



Wer spricht denn da?

Überblick über das Logging in Java

Sebastian Hempel · 21.09.2022



Sebastian Hempel

IT-Consulting Hempel

- selbständiger Software-Entwickler seit 2003
- Java, Puppet
- Linux und OpenSource
- <https://it-hempel.de>



Warum Logging?

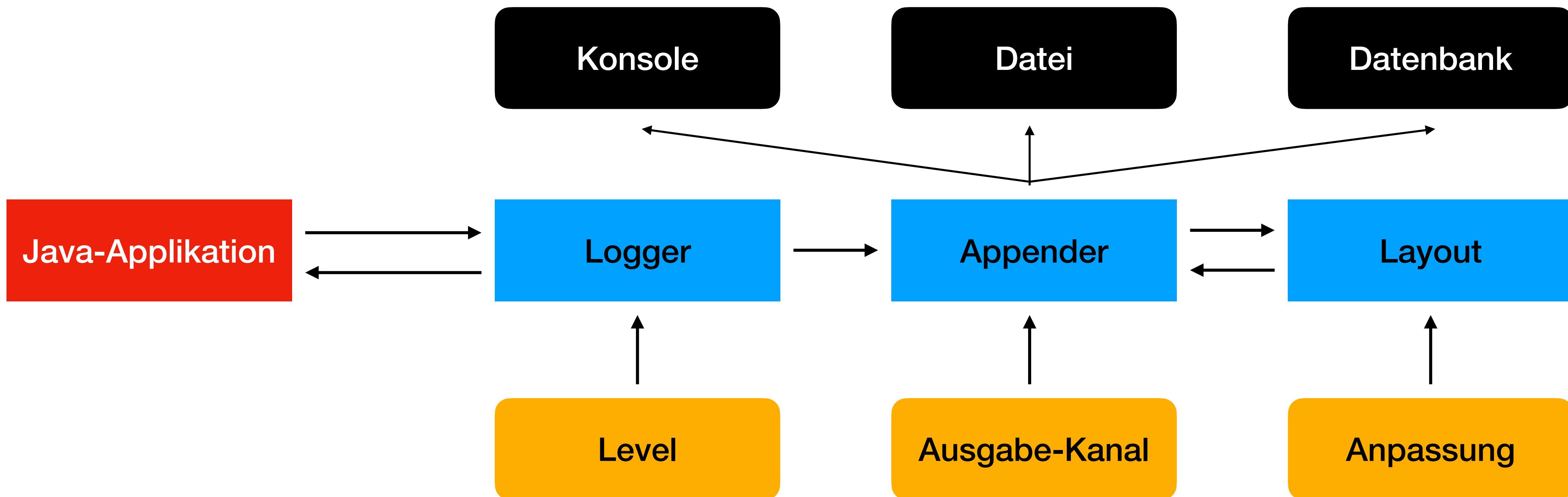
Warum Logging?

- Wie geht es meiner Applikation?
- Warum funktioniert etwas nicht?

Langedal skulekrins fra 1863		1863	
ne eler	8.9	1.1	1.1
g Navn	9.5	2.1	2.1
Barne	10.5	3.1	3.1
olen høst	11.5	4.1	4.1
Parson	12.5	5.1	5.1
og hvem	13.5	6.1	6.1
Opholdsted	14.5	7.1	7.1
den hol	15.5	8.1	8.1
med i	16.5	9.1	9.1
rebetine	17.5	10.1	10.1
	18.5	11.1	11.1
	19.5	12.1	12.1
	20.5	13.1	13.1
	21.5	14.1	14.1
	22.5	15.1	15.1
	23.5	16.1	16.1
	24.5	17.1	17.1
	25.5	18.1	18.1
	26.5	19.1	19.1
	27.5	20.1	20.1
	28.5	21.1	21.1
	29.5	22.1	22.1
	30.5	23.1	23.1
	31.5	24.1	24.1
	32.5	25.1	25.1
	33.5	26.1	26.1
	34.5	27.1	27.1
	35.5	28.1	28.1
	36.5	29.1	29.1
	37.5	30.1	30.1
	38.5	31.1	31.1
	39.5	32.1	32.1
	40.5	33.1	33.1
	41.5	34.1	34.1
	42.5	35.1	35.1
	43.5	36.1	36.1
	44.5	37.1	37.1
	45.5	38.1	38.1
	46.5	39.1	39.1
	47.5	40.1	40.1
	48.5	41.1	41.1
	49.5	42.1	42.1
	50.5	43.1	43.1
	51.5	44.1	44.1
	52.5	45.1	45.1
	53.5	46.1	46.1
	54.5	47.1	47.1
	55.5	48.1	48.1
	56.5	49.1	49.1
	57.5	50.1	50.1
	58.5	51.1	51.1
	59.5	52.1	52.1
	60.5	53.1	53.1
	61.5	54.1	54.1
	62.5	55.1	55.1
	63.5	56.1	56.1
	64.5	57.1	57.1
	65.5	58.1	58.1
	66.5	59.1	59.1
	67.5	60.1	60.1
	68.5	61.1	61.1
	69.5	62.1	62.1
	70.5	63.1	63.1
	71.5	64.1	64.1
	72.5	65.1	65.1
	73.5	66.1	66.1
	74.5	67.1	67.1
	75.5	68.1	68.1
	76.5	69.1	69.1
	77.5	70.1	70.1
	78.5	71.1	71.1
	79.5	72.1	72.1
	80.5	73.1	73.1
	81.5	74.1	74.1
	82.5	75.1	75.1
	83.5	76.1	76.1
	84.5	77.1	77.1
	85.5	78.1	78.1
	86.5	79.1	79.1
	87.5	80.1	80.1
	88.5	81.1	81.1
	89.5	82.1	82.1
	90.5	83.1	83.1
	91.5	84.1	84.1
	92.5	85.1	85.1
	93.5	86.1	86.1
	94.5	87.1	87.1
	95.5	88.1	88.1
	96.5	89.1	89.1
	97.5	90.1	90.1
	98.5	91.1	91.1
	99.5	92.1	92.1
	100.5	93.1	93.1
	101.5	94.1	94.1
	102.5	95.1	95.1
	103.5	96.1	96.1
	104.5	97.1	97.1
	105.5	98.1	98.1
	106.5	99.1	99.1
	107.5	100.1	100.1
	108.5	101.1	101.1
	109.5	102.1	102.1
	110.5	103.1	103.1
	111.5	104.1	104.1
	112.5	105.1	105.1
	113.5	106.1	106.1
	114.5	107.1	107.1
	115.5	108.1	108.1
	116.5	109.1	109.1
	117.5	110.1	110.1
	118.5	111.1	111.1
	119.5	112.1	112.1
	120.5	113.1	113.1
	121.5	114.1	114.1
	122.5	115.1	115.1
	123.5	116.1	116.1
	124.5	117.1	117.1
	125.5	118.1	118.1
	126.5	119.1	119.1
	127.5	120.1	120.1
	128.5	121.1	121.1
	129.5	122.1	122.1
	130.5	123.1	123.1
	131.5	124.1	124.1
	132.5	125.1	125.1
	133.5	126.1	126.1
	134.5	127.1	127.1
	135.5	128.1	128.1
	136.5	129.1	129.1
	137.5	130.1	130.1
	138.5	131.1	131.1
	139.5	132.1	132.1
	140.5	133.1	133.1
	141.5	134.1	134.1
	142.5	135.1	135.1
	143.5	136.1	136.1
	144.5	137.1	137.1
	145.5	138.1	138.1
	146.5	139.1	139.1
	147.5	140.1	140.1
	148.5	141.1	141.1
	149.5	142.1	142.1
	150.5	143.1	143.1
	151.5	144.1	144.1
	152.5	145.1	145.1
	153.5	146.1	146.1
	154.5	147.1	147.1
	155.5	148.1	148.1
	156.5	149.1	149.1
	157.5	150.1	150.1
	158.5	151.1	151.1
	159.5	152.1	152.1
	160.5	153.1	153.1
	161.5	154.1	154.1
	162.5	155.1	155.1
	163.5	156.1	156.1
	164.5	157.1	157.1
	165.5	158.1	158.1
	166.5	159.1	159.1
	167.5	160.1	160.1
	168.5	161.1	161.1
	169.5	162.1	162.1
	170.5	163.1	163.1
	171.5	164.1	164.1
	172.5	165.1	165.1
	173.5	166.1	166.1
	174.5	167.1	167.1
	175.5	168.1	168.1
	176.5	169.1	169.1
	177.5	170.1	170.1
	178.5	171.1	171.1
	179.5	172.1	172.1
	180.5	173.1	173.1
	181.5	174.1	174.1
	182.5	175.1	175.1
	183.5	176.1	176.1
	184.5	177.1	177.1
	185.5	178.1	178.1
	186.5	179.1	179.1
	187.5	180.1	180.1
	188.5	181.1	181.1
	189.5	182.1	182.1
	190.5	183.1	183.1
	191.5	184.1	184.1
	192.5	185.1	185.1
	193.5	186.1	186.1
	194.5	187.1	187.1
	195.5	188.1	188.1
	196.5	189.1	189.1
	197.5	190.1	190.1
	198.5	191.1	191.1
	199.5	192.1	192.1
	200.5	193.1	193.1
	201.5	194.1	194.1
	202.5	195.1	195.1
	203.5	196.1	196.1
	204.5	197.1	197.1
	205.5	198.1	198.1
	206.5	199.1	199.1
	207.5	200.1	200.1
	208.5	201.1	

