Limitations

by

Table of contents

1. Matching scope names

The ScopingAppender is designed to be reasonably tolerant of mismatches of scope names:

- if a start event is not captured (eg is lost in UDP), then its matching finish event will just be discarded.
- Conversely, if a finish event is lost, then the next finish event will walk up the call stack looking for a scope to close. The inner scope will automatically be finished (moreover, it will be marked as having been finished due to a mismatch)

However, if the final finish event goes missing then this algorithm breaks. The way this is represented in the Browser/Collator is that the scope remains hanging and must be closed manually. This in turn causes the XML file to be written out at which point the developer can look to see why the scope was not closed.

The alternative (and in fact original) design would have been to make it the application's responsibility to indicate in some definitive way that an outermost scope has been completed. However, this also makes the system more brittle: it isn't then possible to arbitrarily refactor the application because one must keep in mind where these end points are. We decided that it's better to have hanging scopes because it points to an (easily rectifiable) error in the code.

2. All-or-nothing

Currently there is no capability to switch the Lightweight appender on-or-off dynamically. It also isn't possible to change the hostname where it is sending its UDP messages.

3. Session definitions

Currently session definitions (of names/ports/directories) are not persisted. So, although it is possible to start monitoring a single session using command line options, it is currently a manual process to set up more than one such session.