Logging Frameworks

IDEE, KONZEPTE, LÖSUNGEN IM UMFELD VON JAVA

© BY RENÉ ANDERES

Problemstellung

Welche Informationen?

Wie können die Informationen strukturiert werden?

Wohin sollen die Informationen geschrieben werden?

© BY RENÉ ANDERES

Anforderungen

- einfach und "unaufdringlich"
- zentrale Konfiguration ohne Code-Anpassungen
- Ausgabeziele jederzeit frei und zentral konfigurierbar
- keinen Einfluss auf das Programm (z.B. Performance)
- für unterschiedliche Module unterschiedliche konfigurierbar

© BY RENÉ ANDERES

Lösungsmöglichkeiten

MAKE OR BUY

© BY RENÉ ANDERES

Existierende Lösungen

© BY RENÉ ANDERES

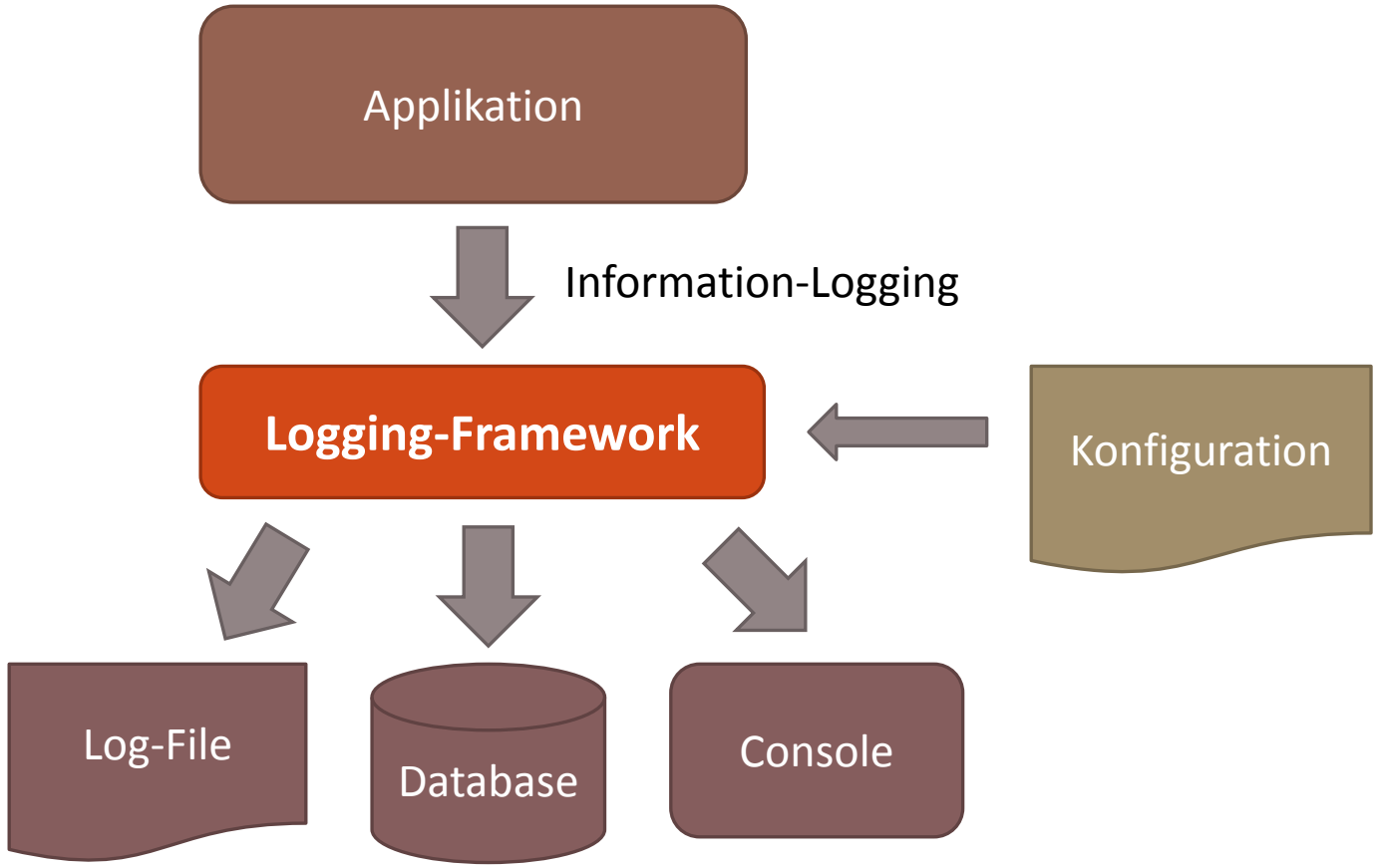
Existierende Lösungen

commons
logging

ORACLE
Java™ Logging

Apache
LOG4J 2™

Grundsätzliche Lösungskonzepte



© BY RENÉ ANDERES