Wie funktioniert Logging?

Wie funktioniert Logging?



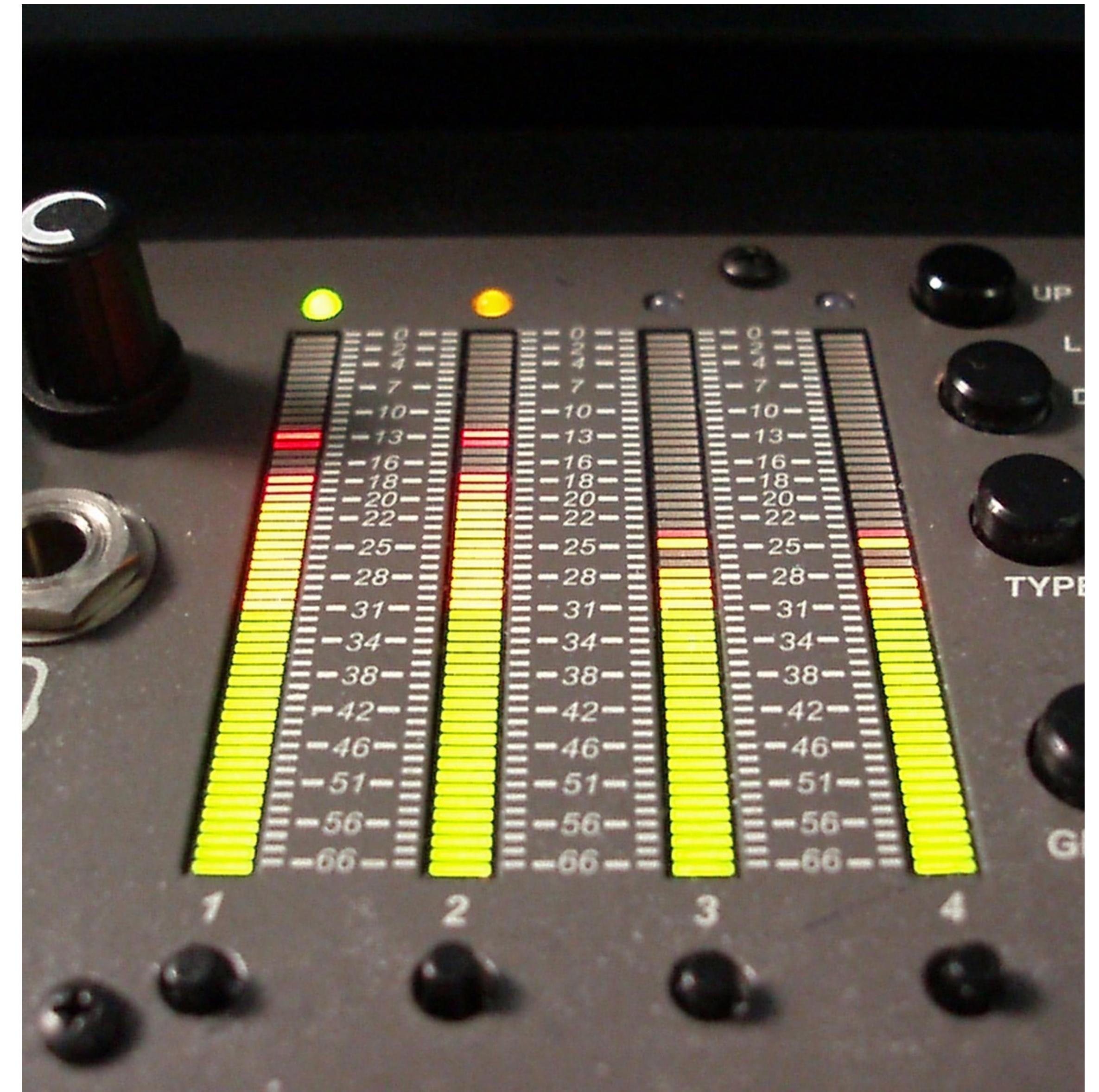
Wie funktioniert Logging?

