**Now, what will happen if no configuration is given?**

Log4j has the ability to automatically configure itself during initialization. When Log4j starts it will locate all the ConfigurationFactory plugins and arrange them in weighted order from highest to lowest. As delivered, Log4j contains four ConfigurationFactory implementations: one for JSON, one for YAML, one for properties, and one for XML.

1. Log4j will inspect the **log4j.confiugrationFile** system property and, if set, will attempt to load the configuration using the **configurationFactory** that matches the file extension.
2. If no system property is set the properties ConfigurationFactory will look for **log4j2-test.properties** in the classpath.
3. If no such file is found the YAML ConfigurationFactory will look for **log4j2-test.yaml** or **log4j2-test.yml** in the classpath.
4. If no such file is found the JSON ConfigurationFactory will look for **log4j2-test.json** or **log4j2-test.jsn** in the classpath.
5. If no such file is found the XML ConfigurationFactory will look for **log4j2-test.xml** in the classpath.
6. If a test file cannot be located the properties ConfigurationFactory will look for **log4j2.properties** on the classpath.
7. If a properties file cannot be located the YAML ConfigurationFactory will look for **log4j2.yaml** or **logj2.yml** on the classpath.
8. If a YAML file cannot be located the JSON ConfigurationFactory will look for **log4j2.json** or **logj2.jsn** on the classpath.
9. If a JSON file cannot be located the XML ConfigurationFactory will try to locate **log4j2.xml** on the classpath.
10. If no configuration file could be located the **DefaultConfiguration** will be used. This will cause logging output to go to the console.

Now, according to defaultConfiguration, logging output will go to the cosole.

Log level will be set to error.

And pattern associated to consoleAppender is set to following:

**%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n**

I.e. it would do the following:

* A [ConsoleAppender](https://logging.apache.org/log4j/2.x/log4j-core/apidocs/org/apache/logging/log4j/core/appender/ConsoleAppender.html) attached to the root logger.
* A [PatternLayout](https://logging.apache.org/log4j/2.x/log4j-core/apidocs/org/apache/logging/log4j/core/layout/PatternLayout.html) set to the pattern "%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" attached to the ConsoleAppender

This will print log like this:

02:31:30.149 [main] FATAL newmongodbaccessor.MongoDBHandler - Authentication is false

02:31:30.150 [main] FATAL newmongodbaccessor.MongoDBHandler - It could happen for multiple reasons. Including wrong IP and password

02:31:30.151 [main] FATAL newmongodbaccessor.NewMongoDBAccessor - MongoDBHandler cannot be instantiated

02:31:30.155 [main] ERROR newmongodbaccessor.MongoDBHandler - The table does not exist into database: newServerApplicationDB

Meaning of formats specified in layout

**%d{HH:mm:ss.SSS} prints hour, minute, second and milliseconds in the specified format.**

**[%t]: prints the thread name in the specified format.**

**%-5level: this will print the log level in five characters.**

**%logger{36} It would print the class name from where the logging request is generated.**

**%msg or %m prints the message**

**%n prints the newline.**