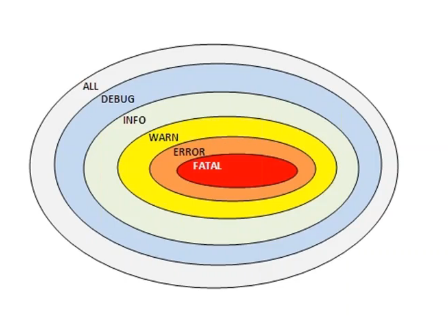
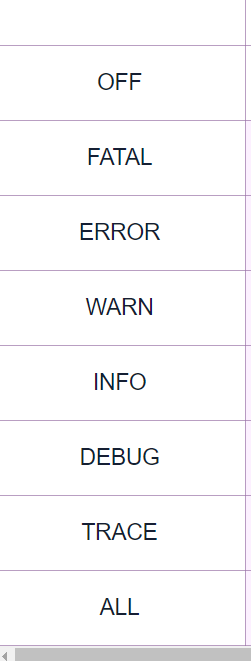
**<https://videoportal.epam.com/video/9oXYrLRw>**

Логирование

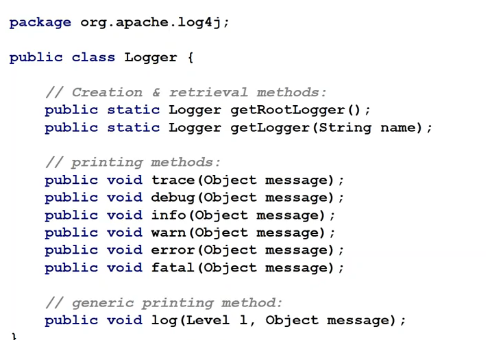
Для разбора упавших тестов.  
Библиотека apache Log4j

Зависимость в ром добавить. Xml файл в ресурсы  
уровени логирования



Верхний уровень показывает внутренние логи.

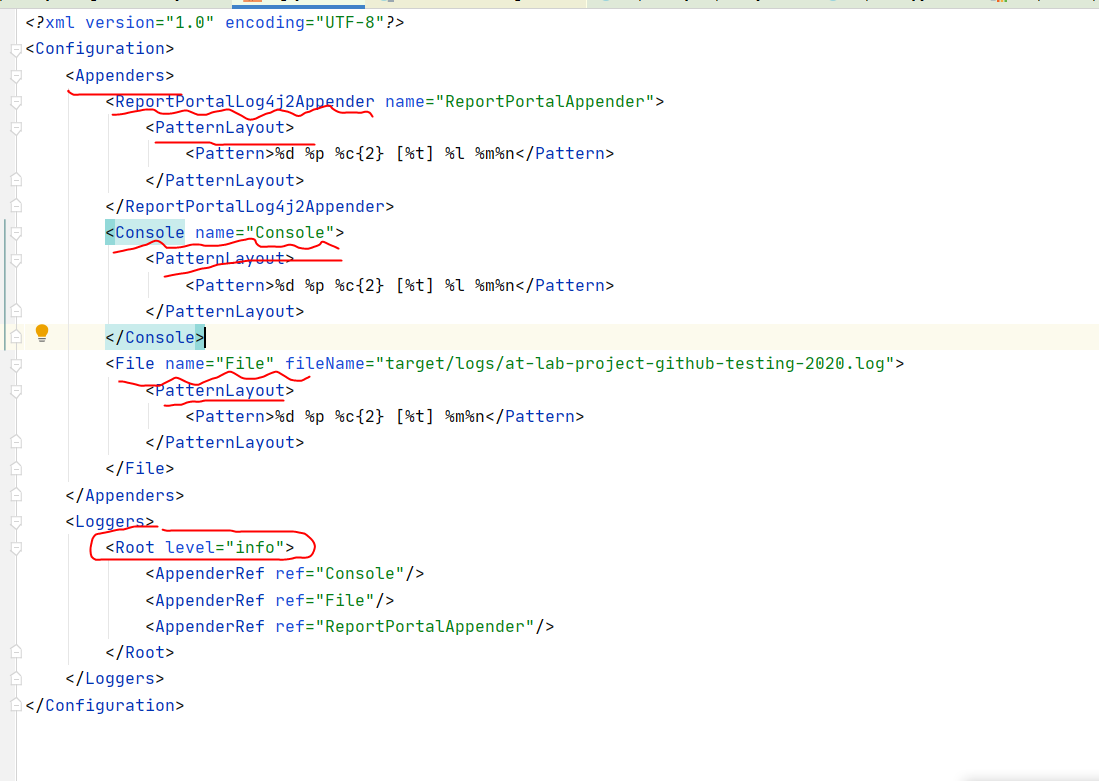
* ****OFF:**** никакие логи не записываются, все будут проигнорированы;
* ****FATAL:**** ошибка, после которой приложение уже не сможет работать и будет остановлено, например, JVM out of memory error;
* ****ERROR:**** уровень ошибок, когда есть проблемы, которые нужно решить. Ошибка не останавливает 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.



