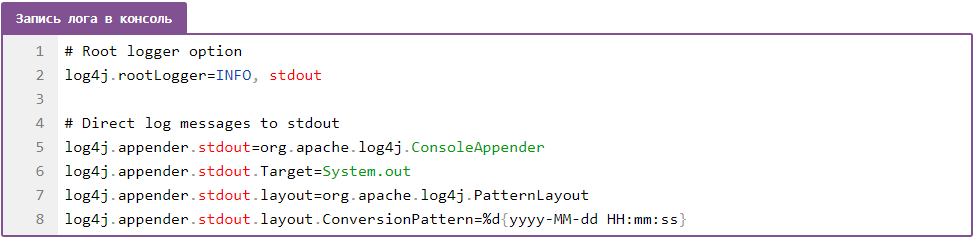
— А как настраивать фильтрацию и где?

— Сейчас расскажу.

Обычно настройки логгера log4j задаются в файле log4j.properties.

В этом файле можно задать несколько appender’ов – объектов, в которые будут писаться данные. Есть источники данных, а есть – аппендеры – противоположные по смыслу объекты. Объекты, куда как бы «стекают» данные, если их можно представить в виде воды.

Вот тебе несколько примеров:



Строки 1 и 4 – это комментарии

Строка 2 – мы указываем уровень сообщений, которые оставляем. Все менее важные уровни будут отброшены (DEBUG,TRACE)

Там же, через запятую, мы указываем имя объекта (сами придумываем), куда будет писаться лог. В строках 5-8 идут его настройки.

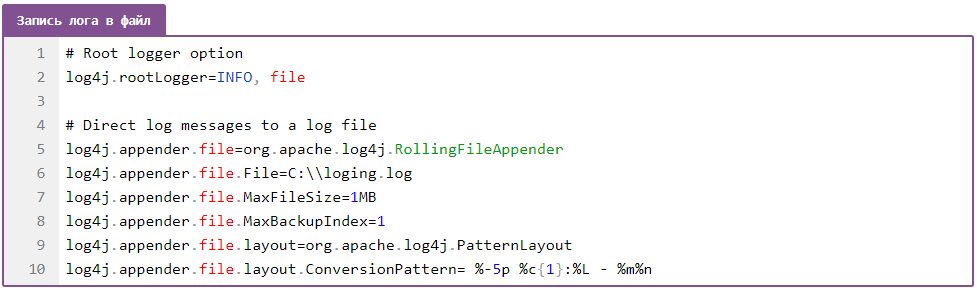
Строка 5 – указываем тип апендера – консоль (ConsoleAppender).

Строка 6 – указываем, куда именно будем писать – System.out.

Строка 7 – задаем класс, который будет управлять шаблонами записей – PatternLayout.

Строка 8 – задаем шаблон для записи, который будет использоваться. В примере выше это дата и время.

А вот как выгладит запись в файл:



Строка 2 задает уровень фильтрации сообщений и имя объекта-апендера (стока).

Строка 5 – указываем тип апендера – файл (RollingFileAppender).

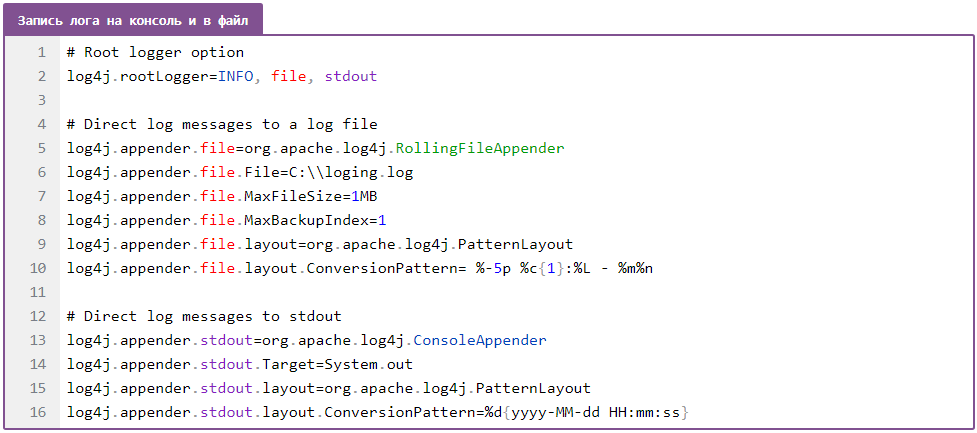
Строка 6 – указываем имя файла – куда писать лог.

