http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:rte3:fdl:logging:log4j_2

Description

Changes made in Log4j 2

- Java 6 or higher required.
- XML properties simplified (not compatible with Log4j 1.x).
- Configuration of property is not supported via property file.
- Configuration of property is to be made in JSON.
- Log message output is made in the form of parameter.
- Configuration changes are promptly applied without reboot.
- Improved filtering.
- Various appenders such as NoSQLApender is supported.

What are new to Log4j 2

1. Substituting Parameters

You can simply substitute parameters in {} to replace the conventional strings combination.
Refer to the following code to check for the log level before you complete the log message and how to set isDebugEnabled to execute methods:

```
logger.debug("Logging in user {} with birthday {}", user.getName(), user.getBirthdayCalendar());
```

2. Formatting Parameters

You can work formatting in your coding by using parameters logging. Generate a logger object using getFormatterLogger() and refer to the class Java.util.Formatter for characters and type of format:

```
public static Logger logger = LogManager.getFormatterLogger("egovframework");

logger.debug("Logging in user %1$s with birthday %2$tm %2$te,%2$tY", user.getName(),
user.getBirthdayCalendar());
logger.debug("Integer.MAX_VALUE = %,d", Integer.MAX_VALUE);
logger.debug("Long.MAX_VALUE = %,d", Long.MAX_VALUE);

// Output
// User John Smith with birthday 05 23, 1995
// Integer.MAX_VALUE = 2,147,483,647
// Long.MAX_VALUE = 9,223,372,036,854,775,807
```

3. Flow Tracing

Log4j 2 provides additional methods for you to better unerstan not only the logging methods such as trace(), debug() and info(), but also to the execution sequence of applications.

Method	Function	Location	Use
entry()	Indicate where a log starts. Outputs the incoming method parameters.	Where a logging method starts.	logger.entry() or logger.entry(Object… params)
exit()	Indicate where a log ends. Outputs the return.	Where a return statement or logging method ends.	logger.exit() or logger.exit(Object… result)
throwing()	Outputs the exception or error thrown.	When exception is thrown.	throw logger.throwing(new MyException);

catching()	Outputs the exception caught.	Catch statement.	logger.catching(e);

The default log level and marker are available to set the method-derived logging events apart from the basic logging events.
Outputs for the methods 'entry' and 'exit' are thus made only on TRACE level, clearly separate (filtered) from other log messages thanks to FLOW Marker. Outputs for the methods 'throwing' and 'catching' is thus made only on ERROR level, filtered from EXCEPTION Marker.

Method	Log Level	Marker
entry()	TRACE	ENTER or FLOW
exit()	TRACE	EXIT or FLOW
throwing()	ERROR	THROWING or EXCEPTION
catching()	ERROR	CATCHING or EXCEPTION

```
public String saveDept(String deptNo) {

    logger.entry(deptNo); // Declared at where a method begins. Outputs the incoming parameter.

    Dept dept = service.saveDept(deptNo);

    String nextPg = "redirect:/dept/deptList.do";

    return logger.exit(nextPg); // Declared at where a method ends. Outputs the parameter to be returned.
}

public static void main(String[] args)
    { saveDept("20");

    // Output
    // TRACE saveDept - entry(20)
    // TRACE saveDept - exit(redirect:/dept/deptList.do)
}
```

4. Markers

Markers help you understand what exactly is the problem in a multitude of logs. With the purpose of logging framework such as Log4j being confirmation and debugging of errors thrown, you need to accurately filter the log information whenever you want.

While logging method and log level configuration for Logger supports filtering of the logs for output, Log4j 2 supports detailed filtering by way of Marker function, For instance, you can simply configure marker as intended to implement separation of Flow Tracing Method from the basic logging event, or separate output of SQL statements.