Log4j 2

© BY RENÉ ANDERES

Log4j 2

Instanzieren eines Logger-Interface:

```
private Logger logger = LogManager.getLogger("Name");
```

Name des Loggers

Nutzung des Logger-Interface:

```
logger.trace("Start Transaktion '{}'", transactionName);
```

Log-Level

Platzhalter

Parameter

siehe API

© BY RENÉ ANDERES

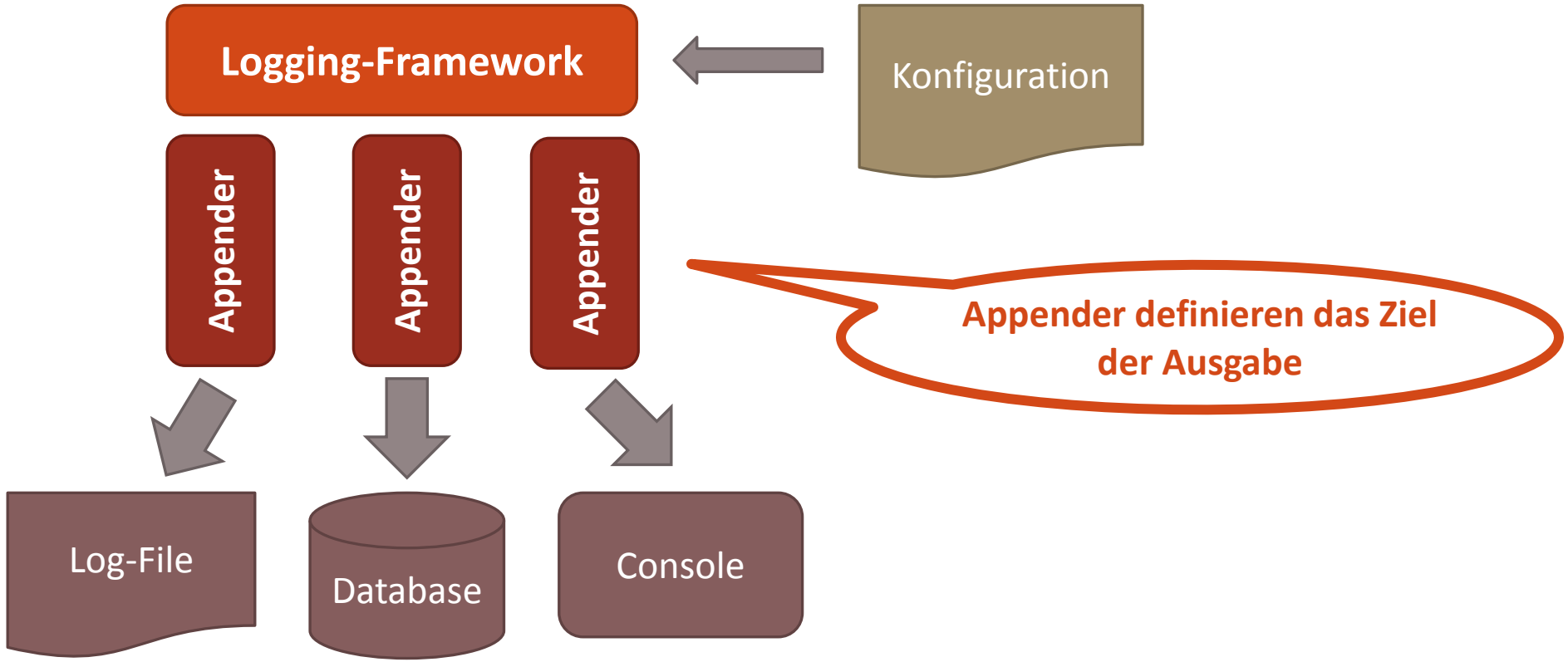
Log4j 2 – Hierarchie

```
root
   |-- ch.abraxas.lernwerkstatt
   |         |-- ch.abraxas.lernwerkstatt.nextstep
   |         |         |-- ch.abraxas.lernwerkstatt.nextstep.main
   |         |-- ch.abraxas.lernwerkstatt.bbm
   |-- java.lang.double
   |-- myLoggerName
```

```
private Logger logger = LogManager.getLogger("myLoggerName");

private Logger logger = LogManager.getLogger("ch.abraxas.lernwerkstatt.nextstep.main");
```