Строка 7 – указываем максимальный размер лога. При превышении размера, начнет писаться новый файл.

Строка 8 – указываем количество старых файлов логов, которые надо хранить.

Строки 9-10 – задание шаблона сообщений.

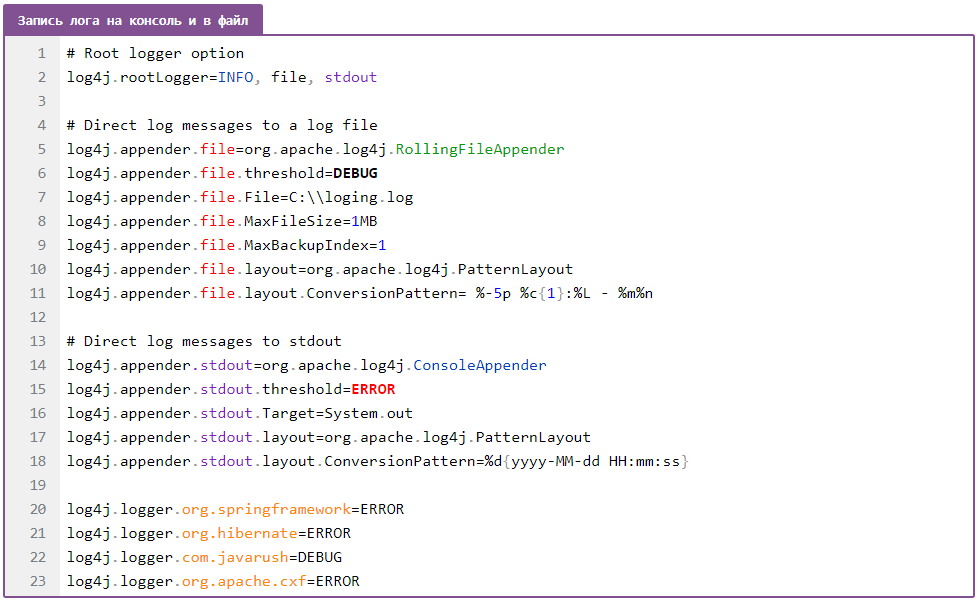
Пример, как писать лог в файл и на консоль:



Можете объявить сколько угодно апендеров и настроить каждый из них по-своему.

Более того, каждому апендеру можно очень гибко настроить фильтр его сообщений. Мы можем не только задать каждому апендеру свой уровень фильтрации сообщений, но и отфильтровать их по пакетам! Вот для чего надо указывать класс при создании логгера (я про **LoggerFactory.getLogger**).

Пример:



Строки 6 и 15 – мы задаем свой уровень фильтрации для каждого апендера.

Строки 20-23 мы указываем имя пакета и тип фильтрации его сообщений. «log4j.logger» — это префикс, имя пакета выделено оранжевым.

Кстати, ни log4j, ни slf4j не входят в JDK, скачивать их надо отдельно. Это можно сделать вот [тут](http://www.slf4j.org/download.html). Но есть и второй способ:

**Шаг 1.** Добавляете в класс импорты:

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**Шаг 2.** Становитесь курсором на эти строчки и нажимаешь Alt+Enter в Intellij IDEA

**Шаг 3.** Выбираете пункт File jar on web.

**Шаг 4**. Выбираете – slf4j-log4j13.jar

**Шаг 5.** Указываете, куда скачать библиотеку (jar)

**Шаг 6.** Пользуетесь нужными вам классами.