Marker features:

- The unique name assigned to each Marker; and
- Declared final and is subject to a single Parent Marker.

```
public class MyClass {
  private static final Logger LOGGER = LogManager.getLogger(MyClass.class);

  // Marker name = "SQL"
  private static final Marker SQL_MARKER = MarkerManager.getMarker("SQL");
```

```
   // Marker name = "SQL_UPDATE", Parent marker = SQL_MARKER
   private static final Marker UPDATE_MARKER = MarkerManager.getMarker("SQL_UPDATE", SQL_MARKER);

   // Marker name = "SQL_QUERY", Parent marker = SQL_MARKER
   private static final Marker QUERY_MARKER = MarkerManager.getMarker("SQL_QUERY", SQL_MARKER);

   public String doQuery(String table) {

      LOGGER.entry(table);

      LOGGER.debug(QUERY_MARKER, “SELECT * FROM {}”, table); // Output in select.log

      return LOGGER.exit();
}

public String doUpdate(String table, Map<String, String> params)

      {  LOGGER.entry(table);

      LOGGER.debug(UPDATE_MARKER, "UPDATE {} SET {}", table, formatCols()); // update.log out

      return logger.exit();
}
...
<Appenders>
   <File name="fileQuery" fileName="./logs/file/select.log">
      <MarkerFilter marker="SQL_QUERY" onMatch="ACCEPT" onMismatch="DENY" />
      <PatternLayout pattern="%level %m%n" />
   </File>
   <File name="fileUpdate" fileName="./logs/file/update.log">
      <MarkerFilter marker="SQL_UPDATE" onMatch="ACCEPT" onMismatch="DENY" />
      <PatternLayout pattern="%level %m%n" />
   </File>
   ...
</Appenders>
...
```

References

Flow Tracing
Markers

Migration to Log4j 2 from Log4j 1.x

1. Adding Log4j 2 jar (log4j-api.jar, log4j-core.jar) + Removing Log4j 1.x jar

```
<!-- Log4j 1.2 -->
<!--
<dependency>
   <groupId>log4j</groupId>
   <artifactId>log4j</artifactId>
   <version>1.2</version>
</dependency>
-->

<!-- Log4j 2 -->
<dependency>
```

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

2. Log4j 1.x -> Converting Log4j 2 and adding bridge jar (log4j-1.2-api.jar)

You can add Log4j 2 Bridge to have the conventional Log4j 1.x API be converted into Log4j 2 API, as follows:

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-1.2-api</artifactId>
    <version>x.x.x</version>
</dependency>
```

3. Changing Logger API of Log4j 1.x

You need to change the logger generator method for Log4j 2 to generate the logger object, as follows:

	Log4j 1.x	**Log4j 2**
Package	**org.apache.log4j**	**org.apache.logging.log4j**
Generating Logger objects	org.apache.log4j.Logger logger = org.apache.log4j.**Logger.getLogger();**	org.apache.logging.log4j.Logger logger = org.apache.logging.log4j.**LogManager.getLogger();**

4. Adding Log4j 2 configuration (log4j2.xml)

You have a number of intuitive configuration tags in Log4j 2. See Log4j 2 Configuration Details for more information.

- Log4j 1.x

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration PUBLIC "-//APACHE//DTD LOG4J 1.2//EN" "log4j.dtd">
<log4j:configuration xmlns:log4j='http://jakarta.apache.org/log4j/'>
    <appender name="STDOUT" class="org.apache.log4j.ConsoleAppender">
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%d %-5p [%t] %C{2} (%F:%L) - %m%n"/>
        </layout>
    </appender>
    <category name="org.apache.log4j.xml">
        <priority value="info" />
    </category>
    <Root>
        <priority value ="debug" />
        <appender-ref ref="STDOUT" />
    </Root>
</log4j:configuration>
```

- Log4j 2

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Configuration>
   <Appenders>
      <Console name="STDOUT" target="SYSTEM_OUT">
         <PatternLayout pattern="%d %-5p [%t] %C{2} (%F:%L) - %m%n"/>
      </Console>
   </Appenders>
   <Loggers>
      <Logger name="org.apache.log4j.xml" level="info"/>
      <Root level="debug">
         <AppenderRef ref="STDOUT"/>
      </Root>
   </Loggers>
</Configuration>
```

References

Migration to Log4j 2
Log4j 2 API Documentation
Log4j 2 Implementation Documentation