- Logger erstellt einen LogRecord
- Appender formatiert LogRecord über Layout
- Appender gibt formatierte Meldung aus

Was sind Loglevel?

Was sind Loglevel?

- FATAL
- ERROR
- WARN
- INFO
- DEBUG
- TRACE

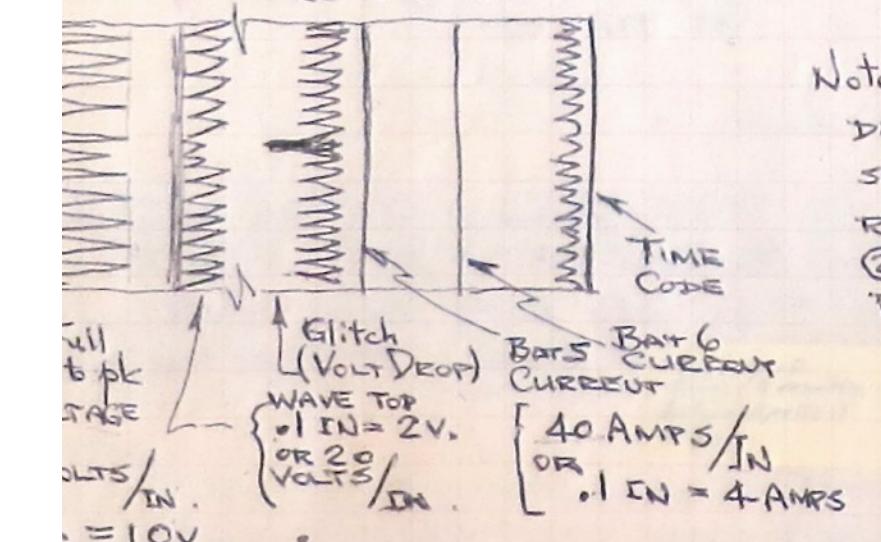


[levels \(Rory McSwiggan\) CC BY-NC](#)

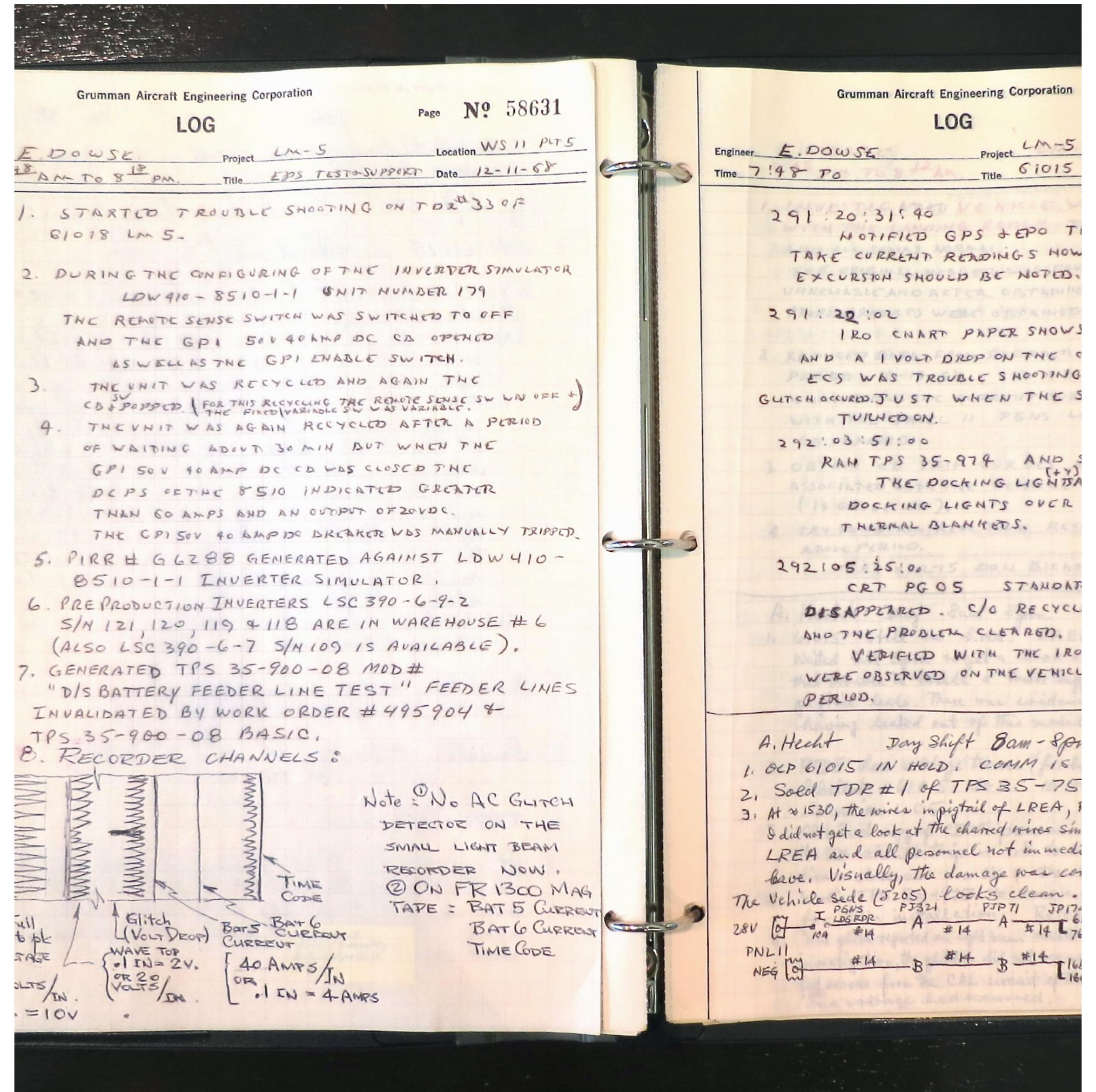
Was beinhaltet der LogRecord
noch?