© BY RENÉ ANDERES

Log4j 2 – Konfiguration



© BY RENÉ ANDERES

Log4j 2 – Konfiguration

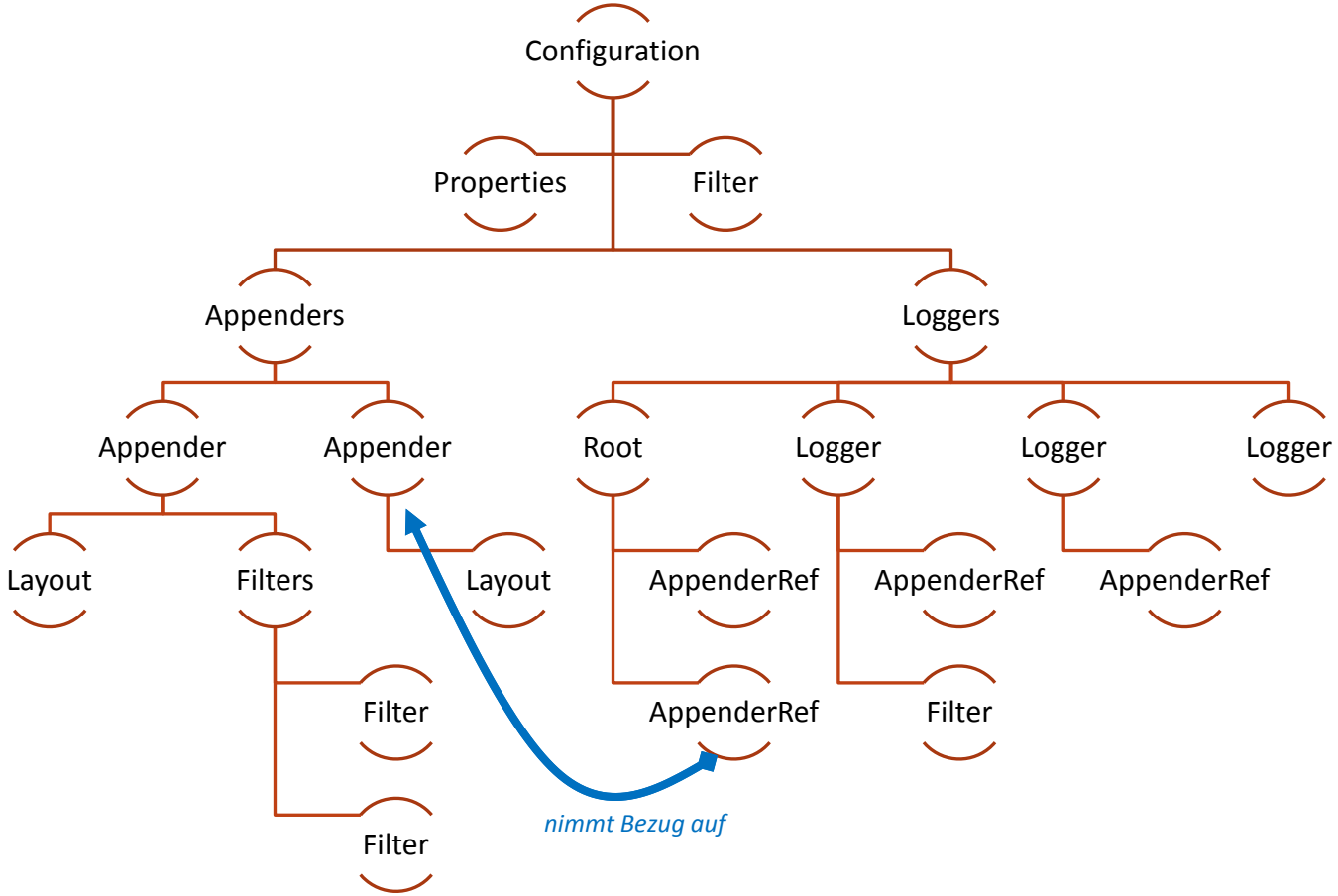
XML-Konfiguration

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="error">

    <Appenders>
        <Console name="Console" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] [%X] %x %-5level %logger{36} - %msg%n" />
        </Console>
    </Appenders>