Можно создавать разные логеры, направлять запись/слушание на разные файлы; уст-ть уровень читаемости. Если loger.set(INFO),то DEBUG уже не прочтется

Логгер сообщает, апендер слушает сохраняет в какое-то место, лэйаут форматирует смс.

Appender - слушает логеры и сохраняет инфо (консоль, файл и т.д)  
Layout - апендер пользуется им для форматирования смс; формат задается. N, временная ветка



В PatternLayout мы указываем формат сообщения:

%d — дата и время

%p — уровень сообщения (DEBUG, WARN, INFO…)

%c{1.} — категория, причём имена пакетов будут сокращаться до первой буквы

%t — название потока

%m — сообщение для лога

%n — перевод строки



# Уровень логирования

log4j.rootLogger=INFO, file

# Апендер для работы с файлами

log4j.appender.file=org.apache.log4j.RollingFileAppender

# Путь где будет создаваться лог файл

log4j.appender.file.File=C:\\TMP\\log\_file.log

# Указываем максимальный размер файла с логами

log4j.appender.file.MaxFileSize=1MB

# Конфигурируем шаблон вывода логов в файл

log4j.appender.file.layout=org.apache.log4j.PatternLayout

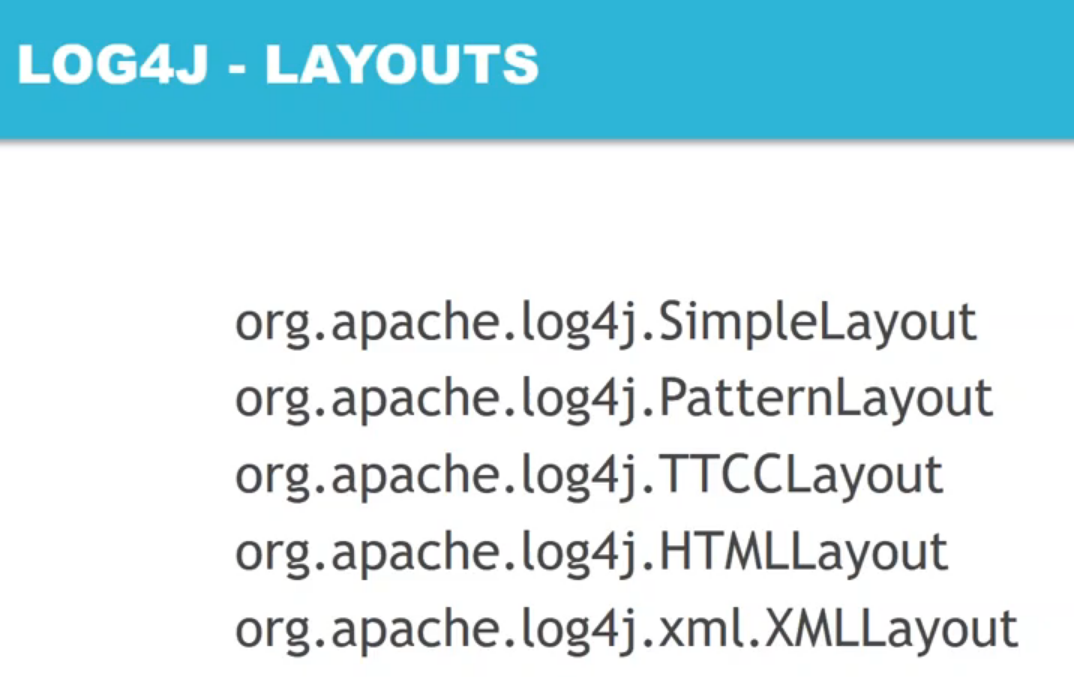
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

<https://devcolibri.com/%D1%83%D1%87%D0%B8%D0%BC%D1%81%D1%8F-%D0%B2%D0%B2%D0%B5%D1%81%D1%82%D0%B8-%D0%BB%D0%BE%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-log4j/>