Inhalt des LogRecord

- Zeitstempel
 - Logger-Name (Package und Class)
 - Message
 - Loglevel
 - Class und Method des Aufrufes
 - Source



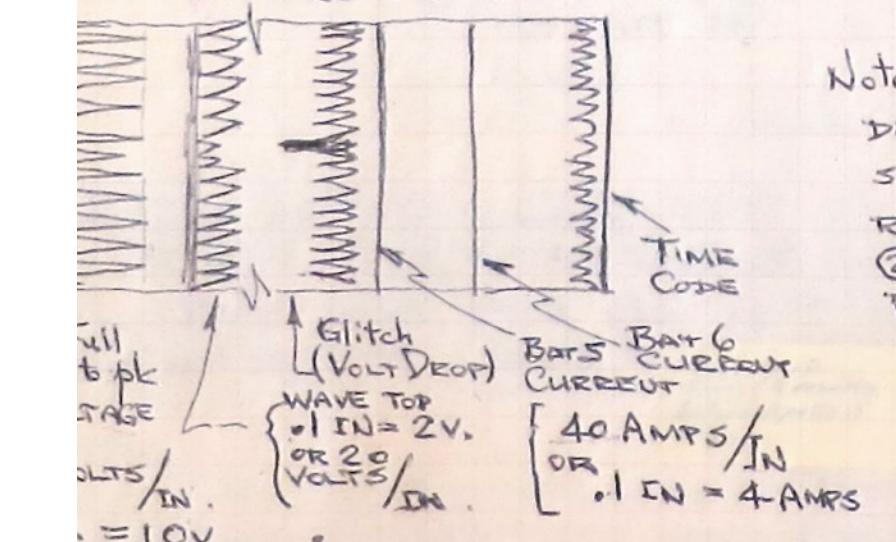
Note : ① No AC GLITCH
DETECTOR ON THE
SMALL LIGHT BEAM
RECORDER NOW .
② ON FR 1300 MAG
TAPE : BAT 5 CURRENT
BAT 6 CURRENT
TIME CODE



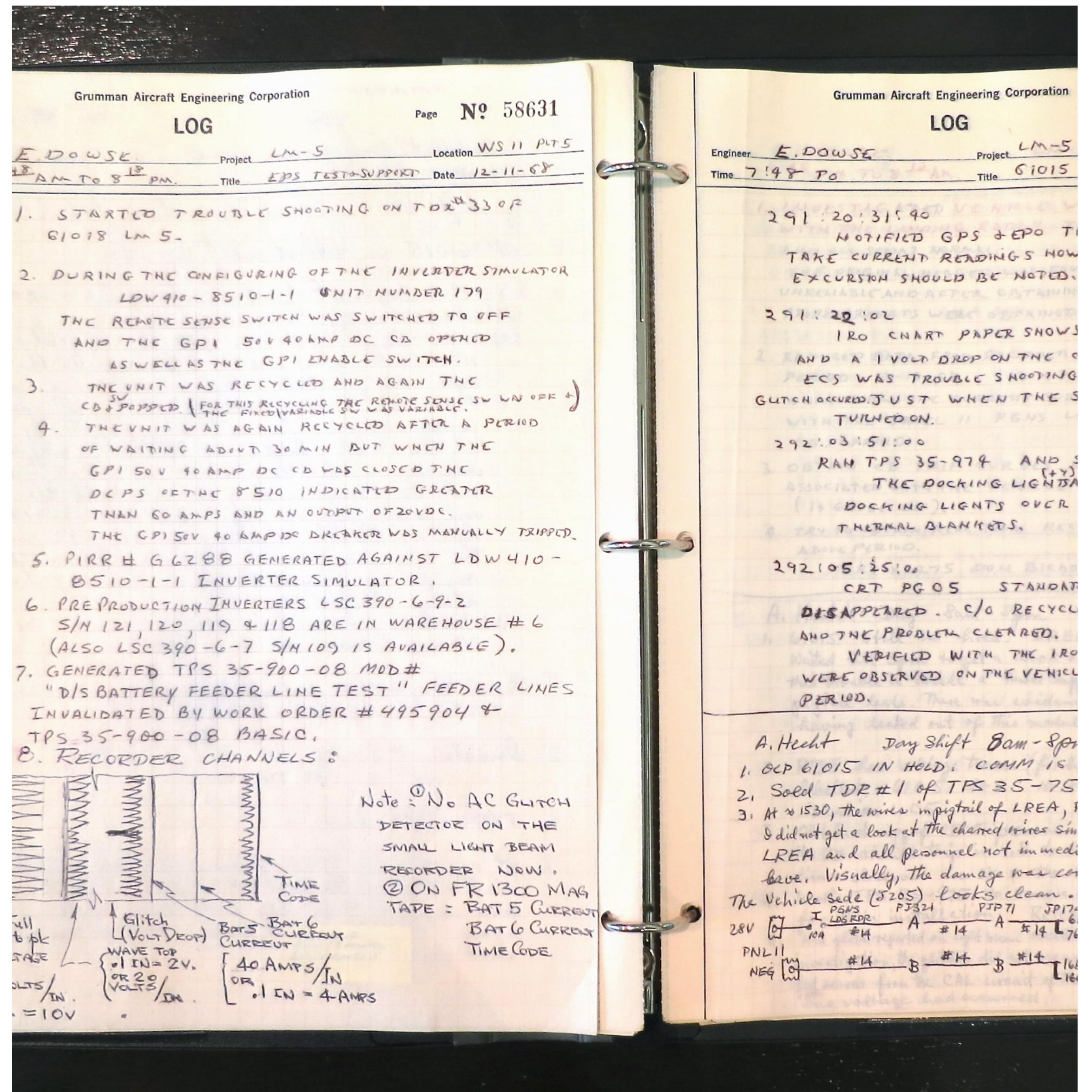
How the Eagle Landed – the Grumman Construction Log (Steve Jurvetson)

Inhalt des LogRecord

- Thread
 - NDC (nested diagnostic context)
 - MDC (mapped diagnostic context)
 - ThreadContext



Note : ① No AC GLITCH
DETECTOR ON THE
SMALL LIGHT BEAM
RECORDER NOW .
② ON FR 1300 MAG
TAPE : BAT 5 CURRENT
BAT 6 CURRENT
TIME CODE



How the Eagle Landed – the Grumman Construction Log (Steve Jurvetson)

Logger Name

- kann frei bestimmt werden
- Enthält das Package- und den Class-Name
- hierarchisch - wie Packages

Thread

- Bezeichnung des Thread / Thread-Pools
- vom Application-Server / Framework festgelegt

Context

- NDC bietet einen Stack
- MDC nutzt eine Map (key-value)
- ThreadContext nutzt eine Map
- zusätzliche, von der Anwendung gesetzte Informationen
 - UserId / UserName
 - Request
 - Tracing ID

Was sollte ich loggen?

