NDCvsMDC

NDC vs MDC - Which one should I use?

The NDC and MDC log4j classes are used to store program/application contextual information that can then be used when logging messages. The NDC class name is org.apache.log4j.NDC. "NDC" stands for "Nested Diagnostic Context". The MDC class name is org.apache.log4j.MDC. "MDC" stands for "Mapped Diagnostic Context". NDC has been part of the log4j framework longer than MDC. If you haven't already, you may want to review the javadoc information for each class.

NDC

The "Nested Diagonostic Context" implements a "stack" onto which context information can be pushed and popped (ie "nested"). The context is stored per thread, so different threads can have different context information. When a program entered section "A" of its code, it could use NDC.push() to put the string "A" into the context. When it exited section "A", it would then NDC.pop() to remove "A" from the context. As you can see, you can continue to push /pop contexts. It is up to the application to make sure that the proper NDC.pop() call is made for each NDC.push().

When a message is logged, the current contents of the NDC are attached to it, and can be displayed in the log messages by using the '%x' option in PatternLayout. In this way, information specific to the context of a particular thread can be displayed in the log.

The beauty of this is that the logger sending the message does not have any clue about the context or contents of the NDC, and it doesn't need to. But appenders and filters can use the NDC information in the log message to affect the routing and display of log message. Besides the '%x' option in

```
PatternLayout
```

, a new log4j filter for v1.3 (see

```
org.apache.log4j.filters.NDCMatchFilter
```

in the current cvs) will accept or deny a log message based on the contents of the NDC information.

MDC

The "Mapped Diagnostic Context" implements a "map" into which key/value pair information can be stored. Just like NDC, the context is stored per thread. Values are stored by key name. Each thread could use the same key name but have different stored values. Values are stored/retreived/removed by using the familiar pattern of MDC.put(), MDC.get(), and MDC.remove() methods.

When a message is logged, the current contents of the MDC are attached to it, and can be displayed in the log messages by using the '%X' option in PatternLayout. More than one MDC value can be displayed in a single log message.

Just as with NDC, appenders and filters can use the MDC information attached to a log message for display and routing. Log4j v1.3 will contain a filter based on the contents of the MDC (see

```
org.apache.log4j.filters.MDCMatchFilter
```

in the current cvs).

Which one to use?

Now that you have some idea of how the NDC and MDC store context information, it should be straight forward to choose which one to use. If nested /stack like information is important when logging information, use NDC. If key/value pair information is more appropriate, use MDC.

Known Gotchas

- MDC requires JDK 1.2 or later. It is not compatible with JDK 1.1, unlike NDC which is.
- NDC use can lead to memory leaks if you do not periodically call the NDC.remove() method. The current NDC implementation maintains a static hard link to the thread for which it is storing context. So, when the thread is released by its creator, the NDC maintains the link and the thread (and its related memory) is not released and garbage collected like one might expect. NDC.remove() fixes this by periodically checking the threads referenced by NDC and releasing the references of "dead" threads. But, you have to write your code to call NDC.remove().
- On Websphere or when using the IBM JVM an Exception might be thrown. Refer to this link (http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg1PQ80288) for the necessary patch.

So, give both NDC and MDC a try. Write some test code to set various values and log messages to see how the output changes. NDC and MDC are powerful tools for logging that no log4j user should be ignorant of.