## Запись и отправка логов: Appender

Этот процесс будем рассматривать на примере log4j: он предоставляет широкие возможности для записи/отправки логов:

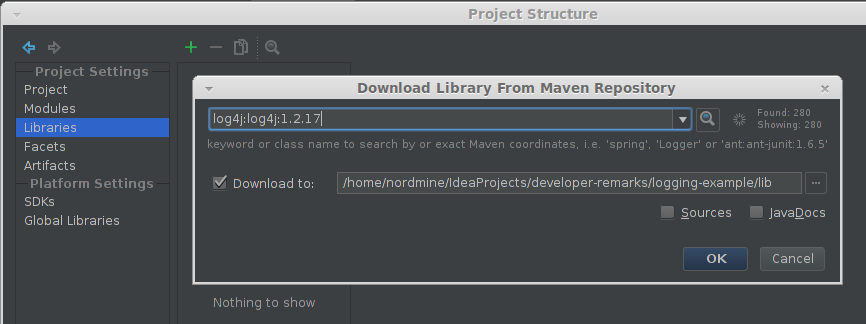
* для записи в файл — решение ****DailyRollingFileAppender****;
* для получения данных в консоль приложения — ****ConsoleAppender****;
* для записи логов в базу данных — ****JDBCAppender****;
* для контроля передачи через TCP/IP — ****TelnetAppender****;
* для того, чтобы запись логов не била по быстродействию — ****AsyncAppender****.



Log4j2 подключение (читать)  
<https://javastudy.ru/log4j/log4j-hello-world-example/>

1. TRACE - сообщения самого низкого уровня, наиболее подробные
2. DEBUG - отладочные сообщения, менее подробные, чем TRACE
3. INFO - стандартные информационные сообщения
4. WARN - некритичные ошибки, не препятствующие работе приложения
5. ERROR - ошибки, которые могут привести к неверному результату
6. FATAL - ошибки, препятствующие дальнейшей работе приложения

Как подключить стороннюю библиотеку к проекту. Открывайте **File** - **Project Structure**, затем в левой панели выберите **Libraries**, затем жмите на зелёный плюс. И тут у вас есть выбор: либо добавить jar-файл к проекту, указав его расположение в файловой системе, либо загрузив его из maven-репозитория. Чтобы загрузить библиотеку из репозитория, просто укажите её название и версию как указано на картинке ниже:

[](http://4.bp.blogspot.com/-Rt3I0vKbYeA/U4yjIV4TJII/AAAAAAAAA6g/m595rbptCkw/s1600/Screenshot-4.png)

Обязательно отметьте пункт **Download to**, не меняя его значения по умолчанию - тогда библиотека будет загружена прямо в ваш проект.

## **Какие существуют типы логов?**

* системы (System);
* безопасности (Security);
* приложения (Application, Buisness).

Пользователь входит в приложение, проверяется пароль. Это действие относится к безопасности (Security). Дальше он запускает какой-нибудь модуль. Это событие уровня приложения (Application). Модуль при старте обращается к другому модулю за какими-то дополнительными данными, производит какие-либо еще вызовы – это уже системные действия (System).

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/log.md" \l "%D0%96%D1%83%D1%80%D0%BD%D0%B0%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

## **Из каких частей состоит система журналирования log4j?**

Система журналирования состоит из трёх основных частей:

* управляющей журналированием - ****logger****;
* добавляющей в журнал - ****appender****;
* определяющей формат добавления - ****layout****.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/log.md" \l "%D0%96%D1%83%D1%80%D0%BD%D0%B0%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

## **Что такое Logger в log4j?**

****Logger**** представляет собой объект класса org.apache.log4j.Logger, который используется как управляющий интерфейс для журналирования сообщений с возможностью задавать уровень детализации. Именно logger проверяет нужно ли обрабатывать сообщение и если журналирование необходимо, то сообщение передаётся в appender, если нет - система завершает обработку данного сообщения.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/log.md" \l "%D0%96%D1%83%D1%80%D0%BD%D0%B0%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

## **Что такое Appender в log4j?**

****Appender**** - это именованный объект журнала событий, реализующий интерфейс org.apache.log4j.Appender и добавляющий события в журнал. Appender вызывает разные вспомогательные инструменты - компоновщик, фильтр, обработчик ошибок (если они определены и необходимы). В ходе этой работы окончательно устанавливается необходимость записи сообщения, сообщению придаются окончательные содержание и форма.