Was wird ausgegeben?

fachlich

- (fachliche) Ereignisse der Applikation
- (unerwarteter) Zustand der Applikation
- Ergebnisse von Berechnungen
- Änderung an Daten

Was wird ausgegeben?

technisch

- Requests
- Datenbank-Zugriffe
- Start von Threads / Aktionen

Was wird ausgegeben?

from a higher level

- authentication, authorization and access
- changes
- availability (startup and shutdown)
- resources
- threats

Was ist zu beachten?

Was ist zu beachten?

- Ausgabe von Personen bezogenen Daten (DSGVO)
- für die Sicherheit relevante Informationen (Benutzernamen und Kennwörter)
- Informationen zur Infrastruktur (IP-Adressen)



[Security \(Patio Tours\) CC BY](#)

java.util.logging (JUL)

**Gibt's da auch etwas im
JDK?**



javaTM

[Java PNG Clipart CC BY-NC](#)

java.util.logging (JUL)



- Verfügbar seit JDK 1.4 (13.02.2002)
- keine externen Abhängigkeiten notwendig

Konfiguration



- logging.properties in \$JAVA_HOME/jre/lib bis Java 8
- logging.properties in \$JAVA_HOME/conf ab Java 9
- System-Property java.util.logging.config.file für Konfigurationsdatei



Nutzung

```
package de.ithempel.javalogging;

import java.util.logging.Level;
import java.util.logging.Logger;

public class HelloWorld {

    private static Logger logger = Logger.getLogger(HelloWorld.class.getName());

    public static void main(String[] args) {
        logger.info("Hello World!");
        logger.log(Level.WARNING, "Logging at WARNING level");
    }

}
```



Nutzung

```
package de.itempel.javalogging;

import java.util.logging.Level;
import java.util.logging.Logger;

public class HelloWorldConfig {

    static {
        String propertiesPath = HelloWorldConfig.class.getClassLoader()
            .getResource("logging.properties").getFile();
        System.setProperty("java.util.logging.config.file", propertiesPath);
    }

    private static Logger logger = Logger.getLogger(HelloWorldConfig.class.getName());

    public static void main(String[] args) {
        logger.info("Hello World!");
        logger.log(Level.WARNING, "Logging at WARNING level");
    }
}
```

andere Libraries



- JUL kennt keine anderen Libraries
- andere Libraries aber kennen JUL

log4j / log4j2

Der Urahn aller Logger

LOG4J



log4j



- erste Veröffentlichung vom 08.01.2001
- Entwickelt von Ceki Gülcü bei IBM (1997)
- Übergabe an die Apache Foundation im Jahr 2000
- Apache License 2.0



Ceki Gülcü PD

log4j

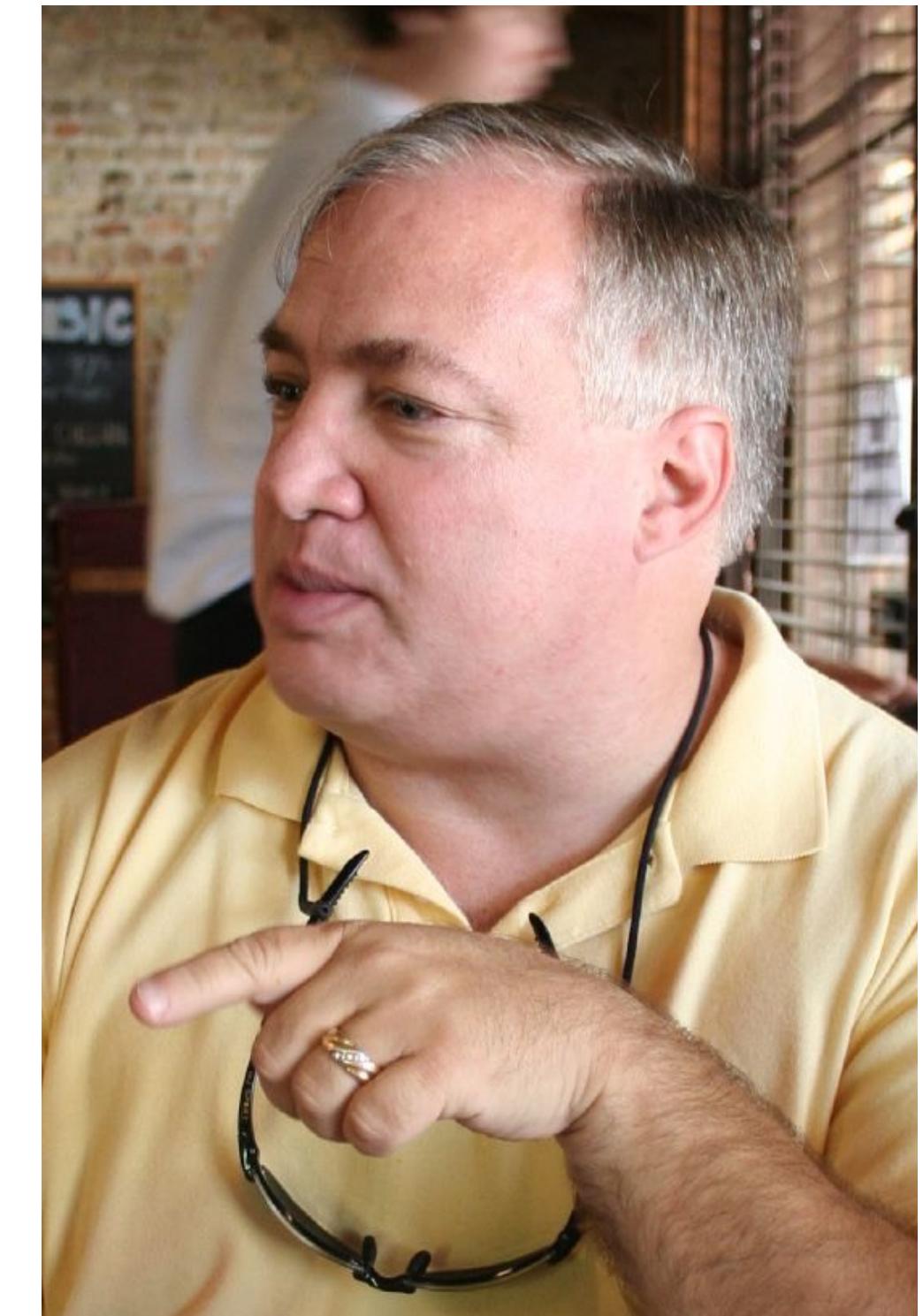


- Nach „feature complete“ keine weitere aktive Entwicklung
- Keine Trennung zwischen API und Implementierung
- Fork Reload4j für Security Fixes

