Table of Contents

Ohne Log Mechanismen ist die Software Entwicklung bei weitem anstrengender als es sein müsste. Aus diesem Grunde ist bereits im Kern von JVx ein Log Mechanismus verankert.

Durch UIComponent werden folgende Methoden bereitgestellt:

```
public void info(Object... pInfo)
public void debug(Object... pInfo)
public void error(Object... pInfo)
```

Mit Informationen (info) verknüpfen wir Ausgaben die nur für Entwickler, während der Applikationserstellung, relevant sind. Die Debug Ausgaben können auch auch für Endanwender hilfreich sein. Die Ausgabe von Fehlern (Error) ist für die Problembehebung von großem Vorteil und sollte im produktiven Betrieb unbedingt genutzt werden.

Dadurch ist bei der Entwicklung von Applikationen, WorkScreens oder Komponenten ein einfacher Methodenaufruf nötig um das Logging zu verwenden. Es müssen keine speziellen Klassen verwendet werden.

JVx Logging

Der Log Mechanismus von JVx definiert kein neues Logging API, doch sehr wohl eine Schnittstelle für die Integration von beliebigen Logging Frameworks.

Die Schnittstelle wird definiert durch com.sibvisions.util.log.ILogger und verwendet mit com.sibvisions.util.log.LoggerFactory.

In JVx ist bereits eine Implementierung für das Java Logging API enthalten!

Um das Logging von JVx unabhängig vom UI zu verwenden sind folgende Schritte notwendig:

Erstellen eines Loggers:

```
ILogger log = LoggerFactory.getInstance(...);
```

Logger verwenden:

```
log.info(...);
```

Die Standard Implementierung von JVx geht äußerst Sparsam mit den Loggern um. Die tatsächliche Instanz eines Loggers (ILogger) wird erst initialisiert, wenn darauf zugegriffen wird, sprich eine Log Ausgabe erfolgt.

Die Implementierung von vararg Parametern (Object...) wirkt sich ebenfalls positiv auf die Performance und auf die Lesbarkeit des Source Codes aus. Für gewöhnlich sind Konstrukte wie:

```
otherlogger.log("Row number: " + nr +" of " + count);
```

oder

```
if (otherlogger.isLoggable(Level))
{
```

```
    otherlogger.log("Row number: " + nr +" of " + count);
}
```

nicht unüblich. Der erste Aufruf verursacht in jedem Fall mehrere String Operationen, auch wenn der Logger die Meldung nicht ausgibt. Beim zweiten Aufruf werden diese String Operationen bewusst vermieden, machen den Source Code aber länger.

Mit JVx würde die Log Ausgabe wie folgt codiert werden:

```
log.debug("Row number: ", nr, " of ", count);
```

Zugegeben findet hier eine Array Operation statt, doch diese wirkt sich - im Gegensatz zu mehreren String Operationen - unwesentlich auf die Performance aus.

Die Konfiguration der Logger ist abhängig vom eingesetzten Logging Framework. Im Falle des Java Logging API kann die Datei logging.properties verwendet werden:

logging.properties

```
############################################################
#
# HANDLER definition
############################################################
#

# development handlers
handlers = java.util.logging.ConsoleHandler

# application handlers
#handlers = java.util.logging.FileHandler

############################################################
#
# HANDLER configuration
############################################################
#

java.util.logging.ConsoleHandler.level = ALL
java.util.logging.ConsoleHandler.formatter =
com.sibvisions.util.log.jdk.JdkLineFormatter

java.util.logging.FileHandler.level = ALL
java.util.logging.FileHandler.pattern = application_%g.log
java.util.logging.FileHandler.limit = 10000
java.util.logging.FileHandler.count = 5
java.util.logging.FileHandler.formatter =
com.sibvisions.util.log.jdk.JdkLineFormatter

############################################################
#
```

```
# Package specifig log levels
#########################################################################
#

.level = OFF

#com.sibvisions.level = OFF
#com.sibvisions.rad.model.level = OFF
#com.sibvisions.rad.persist.level = ALL
#com.sibvisions.rad.server.level = OFF

#jvx.rad.level = ALL
```

Für die Veränderung von Log Levels können auch folgende Methoden hilfreich sein:

```
log.setLevel(level);

LoggerFactory.setLevel(name, level);
```

Die Log Implementierung von JVx ist sowohl für RIAs, Desktop, Web und Android Anwendungen geeignet. Im Falle von WebUI Anwendungen ist zu beachten, daß die Log Ausgaben am Server stattfinden.