Verteilte Systeme und Komponenten FS23
Version 1.0.0 / 09. März 2023

Logger-Schnittstelle

Spezifikation

Dominik Strässle, Patrick Henseler, Silvan Büchi

Inhalt

Version	Datum	Autor	Bemerkung	Status
1.0.0	09.03.2023			

1. Steckbrief

Name	**LoggerSetup**
Beschreibung	Definiert eine Schnittstelle für ein Builder-Pattern, durch welches eine Instanz des Logger-Interface konfiguriert und erzeugt werden kann.
Typ	Java Interface

Tabelle 1: Steckbrief der Schnittstelle LoggerSetup.

Name	**Logger**
Beschreibung	Definiert eine Schnittstelle, welche es ermöglicht Log-Nachrichten auf einem bestimmten Log-Level zu erfassen.
Typ	Java Interface

Tabelle 1: Steckbrief der Schnittstelle Logger.

2. Operationen und Datenstukturen

Die Schnittstelle stellt folgende Methoden und Datentypen zur Verfügung:

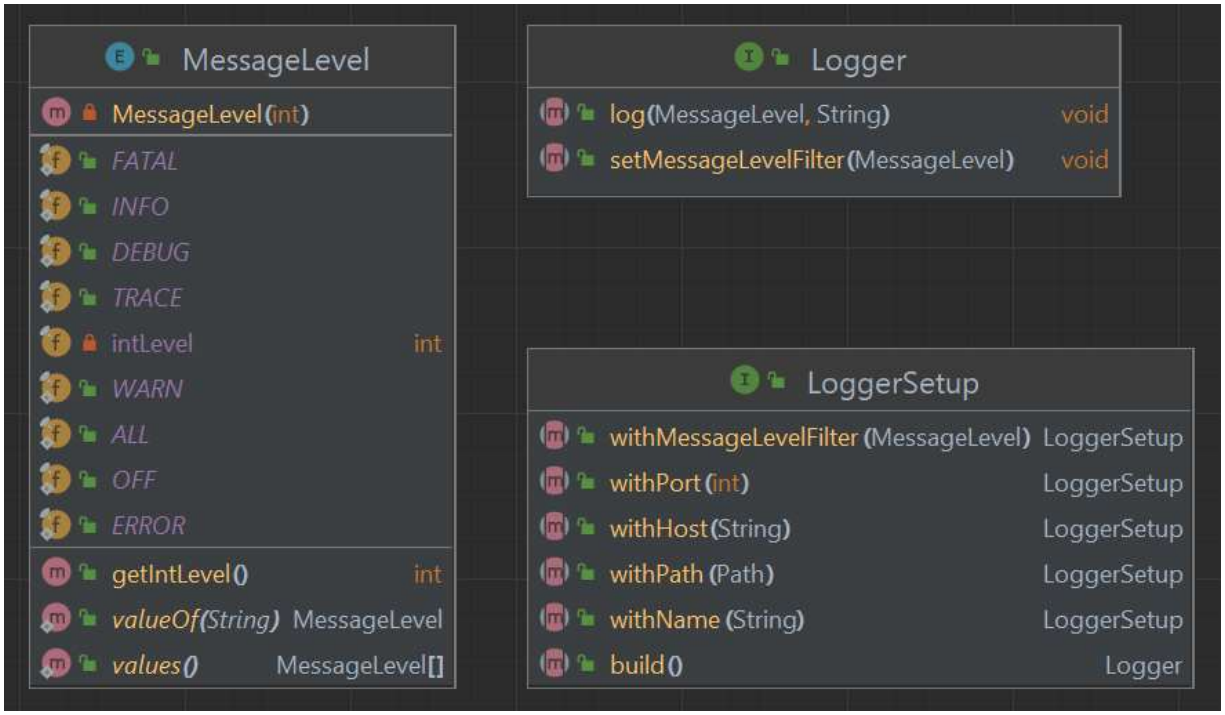

Abbildung 1: Klassendiagramm der Schnittstelle Logger, LoggerSetup und dem Enum MessageLevel.

