By default the root logger is assigned the DEBUG level.

The root configuration is inherited by children logger configurations and can be overriden.

Reference URL: <https://logback.qos.ch/manual/configuration.html>

**Configuring loggers, or the <logger> element**

At this point you should have at least some understanding of [level inheritance](https://logback.qos.ch/manual/architecture.html#effectiveLevel) and the [basic selection rule](https://logback.qos.ch/manual/architecture.html#basic_selection). Otherwise, and unless you are an Egyptologist, logback configuration will be no more meaningful to you than are hieroglyphics.

A logger is configured using the <logger> element. A <logger> element takes exactly one mandatory *name* attribute, an optional *level* attribute, and an optional *additivity* attribute, admitting the values *true* or *false*. The value of the *level* attribute admitting one of the case-insensitive string values TRACE, DEBUG, INFO, WARN, ERROR, ALL or OFF. The special case-insensitive value *INHERITED*, or its synonym *NULL*, will force the level of the logger to be inherited from higher up in the hierarchy. This comes in handy if you set the level of a logger and later decide that it should inherit its level.

The <logger> element may contain zero or more <appender-ref> elements; each appender thus referenced is added to the named logger. Note that unlike log4j, logback-classic does *not* close nor remove any previously referenced appenders when configuring a given logger.

#### Configuring the root logger, or the <root> element

The <root> element configures the root logger. It supports a single attribute, namely the *level* attribute. It does not allow any other attributes because the additivity flag does not apply to the root logger. Moreover, since the root logger is already named as "ROOT", it does not allow a name attribute either. The value of the level attribute can be one of the case-insensitive strings TRACE, DEBUG, INFO, WARN, ERROR, ALL or OFF. Note that the level of the root logger cannot be set to INHERITED or NULL.

Similarly to the <logger> element, the <root> element may contain zero or more <appender-ref> elements; each appender thus referenced is added to the root logger. Note that unlike log4j, logback-classic does not close nor remove any previously referenced appenders when configuring the root logger.

**Example**

Setting the level of a logger or root logger is as simple as declaring it and setting its level, as the next example illustrates. Suppose we are no longer interested in seeing any DEBUG messages from any component belonging to the "chapters.configuration" package. The following configuration file shows how to achieve that.

*Example: Setting the level of a logger (logback-examples/src/main/resources/chapters/configuration/sample2.xml)*

<configuration>  
  
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">  
    <!-- encoders are assigned the type  
         ch.qos.logback.classic.encoder.PatternLayoutEncoder by default -->  
    <encoder>  
      <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>  
    </encoder>  
  </appender>  
  
  **<logger name="chapters.configuration" level="INFO"/>**  
  
  <!-- Strictly speaking, the level attribute is not necessary since -->  
  <!-- the level of the root level is set to DEBUG by default.       -->  
  <root level="DEBUG">            
    <appender-ref ref="STDOUT" />  
  </root>    
    
</configuration>

When the above configuration file is given as argument to the *MyApp3* application, it will yield the following output:

17:34:07.578 [main] INFO chapters.configuration.MyApp3 - Entering application.

17:34:07.578 [main] INFO chapters.configuration.MyApp3 - Exiting application.

Note that the message of level DEBUG generated by the ["chapters.configuration.Foo"](https://logback.qos.ch/xref/chapters/configuration/Foo.html) logger has been suppressed. See also the Foo class.

You can configure the levels of as many loggers as you wish. In the next configuration file, we set the level of the*chapters.configuration* logger to INFO but at the same time set the level of the *chapters.configuration.Foo* logger to DEBUG.

*Example: Setting the level of multiple loggers (logback-examples/src/main/resources/chapters/configuration/sample3.xml)*

<configuration>  
  
  <appender name="STDOUT"  
    class="ch.qos.logback.core.ConsoleAppender">  
    <encoder>  
      <pattern>  
        %d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n  
     </pattern>  
    </encoder>  
  </appender>  
  
  **<logger name="chapters.configuration" level="INFO" />**  
  **<logger name="chapters.configuration.Foo" level="DEBUG" />**  
  
  <root level="DEBUG">  
    <appender-ref ref="STDOUT" />  
  </root>  
  
</configuration>

Running MyApp3 with this configuration file will result in the following output on the console:

17:39:27.593 [main] INFO chapters.configuration.MyApp3 - Entering application.

