Nachdenkzettel Logging

Kristin Behringer, 40898; Felix Schindler, 40892; Bastian Bodenhöfer, 39634

Vorname, Name, Matrikelnummer

1. Kennzeichnen Sie in der Config die Stellen wo über das

- was geloggt wird

- wieviel geloggt wird

- wo geloggt wird

- wie geloggt wird

entschieden wird

<Configuration>

<Appenders>

<File name="A1" fileName="A1.log" append="false">

<PatternLayout pattern="%t %-5p %c{2} - %m%n"/>

</File>

<Console name="STDOUT" target="SYSTEM\_OUT">

<PatternLayout pattern="%d %-5p [%t] %C{2} (%F:%L) - %m%n"/>

</Console>

</Appenders>

<Loggers>

<!-- You may want to define class or package level per-logger rules -->

<Logger name="se2examples.core.businessLogic.VehicleManager" level="debug">

<AppenderRef ref="A1"/>

</Logger>

<Root level="debug">

<AppenderRef ref="STDOUT"/>

</Root>

</Loggers>

</Configuration>

2. Geben Sie je ein Beispiel wann Sie den loglevel

- error 🡪 bei einem fatalen Fehler von besonderer Wichtigkeit: z.B. Datenbank-Verbindung fehlgeschlagen

- info 🡪 um z.B. während der Runtime das Funktionieren bestimmter Anwendungen zu überprüfen (dass z.B. die Applikation gestartet ist)

- debug (🡪 um z.B. während der Runtime das Funktionieren bestimmte Prozesse wie z.B. Anwendungen zu dokumentieren, wann was geschieht (dass z.B. dass package X aufgerufen wird etc.)

verwenden

3. Sie verwenden einen FileAppender für das Logging. Jetzt

soll Ihre Application im Datacenter laufen. Was machen Sie mit dem FileAppender?

Wir konfigurieren die Appender Klasse auf einen JDBCAppender.

4. Macht logging Ihre Application langsamer? Was passiert wenn Sie log.debug(„foobar“); aufrufen? Wie sollte sich das Logging Subsystem verhalten?

Ja, um die paar Millisekunden, die Java intern braucht, um etwas auszugeben.

In der Log-Datei kommt als Debug meldung „foobar“

Das Logging Subsystem überprüft die Konfiguration nach einer festgelegten Logging-Datei, falls es eine gibt, wird die Debug Meldung in die Datei geschrieben.

5. Ein Request an Ihre Application durchläuft einen Proxy Server, dann einen Web Server, dann einen Application Server und dann die Datenbank. Auf jedem Server loggen Sie die Requests. Welches Problem tritt auf?

- Wenn das alles auf einem Server läuft, wäre es unnötig die gleiche Request 4x zu loggen.

- Es ist schwierig etwas Bestimmtes zu finden, wenn man zuerst alle Logs aus den verschiedenen Anwendungen / Servern zusammentragen muss.

- Durch die vielen Logger wird es unübersichtlich.

6. Was sollten Sie pro Komponente/Tier loggen?

* Userverhalten: Fehlerhafter Input, allg. Verhalten
* Exceptions
  + z.B. Fehlerhafter Input, (Runtime-) Exceptions etc.
* Anwendungsereignisse
  + z.B. Start der Anwendung/Komponente/GUI-Fenster
* Anwendungszustand
  + z.B. Freigabe von Verbindungen (oder die Nicht-Freigabe)
* Zeit/Performance
  + Wie lange hat etwas gedauert? Wie lange musste man warten auf eine Resource?
* Debugginginformationen
  + z.B. Loggen von Variablenwerten bei schwierigen Algorithmen
* Abläufe: Wo ist meine Applikation durchgelaufen?
  + Im Methodenkopf eine logging Ausgabe der Parameter
* DB-Abfragen
  + z.B. Loggen der SQL-Statments,
* Große, teure, wichtige Dinge wie z.B: Threads
  + z.B. Start und Ende eines Threads
* Anzahl von Dingen
  + Requests, Verbindungen, Elemente in einem Pool, User,
* Zeitstempel für die Dauer von Abläufen
  + Anfragen, Verbindungsaufbau, Session, Berechnungen

7. Aus Geschwindigkeitsgründen halten Sie teure DB-Connections auf Vorrat in einem Pool. Jeder Request vom Client braucht dann eine Connection. Der Pool hat die Methoden:

DB Connection con = ConnectionPool.getConnection();

ConnectionPool.freeConnection( DBConnection dbCon);

Was loggen Sie in Ihrem App Server? Oder anders gefragt: Was wollen Sie beim Umgang mit dem Pool als Software-Architekt:in wissen?

Alles mögliche.

* Wer wann welche Verbindung nutzt/braucht oder missbraucht.
* Für was die Verbindung genutzt wird.