Using the Scoping Appender

by

Table of contents

1. Simple Scoping

The following example assumes that you are using the StandardLoggingEventInterpretor.

```
public class WebServiceFacade {
    private Logger LOG = Logger.getLogger("Web"); // used as category
    public void invoke() {
        String method = "invoke";
        LOG.info("scoped://start:" + getCurrentUserFromThread() + "@" +
method + "@id="+id);
        try {
            ...
            _dbFacade.lookupById(id);
            ...
        } finally {
            LOG.info("scoped://finish:" + getCurrentUserFromThread() +
"@" + method);
        }
    }
}
public class DatabaseFacade {
    private Logger LOG = Logger.getLogger("DB"); // used as category
    public void lookupById(String id) {
        String method = "lookupById";
        LOG.info("scoped://start:" + getCurrentUserFromThread() + "@" +
method + "@id="+id);
        try {
            ...
        } finally {
            LOG.info("scoped://finish:" + getCurrentUserFromThread() +
"@" + method);
        }
    }
}
```

2. Scoping with Error Termination

The following example again assumes that you are using the StandardLoggingEventInterpretor. It differs in that we capture whether the scope finished as a result of an error or not.

```
public class WebServiceFacade {
    private Logger LOG = Logger.getLogger("Web");
    public void invoke() {
        String method = "invoke";
        LOG.info("scoped://start:" + getCurrentUserFromThread() + "@" +
method + "@id="+id);
        try {
            ...
            _dbFacade.lookupById(id);
            ...
            LOG.info("scoped://finish:" + getCurrentUserFromThread() +
"@" + method);
        } catch(Throwable t) {
            LOG.info("scoped://error:" + getCurrentUserFromThread() +
"@" + method);
            throw t;
```

```
        }
    }
}
```