

## Nachdenkzettel Logging

.....  
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>
```

1.2 Wie würde man erreichen, dass für alle Klassen innerhalb eines Packages ein spezieller Loglevel gelten würde? Könnte man auch alle Klassen eines Packages in ein anderes File loggen?

Man muss die Konfiguration des Logging Frameworks anpassen

2. Geben Sie je ein Beispiel wann Sie den loglevel

- error: Exceptions, Fehlverhalten des Users/ der Software
- info: Start / Stopp der Anwendung
- debug: Fehlerbehebung, Leistungsanalyse

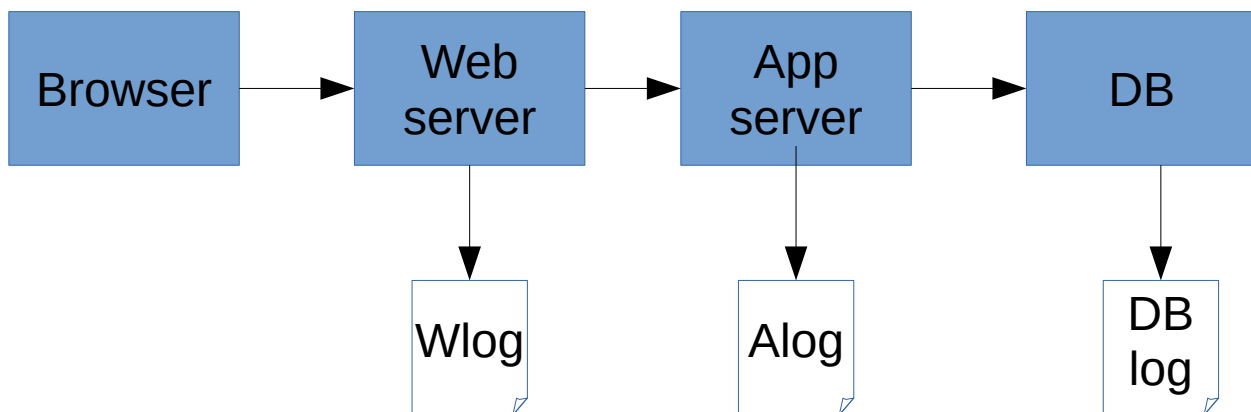
3. Sie verwenden einen FileAppender für das Logging. Jetzt soll Ihre Application im Datacenter laufen. Was machen Sie mit dem FileAppender?

Rolling Appender. Dieser loggt bis zu einer bestimmten Menge und beginnt dann von vorne zu überschreiben

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

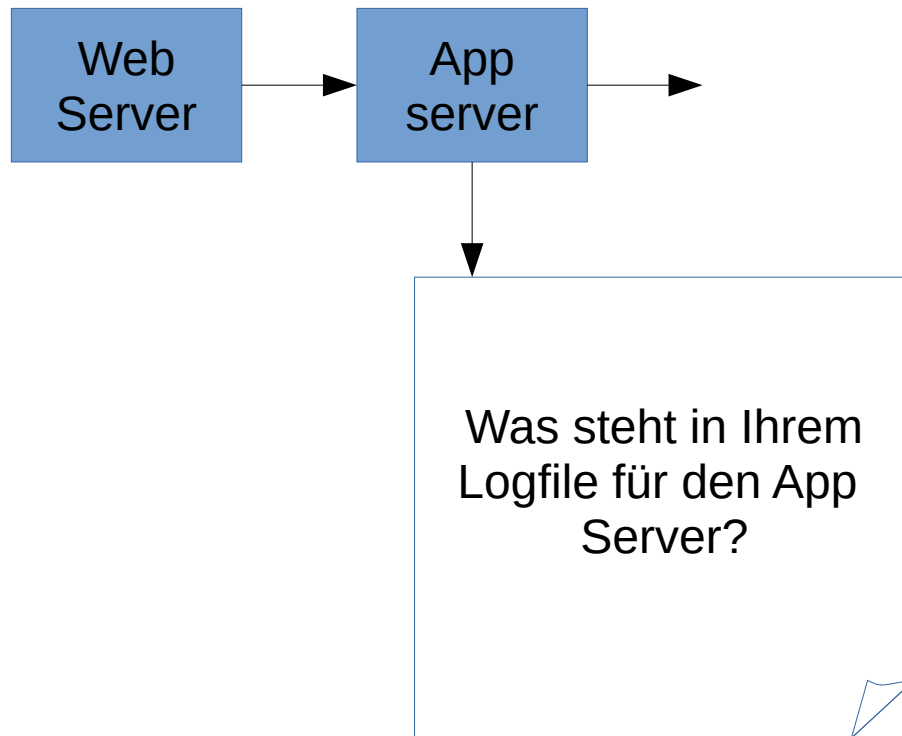
- In Java macht logging die software höchstens minimal langsamer
- `log.debug("foobar")`: Normalerweise wird das Logging-Subsystem prüfen, ob das aktuelle Log-Level (in diesem Fall "debug") für die Ausgabe von Protokolldaten aktiviert ist. Wenn das Level aktiviert ist, wird die Nachricht "foobar" an das Logging-Subsystem übergeben, das sie dann je nach Konfiguration weiterverarbeitet.

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?



Es werden unterschiedliche Dinge auf unterschiedlichen log-leveln geloggt, aufgrund der verschiedenen Konfigurationen. Wenn also ein Fehler auftritt, weiß man erst nicht genau woher dieser kommt und muss alle 3 logfile durchsuchen

6. Was sollten Sie pro Komponente/Tier loggen?



- Webserver logfile: Logging der HTTP Statuscodes (hat Request funktioniert/ ist fehlgeschlagen?)
- App Server: Logging aller wichtigen Informationen zu der Applikation (Zeiten, Nutzerverhalten...)

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-Architektin wissen?

Man möchte wissen, welche freie Connection es gibt, welche zur Verbindung zur DB benutzt wurde, wie lange diese Connections dauern und ob es Abbrüche gab.

8. Sie fügen log-statements in die Login-Klasse ein. Was müssen Sie unbedingt beachten???

Tipp: Denken Sie über Userverhalten nach. Und über Mitarbeiter....

- Sensible Informationen schützen
- Anonymisierung
- Zugriffssteuerung und Berechtigung
- Log-Level angemessen einstellen