Details zu den Methoden und den verwendeten Datentypen sind in der JavaDoc festgehalten. Diese wird Ihnen auf dem ILIAS zur Verfügung gestellt. Die Distribution erfolgt über das Maven Repository https://repohub.enterpriselab.ch und steht unter den folgenden Koordinaten zur Verfügung:

Group:	**ch.hslu.vsk23fs**	Name:	**loggerinterface**

3. Einsatz, Abläufe, Voraussetzungen und Zusicherungen

- Einer Implementation von LoggerSetup steht es offen, Standardwerte für die Parameter zu definieren.
- Der aktuelle MessageLevelFilter kann initial via LoggerSetup oder nachträglich via Logger definiert werden. Alle Logs, welche einen Wert höher als das aktuelle MessageFilterLevel besitzen, werden nicht an den

Server übertragen. Die Wertung ist von Log4J übernommen und kann folgender Tabelle entnommen werden:

MessageLevel	**Wertung**
OFF	0
FATAL	100
ERROR	200
WARN	300
INFO	400
DEBUG	500
TRACE	600
ALL	Integer.MAX_VALUE

Tabelle 3: Wertung der MessageLevels

Parameter	**Beschreibung**	**Gültige Beispiele**	**Ungültige Beispiele**
Host	Hostname des Logger Servers. Vorgaben sind definiert durch den Host-Parameter des URI Klasse von Java. Wert kann auch eine IPv4 oder IPv6 Adresse sein.	192.168.1.2 hslu.ch	Hslu ch null
Port	Port des Logger Servers. Muss eine positive Ganzzahl zwischen 0 – 65535 sein.	8080 8181	-100 6553599
File	Eine Instanz von File. Diese darf nicht «null» sein und der aktuelle Prozess muss Schreib- und Lese-Rechte auf diesem File besitzen.		null
MessageLevel Filter	Eine Konstante aus dem **MessageLevel** Enum.	MessageLevel.ALL	Invalid null
Name	Name des Clients. Darf beliebig gewählt werden, aber darf nicht «null» sein.	Fjl;adsjfldasfl valid anything «»	null

Tabelle 4: Beschreibung der Parameter von LoggerSetup

- Die Methode **log()** vom Logger soll in der benutzenden Applikation keine Performance-Einbussen verursachen. Demnach ist das Senden der Logs an den Server asynchron zu implementieren.

- Falls der Logger beim Stop der Applikation noch Aufräumen und die Verbindung stoppen muss, so muss er die selbständig via Runtime::ShutdownHook machen. Die benutzende Applikation darf ausser von einer möglichen Verzögerung von maximal 10 Sekunden beim Stoppen der Applikation nicht beeinflusst werden.

4. Aufbau und Konfiguration

Keine zusätzlichen Informationen.

5. Fehlerbehandlung

- Werden dem LoggerSetup ungültige Parameter mitgegeben oder benötigte Eingaben vergessen, so kann bei Aufruf der build() Methode eine unchecked Exception geworfen werden. Die Vorgaben der Parameter können der Tabelle 4 entnommen werden.

- Wenn auf dem angegebenen Port des angegebenen Hosts kein Logger Server verfügbar ist, so soll eine unchecked exceptions geworfen werden bei Aufruf der build() Methode.

6. Qualitätsmerkmale

- Mit **log()** kann ein String gespeichert werden mit bis zu 1000 Zeichen.
- **log()** soll beliebig oft aufrufbar sein und asynchron abgearbeitet werden.

7. Entwurfsentscheidungen

- Die Schnittstelle wurde bewusst sehr schmal gehalten, um den Implementationsaufwand in Grenzen zu halten. Deshalb wurde auch auf zusätzliche Auslagerungen in weitere Schnittstellen verzichtet.
- Der komplexe Datentyp **Path** wird verwendet um eine Vielzahl von Möglichkeiten für die Angabe einer Datei zur ermöglichen. Des Weiteren wird **Path** bereits für den **StringPersistor (Version 7.0.0)** verwendet.
- Diese Schnittstelle setzt Java Version **17** (LTS) voraus. Eine Verwendung mit älteren Versionen ist nicht möglich.
- Auf die Implementation einer Methode zur Trennung der Verbindung wurde verzichtet. Dies wird mit JVM Shutdown Hooks umgesetzt.

8. Beispielverwendung

Der folgende Codeausschnitt zeigt die Verwendung der Schnittstelle anhand einer beispielhaften Implementation **G07LoggerSetup** und **G07Logger**:

```
// instanziierung eines Loggers mit allen möglichen Parametern
LoggerSetup loggerSetup = new G04LoggerSetup()
      .withPath(Paths.get("test.txt"))
      .withMessageLevelFilter(MessageLevel.ALL)
      .withHost("localhost")
      .withPort(8181)
      .withName("123");

Logger logger = loggerSetup.build();
```

```
// loggen einer Nachricht
logger.log(MessageLevel.DEBUG, "my message");
```

```
// Ändern des MessageLevel Filters nach der Initialisierung
logger.setMessageLevelFilter(MessageLevel.TRACE);
```

9. Änderungsmanagement

Änderungswünsche an dieser Schnittstellendefinition oder an der API-Implementation können mit entsprechender Begründung per Mail oder als Forumsbeitrag an das Dozierendenteam eingereicht werden. Von diesem wird dann entschieden (ggf. im Plenum) ob die Änderungswünsche erfüllt werden.