В log4j журнал может представлять:

* консоль;
* файл;
* сокет;
* объект класса реализующего java.io.Writer или java.io.OutputStream;
* JDBC хранилище;
* тему (topic) JMS;
* NT Event Log;
* SMTP;
* Syslog;
* Telnet.

Наиболее часто используемые log4j appender-ы:

* org.apache.log4j.ConsoleAppender - вывод в консоль;
* org.apache.log4j.FileAppender - добавление в файл;
* org.apache.log4j.DailyRollingFileAppender - добавление в файл с обновлением файла через заданный промежуток времени;
* org.apache.log4j.RollingFileAppender - добавление в файл с обновлением файла по достижению определенного размера;
* org.apache.log4j.varia.ExternallyRolledFileAppender - расширение RollingFileAppender обновляющее файл по команде принятой с заданного порта;
* org.apache.log4j.net.SMTPAppender - сообщение по SMTP;
* org.apache.log4j.AsyncAppender - позволяет, используя отдельный поток, организовать асинхронную работу, когда сообщения фиксируются лишь при достижении определенного уровня заполненности промежуточного буфера.
* org.apache.log4j.nt.NTEventLogAppender - добавление в NT Event Log;
* org.apache.log4j.net.SyslogAppender - добавление в Syslog;
* org.apache.log4j.jdbc.JDBCAppender - запись в хранилище JDBC;
* org.apache.log4j.lf5.LF5Appender - сообщение передаётся в специальный GUI интерфейс LogFactor5
* org.apache.log4j.net.SocketAppender - трансляция сообщения по указанному адресу и порту;
* org.apache.log4j.net.SocketHubAppender - рассылка сообщения сразу нескольким удалённым серверам соединённым по заданному порту;
* org.apache.log4j.net.TelnetAppender - отсылка сообщения по протоколу Telenet;
* org.apache.log4j.net.JMSAppender - добавление сообщения в JMS.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/log.md" \l "%D0%96%D1%83%D1%80%D0%BD%D0%B0%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

## **Что такое Layout в log4j?**

****Layout**** - наследник класса org.apache.log4j.Layout предоставляющий возможность форматирования сообщения перед добавлением в журнал.

В log4j существуют следующие типы layout-ов:

* org.apache.log4j.SimpleLayout - на выходе получается строка содержащая лишь уровень вывода и сообщение;
* org.apache.log4j.HTMLLayout - форматирует сообщение в виде элемента HTML-таблицы;
* org.apache.log4j.xml.XMLLayout - компанует сообщение в виде XML формате;
* org.apache.log4j.TTCCLayout - на выходе сообщение дополняется информацией о времени, потоке, имени логгера и вложенном диагностическом контексте;
* org.apache.log4j.PatternLayout / org.apache.log4j.EnhancedPatternLayout - настройка форматирования сообщения при помощи шаблона заданного пользователем.

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/log.md" \l "%D0%96%D1%83%D1%80%D0%BD%D0%B0%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

## **Перечислите уровни журналирования в log4j? Назовите порядок их приоритетности.**

* ****OFF**** - отсутствие журналирования;
* ****FATAL**** - фатальная ошибка;
* ****ERROR**** - ошибка;
* ****WARN**** - предупреждение;
* ****INFO**** - информация;
* ****DEBUG**** - детальная информация для отладки;
* ****TRACE**** – трассировка всех сообщений.

Между уровнями логирования установлен следующий порядок приоритетов:

OFF < TRACE < DEBUG < INFO < WARN < ERROR < FATAL < ALL

[к оглавлению](https://github.com/enhorse/java-interview/blob/master/log.md" \l "%D0%96%D1%83%D1%80%D0%BD%D0%B0%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

## **Какие существуют способы конфигурирования log4j?**

Для того, чтобы log4j начал работать нужно предоставить ему конфигурацию. Это можно сделать несколькими путями:

* Создать конфигурацию программно, т.е. получить logger, определить уровень журналирования, прикрепить appender и задать способ форматирования.
* Указать файл или URL как аргумент при запуске java-машины -Dlog4j.configuration=путь/к/файлу/конфигурации, а затем прочитать его в программе при помощи PropertyConfigurator.configure(...)/ DOMConfigurator.configure(...) для формата .properties или XML соответственно.
* Загрузить конфигурацию из файла в формате XML или .properties: log4j ищет файл конфигурации в classpath. Сначала ищется файл log4j.xml и, если таковой не найден, - файл log4j.properties.