17:39:27.593 [main] DEBUG chapters.configuration.Foo - Did it again!

17:39:27.593 [main] INFO chapters.configuration.MyApp3 - Exiting application.

The table below list the loggers and their levels, after JoranConfigurator has configured logback with the *sample3.xml*configuration file.

|  |  |  |
| --- | --- | --- |
| **Logger name** | **Assigned Level** | **Effective Level** |
| root | DEBUG | DEBUG |
| chapters.configuration | INFO | INFO |
| chapters.configuration.MyApp3 | null | INFO |
| chapters.configuration.Foo | DEBUG | DEBUG |

It follows that the two logging statements of level INFO in the MyApp3 class as well as the DEBUG messages in Foo.doIt()are all enabled. Note that the level of the root logger is always set to a non-null value, DEBUG by default.

*Example: Logger level sample (logback-examples/src/main/resources/chapters/configuration/sample4.xml)*

<configuration>  
  
  <appender name="STDOUT"  
   class="ch.qos.logback.core.ConsoleAppender">  
   <encoder>  
     <pattern>  
        %d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n  
      </pattern>  
    </encoder>  
  </appender>  
  
  **<logger name="chapters.configuration" level="INFO" />**  
  
  <!-- turn OFF all logging (children can override) -->  
  <root **level="OFF"**>  
    <appender-ref ref="STDOUT" />  
  </root>  
  
</configuration>

The following table lists the loggers and their levels after applying the *sample4.xml* configuration file.

|  |  |  |
| --- | --- | --- |
| **Logger name** | **Assigned Level** | **Effective Level** |
| root | OFF | OFF |
| chapters.configuration | INFO | INFO |
| chapters.configuration.MyApp3 | null | INFO |
| chapters.configuration.Foo | null | INFO |

The ConsoleAppender named *STDOUT*, the only configured appender in *sample4.xml*, is attached to the root logger whose level is set to OFF. However, running *MyApp3* with configuration script *sample4.xml* will yield:

17:52:23.609 [main] INFO chapters.configuration.MyApp3 - Entering application.

17:52:23.609 [main] INFO chapters.configuration.MyApp3 - Exiting application.

Thus, the level of the root logger has no apparent effect because the loggers in chapters.configuration.MyApp3 andchapters.configuration.Foo classes are all enabled for the INFO level. As a side note, the *chapters.configuration* logger exists by virtue of its declaration in the configuration file - even if the Java source code does not directly refer to it.

**Appenders accumulate**

By default, **appenders are cumulative**: a logger will log to the appenders attached to itself (if any) as well as all the appenders attached to its ancestors. Thus, attaching the same appender to multiple loggers will cause logging output to be duplicated.

*Example: Duplicate appender (logback-examples/src/main/resources/chapters/configuration/duplicate.xml)*

<configuration>  
  
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">  
    <encoder>  
      <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>  
    </encoder>  
  </appender>  
  
  <logger name="chapters.configuration">  
    <appender-ref ref="STDOUT" />  
  </logger>  
  
  <root level="debug">  
    <appender-ref ref="STDOUT" />  
  </root>  
</configuration>

Running MyApp3 with *duplicate.xml* will yield the following output:

14:25:36.343 [main] INFO chapters.configuration.MyApp3 - Entering application.

14:25:36.343 [main] INFO chapters.configuration.MyApp3 - Entering application.

14:25:36.359 [main] DEBUG chapters.configuration.Foo - Did it again!

14:25:36.359 [main] DEBUG chapters.configuration.Foo - Did it again!

14:25:36.359 [main] INFO chapters.configuration.MyApp3 - Exiting application.

14:25:36.359 [main] INFO chapters.configuration.MyApp3 - Exiting application.

Notice the duplicated output. The appender named *STDOUT* is attached to two loggers, to root and to*chapters.configuration*. Since the root logger is the ancestor of all loggers and *chapters.configuration* is the parent of both *chapters.configuration.MyApp3* and *chapters.configuration.Foo*, each logging request made with these two loggers will be output twice, once because *STDOUT* is attached to *chapters.configuration* and once because it is attached to *root*.