    <Loggers>
        <Root level="all">
           <AppenderRef ref="Console"/>
        </Root>
    </Loggers>

</Configuration>
```

© BY RENÉ ANDERES

Log4j 2 – Konfiguration

Appenders



© BY RENÉ ANDERES

Aufgaben I

Log4j 2 im Einsatz

- Maven Dependencies einbauen
- System.out.print… umbauen auf Logger
  - Unterschiedliche Log-Level
  - Unterschiedliche Logger-Namen
- Log4j konfigurieren
  - Ausgabe in Konsole
    - Console-Appender
  - Ausgabe (minimal):
    - Zeit, Log-Level, Logger-Name, Meldung
    - Pattern-Layout

© BY RENÉ ANDERES

Aufgaben II

Log4j 2 im Einsatz

- Zusätzlicher Appender
  - zusätzlich in ein File schreiben
  - File-Appender
  - Rolling-File-Appender
    - bei jedem Start der App bzw. der Junit-Test's soll das bestehende File in eine ZIP Datei gepackt werden und ein neues Log-File erstellt werden.

- Die Logs der Klasse `MeasuredValuesImport` nicht in das File schreiben, nur ein Ausgabe auf die Konsole
  - siehe Logger-Hierarchie & Additivity

© BY RENÉ ANDERES

Aufgaben III

Log4j 2 im Einsatz