log4j2



- Antwort auf die Entwicklung von Slf4j / Logback durch Gülcü
- komplette Neuimplementierung durch Ralph Goers (2009)
- erste Veröffentlichung im Juli 2014
- Apache License 2.0



Ralph Goers ([arjecahn](#)) CC BY-NC-ND

log4j2



- Trennt API und Implementierung
- nur Dependency auf API Artefakt
- Formatierung mittels {}
- Unterstützung von Lambdas

log4j



- asynchrone Logger
- Thread Context
- verliert keine Nachrichten beim Konfigurationswechsel wie Logback

Einbindung



```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.18.0</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.18.0</version>
</dependency>
```

Nutzung



```
package de.ithempel.log4j;

import org.apache.logging.log4j.Level;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class HelloWorld {

    private static Logger logger = LogManager.getLogger(HelloWorld.class.getName());

    public static void main(String[] args) {
        logger.info("Hello World!");
        logger.log(Level.WARN, "Logging at WARN level");

        logger.info("Logging with logger {}", logger.getName());
    }

}
```

Nutzung



```
package de.ithempel.log4j;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class HelloWorldLambda {

    private static Logger logger = LogManager.getLogger();

    public static void main(String[] args) {
        logger.debug("Logging with logger {}", () -> logger.getName());
    }

}
```

Konfiguration



- Dateien auf dem classpath
- Unterschiedliche Dateien für Test und Production
- Formate: Properties, yaml, json und XML
- System-Property `log4j2.configurationFile` zur Angabe des Pfads zur Konfigurationsdatei

Konfiguration



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

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

    <Loggers>
        <Logger name="de.ithempel.log4j" level="debug" additivity="false">
            <AppenderRef ref="Console"/>
        </Logger>
        <Root level="error">
            <AppenderRef ref="Console"/>
        </Root>
    </Loggers>
</Configuration>
```

andere Libraries

logback als Implementierung

```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-to-slf4j</artifactId>
  <version>2.18.0</version>
</dependency>
```





andere Libraries

log4j2 als Implementierung von Slf4j

```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-slf4j-impl</artifactId>
  <version>2.18.0</version>
</dependency>
```

andere Libraries

log4j2 als Implementierung von JUL

```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-jul</artifactId>
  <version>2.18.0</version>
</dependency>
```





andere Libraries

log4j2 als Backend von log4j

```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-1.2-api</artifactId>
  <version>2.18.0</version>
</dependency>
```

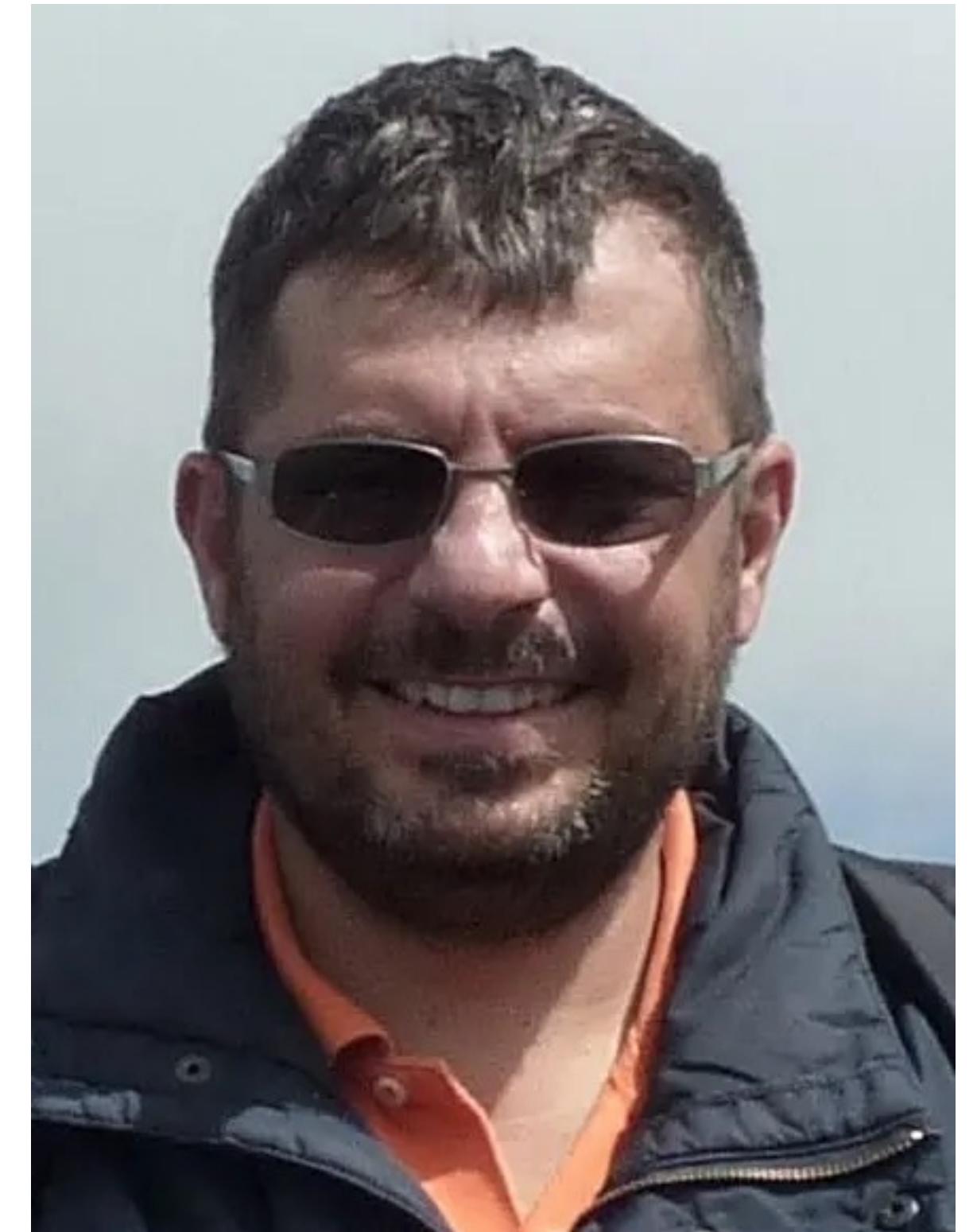
Slf4j / Logback

**A program has to be written
at least twice.**



[SLF4J Logo MIT License](#)

- Neuimplementierung von log4j von Ceki Gülcü
- Fassade vor den eigentlichen Loggern (log4j, logback)
- Erste Veröffentlichung 08.05.2006
- MIT License



Ceki Gülcü PD

- Nutzung des Service Loader Mechanismus von Java
- Fluent API seit Version 2.0
- Formatierung mittels {}
- Unterstützung von Lambdas
- Mapped Diagnostic Context

Logback



- Logback als native Implementierung von Slf4j
- erste Veröffentlichung 09.02.2006
- LGPL

Einbindung



