Date / /

# Log4j and Junit ~~&~~ Mockito :

## Log4j :

Logging events is a critical aspect of s/w development.
while where are lots of framework available in java.
Log4j has been the most popular for decades due to flexibility and simplicity it provides.

Log4j2 is a new and improved version of the classic Log4j framework.

Log4j2, an appender is simply a destination for log events.
it can be as sample as console and can be complex like any RDBMS.
Layouts determine how the logs will be presented and filters filter the data according to the various criterion.

## Setup :

In order to understand several logging components and their configuration let's set up different test use-cases.
Each consisting of a log4j2.xml configuration file and Junit 4 test cases.
Two maven dependencies are common to all examples

```
<dependency>
<groupId> org.apache.logging.log4j</groupId>
<artifactId> log4j-core</artifactId>
<version> 2.19.0</version>
</dependency>
```

```
<dependency>
<groupId> org.apache.logging.log4j</groupId>
 <artifactId> log4j-core</artifactId>
 <version> 2.19.0</version>
 <type> test-jar</type>
 <scope> test</scope>
 </dependency>
```

Default configuration:

consoleAppender is the default configuration of the Log4j2 core package.

```
<?xml version="1.0" encoding ="UTF-8">
<configuration status ="WARN">
      <Appenders>
      < console
            name = "consoleAppender"
            target = "System_OUT"/>
         <PatternLayout
                  pattern ="%d [%t]
            % -5level %logger{36} - %msg%n%throwable"
         </Console>
         </Appenders>
         <Loggers>
            < Root level = "ERROR">
            <AppenderRef
   ref ="ConsoleAppender"/>
            </Root>
            </Loggers>
            </configuration>
```

Page No.
Date / /

configuration: The root element of a Log4j2 configuration file and attribute status is the level of internal Log4j events.

Appenders: This element is holding one or more appenders.
Here well configure and appendor that outputs to the system console at standard out.

Loggers: This element can concist of multiple configured logger elements with the special Root tag.
you can configure a nameless standard logger that will receive all log messages from the application.
Each logger can be sent to minimum log level.

AppendorRef: This element defines a reference to an element from the Appendor section.
Therefore the attribute 'ref' is linked with an appenders 'name' attribute

We use system.out.println to print the flow of the program onto the console during the runtime environment. system.out.pritln is often termed as heavy weight. it is because of synchronized method. this makes it expensive and time consuming.
At the time of production, we can neither use system.out.pritln nor debugging tool.
we could use debugging tool to look for any issues But in case of remote environment debugging becomes tougher as debugging tool are not meant to work for remote systems.
Log4j comes as a solution that could solve the above problems making the developers life easier.

What is Log4j

To track the flow of application and to maintain a record of the overall process we go for Log4j framework

* Highly configurable
* Logging is highly flexible and could be set to various levels

Log4j Main components
- Loggers → To register the class to logger
- Appenders → To append the logging information to the respective destination
- Layouts → Styling the logging information

Log4j features

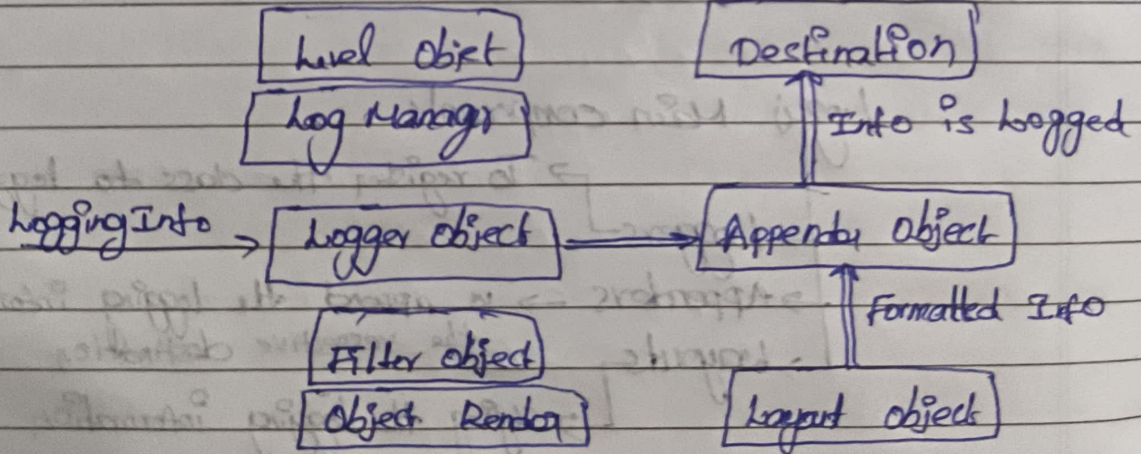
* Thread safe
* High speed optimization due to tracking capability
* Multiple appender support
* Supports I18N - Internationalization
* .xml / .properties way of configuring
* various level logging
* customized formatting of the log messages using Layout class

Advantages:

* Quick debugging
* Easy maintenance
* Structured storage

Disadvantages:

* It slows down application
* scrolling blindness due to configuration file being too verbose



core objects - Loggers, Layout, Appenders are core objects

Page No.

Date / /

Logger: It gather the logging information and are capatured in the namespace heirarchy.

Layout: styles the logging information as how we want it to be which is legible to anyone.

Appenders: Lower level object, it publishes logging information to console, file, database, sockets etc.

Support core objects:

Level : defines ~~priori~~ priority of logging information
Filter: Analyze and decide whether to keep the logged information.
Appenders could make use of multiple filters.
Filters tell the appenders whether or not to print the logs to respective destination.

Object Renderer: provides string representation of the logging information.

Log Manager: reads the configuration parameters