- Wird die Applikation mittels JUnit-Test's getestet, sollen alle Ausgaben nur an die Konsole erfolgen. Wird die Applikation normal gestartet, sollen alle Logs in ein File geschrieben werden.
  - siehe "Automatic Configuration"
- In ein Log-File sollen nur Meldung ab Log-Level "INFO" geschrieben werden.

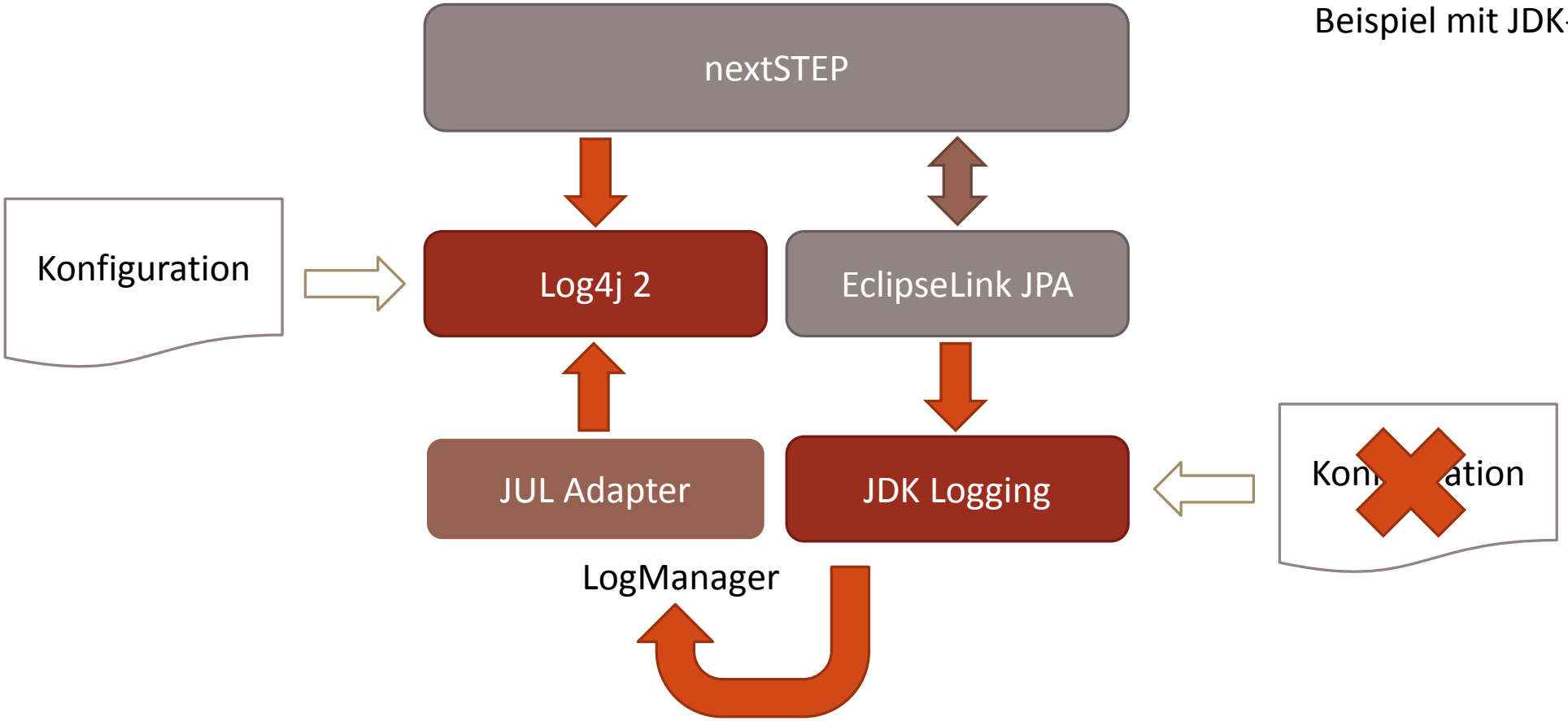
Event Level	LoggerConfig Level						
	TRACE	DEBUG	INFO	WARN	ERROR	FATAL	OFF
ALL	YES	YES	YES	YES	YES	YES	NO
TRACE	YES	NO	NO	NO	NO	NO	NO
DEBUG	YES	YES	NO	NO	NO	NO	NO
INFO	YES	YES	YES	NO	NO	NO	NO
WARN	YES	YES	YES	YES	NO	NO	NO
ERROR	YES	YES	YES	YES	YES	NO	NO
FATAL	YES	YES	YES	YES	YES	YES	NO
OFF	NO	NO	NO	NO	NO	NO	NO

© BY RENÉ ANDERES

Log4j 2 – Bridge

Log4j 2 – and other Logging-Frameworks

Beispiel mit JDK-Logger



© BY RENÉ ANDERES

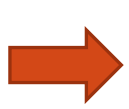
Log4j 2 – and other Logging-Frameworks

[SLF4J Binding]

commons logging

[Commons Logging Bridge]

ORACLE
Java™ Logging



[JDK Logging Adapter]

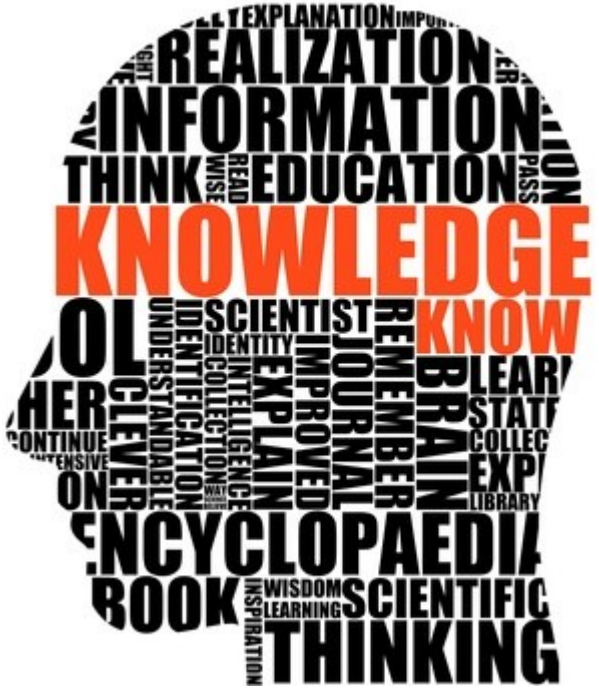
[Log4j 1.2 Bridge]

© BY RENÉ ANDERES

Code-Walkthrough

Anaylse durch bestehenden Code ….

© BY RENÉ ANDERES



Logging Framework

Log4j 2

© BY RENÉ ANDERES