```
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.3.0-alpha13</version>
</dependency>
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-core</artifactId>
    <version>1.3.0-alpha13</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>2.0.1</version>
</dependency>
```



Nutzung

```
package de.ithempel.slf4j;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class HelloWorld {

    private static Logger logger = LoggerFactory.getLogger(HelloWorld.class);

    public static void main(String[] args) {
        logger.info("Hello World!");
        logger.info("Logging with logger {}", logger.getName());
    }

}
```



Nutzung

```
package de.ithempel.slf4j;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class HelloWorldFluent {

    private static Logger logger = LoggerFactory.getLogger(HelloWorldFluent.class);

    public static void main(String[] args) {
        logger.atInfo()
            .setMessage("Logging with logger {}").addArgument(() -> logger.getName())
            .log();
    }
}
```

Logback Konfiguration



- XML-Konfiguration im classpath (Logback)
- Unterschiedliche Dateien für Test und Production
- System-Property `logback.configurationFile` zur Angabe des Pfads zur Konfigurationsdatei
- Wird nach Änderungen automatisch neu geladen

Konfiguration



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration>

<configuration scan="true">
    <import class="ch.qos.logback.classic.encoder.PatternLayoutEncoder"/>
    <import class="ch.qos.logback.core.ConsoleAppender"/>

    <appender name="STDOUT" class="ConsoleAppender">
        <encoder class="PatternLayoutEncoder">
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>

    <logger name="de.ithempel.slf4j" level="DEBUG" />

    <root level="INFO">
        <appender-ref ref="STDOUT"/>
    </root>
</configuration>
```



andere Libraries

SLf4j als Implementierung von JUL

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>jul-to-slf4j</artifactId>
  <version>2.0.1</version>
</dependency>
```



andere Libraries

SLf4j als Implementierung von log4j

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>log4j-over-slf4j</artifactId>
  <version>2.0.1</version>
</dependency>
```

JBoss Logmanager / Logging

**„Wir haben dann einmal
selbst ein Logging-
Framework implementiert.“**



JBoss Logging



- Logging Fassade
- Unterstützt JBoss Logmanager, Log4j, Slf4j und Logback
- erste Veröffentlichung 16.02.2011
- Apache 2.0 License

Jboss Logmanager



- Genutzt im JBoss Application Server und Quarkus
- Replacement für JUL
- erste Veröffentlichung 15.10.2010
- Apache License 2.0

Jboss Logmanager



- Außerhalb der „JBoss Welt“ keine Verbreitung
- Logging Adapter um die APIs von anderen Libraries zu unterstützen
- JUL, JBoss Logging, Slf4j, Apache Commons Logging, log4j, log4j2

Einbindung



```
<dependency>
    <groupId>org.jboss.logmanager</groupId>
    <artifactId>jboss-logmanager</artifactId>
    <version>2.1.18.Final</version>
</dependency>
<dependency>
    <groupId>org.jboss.logging</groupId>
    <artifactId>jboss-logging-spi</artifactId>
    <version>2.1.2.GA</version>
</dependency>
<dependency>
    <groupId>org.jboss.logging</groupId>
    <artifactId>jboss-logging-logmanager</artifactId>
    <version>2.2.0.CR2</version>
</dependency>
```



Nutzung

```
package de.ithempel.jbosslogging;

import org.jboss.logging.Logger;

public class HelloWorld {

    private static Logger logger = Logger.getLogger(HelloWorld.class);

    public static void main(String[] args) {
        logger.info("Hello World!");
        logger.debugf("Logging with logger %s", logger.getName());
    }

}
```

JBoss Logmanager Konfiguration



- JUL durch JBoss Logmanager ersetzen
- -Djava.util.logging.manager=org.jboss.logmanager.LogManager
- JBoss Logger wird wie JUL konfiguriert

andere Libraries

Adapter um andere Logausgaben umzuleiten

```
<dependency>
    <groupId>org.jboss.logmanager</groupId>
    <artifactId>log4j2-jboss-logmanager</artifactId>
    <version>1.1.1.Final</version>
</dependency>
<dependency>
    <groupId>org.jboss.slf4j</groupId>
    <artifactId>slf4j-jboss-logmanager</artifactId>
    <version>1.2.0.Final</version>
</dependency>
```

Apache Commons Logging

**Da gibt es natürlich auch
etwas von Apache**

**commons
logging**

Apache Commons Logging

commons
logging

- Fassade vor unterschiedlichen Implementierungen
- erste Veröffentlichung 13.08.2002
- Apache License 2.0
- relativ inaktiv (letzte Version von 2014)

Einbindung

commons
logging

```
<dependency>
  <groupId>commons-logging</groupId>
  <artifactId>commons-logging</artifactId>
  <version>1.2</version>
</dependency>
```

Nutzung

commons
logging

```
package de.ithempel.apachecommons;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

public class HelloWorld {

    private static Log logger = LogFactory.getLog(HelloWorld.class);

    public static void main(String[] args) {
        logger.info("Hello World!");
    }

}
```

Konfiguration

commons
logging

- SystemProperties
- Properties Datei in classpath (commons-logging.properties)

Tinylog

New kid on the block



Tinylog

- leichtgewichtiger Logger
- entwickelt von Martin Winandy
- Version 0.1 vom 23.01.2012
- Version 1.0 vom 01.04.2015
- Apache License 2.0



Martin Winandy PD

Tinylog

- Entstand aus Problemen mit log4j
- Logger müssen nicht erst erstellt werden
- Formatierung durch {}
- Hohe Performance bei Logausgaben
- Konfiguration über Properties
- Unterstützung von Lambdas
- Liegt inzwischen in der Version 2 vor

Einbindung

```
<dependency>
    <groupId>org.tinylog</groupId>
    <artifactId>tinylog-api</artifactId>
    <version>2.5.0</version>
</dependency>
<dependency>
    <groupId>org.tinylog</groupId>
    <artifactId>tinylog-impl</artifactId>
    <version>2.5.0</version>
</dependency>
```

Nutzung

```
package de.ithempel.tinylog;

import org.tinylog.Logger;

public class HelloWorld {

    public static void main(String[] args) {
        Logger.info("Hello World!");
        Logger.debug("Logging from class {}", HelloWorld.class.getName());
    }

}
```

Nutzung

```
package de.ithempel.tinylog;

import org.tinylog.Logger;

public class HelloWorldLambda {

    public static void main(String[] args) {
        Logger.debug("Logging from class {}", () -> HelloWorldLambda.class.getName());
    }

}
```

tinylog Konfiguration

- SystemProperties
- Properties Datei
- Separate Dateien zur Konfiguration von PROD und TEST

Konfiguration

```
writer1      = console  
writer1.level = debug
```

```
writer2      = file  
writer2.file = log.txt  
writer2.level = info  
writer2.append = true
```

andere Libraries

APIs für tinylog Backend

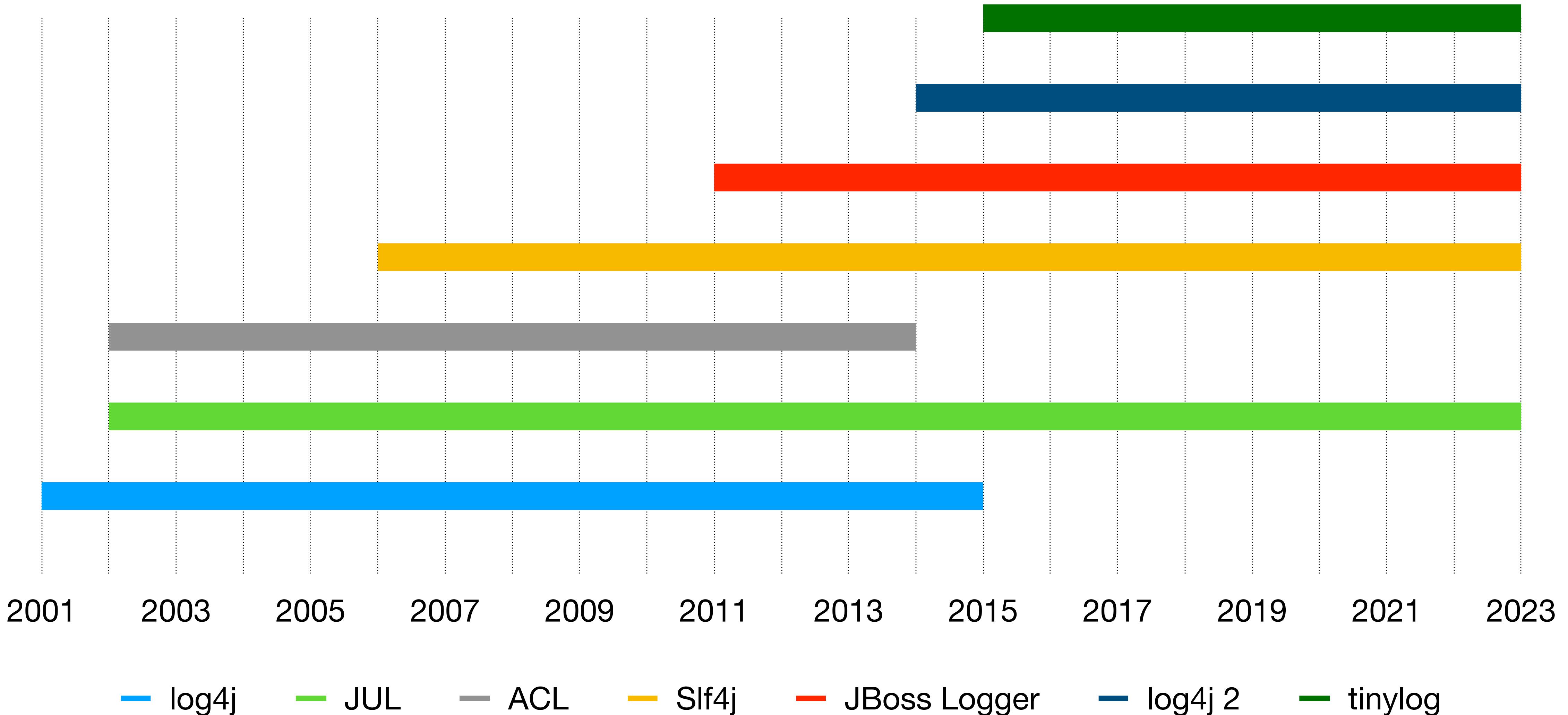
```
<dependency>
  <groupId>org.tinylog</groupId>
  <artifactId>log4j1.2-api</artifactId>
  <version>2.5.0</version>
</dependency>
<dependency>
  <groupId>org.tinylog</groupId>
  <artifactId>slf4j-tinylog</artifactId>
  <version>2.5.0</version>
</dependency>
<dependency>
  <groupId>org.tinylog</groupId>
  <artifactId>jboss-tinylog</artifactId>
  <version>2.5.0</version>
</dependency>
```

Fazit

Was man sich merken sollte



zeitlicher Überblick



Nur ein Backend verwenden

Nur ein Backend verwenden

- So gut wie alle Logausgaben lassen sich auf ein „Backend“ umbiegen.
- Damit ist nur noch eine Konfigurationsdatei notwendig.

In Bibliotheken nur API einbinden

In Bibliotheken nur API einbinden

- Es ist egal, welchen Logger eine Bibliothek einsetze.
- Diese sollte aber nur die API einbinden.

Fassaden werden nicht benötigt

Fassaden werden nicht benötigt

- Fassaden bringen keinen Mehrwert
 - Slf4j
 - Apache Commons Logging
 - JBoss Logging