Nachdenkzettel Logging

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 my 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 verwenden:

error: Fehler-Protokullation egal welcher Art
info: Informationen zur Schnittstelle
debug: Analyse von Sachverhalten zumeist Fehler/Bugs

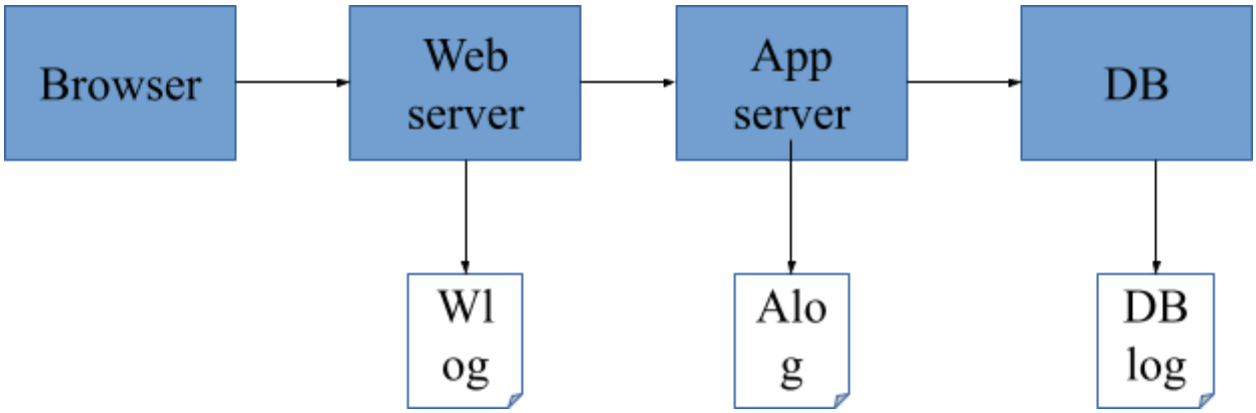
3. Sie verwenden einen FileAppender für das Logging. Jetzt soll Ihre Application im Datacenter laufen. Was machen Sie mit dem FileAppender?

In einer Cloud werden logfiles automatisch gesammelt, heisst, es fallen meist keine lokalen Files mehr an.

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

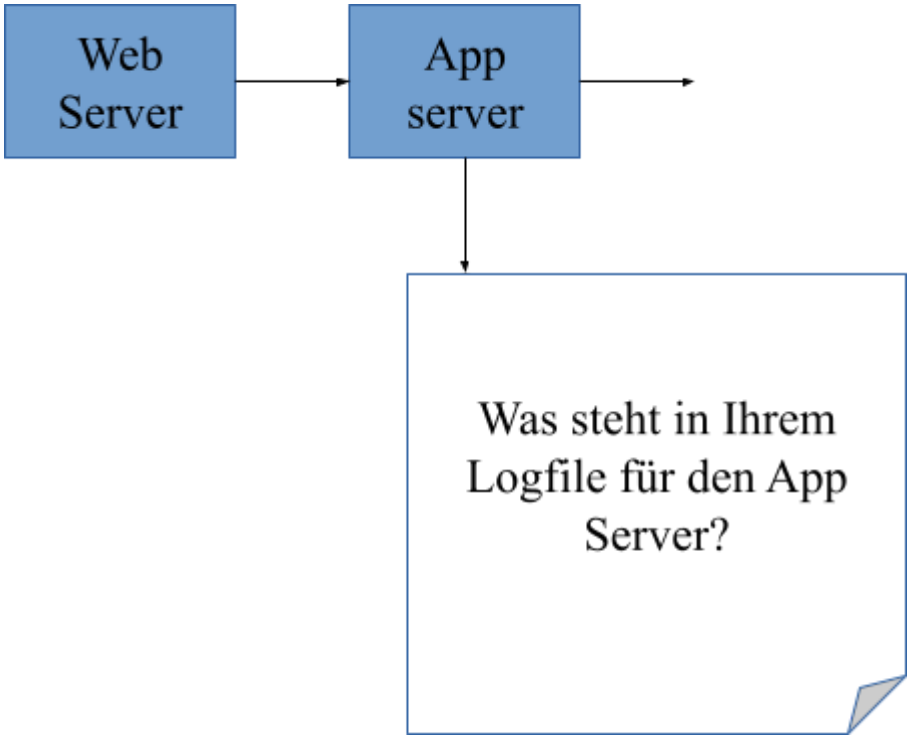
Ja, logging macht die Application langsamer. log.debug(“foobar”) loggt die Debug Daten.

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?



Die Performance der Application nimmt ab und der Zugriff verzögert sich. Denn es werden für eine Anfrage drei Logg-Requests erstellt und durchlaufen.
Ggf. kommt es zu Problemen im Zwischenspeicher.

6.



Fehler/Probleme bsp2. als Exceptions. Notwendige Informationen damit die Performance nicht verloren geht. RequestIDs; UserURLs, Timestamp, Rechnername, Anwendungsname, Anwendungsfunktion, SessionID und Protokollport.

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?

Ich möchte wissen was in dem Pool passiert, sprich;

- die Anzahl an Requests, wer fragt an?

- welche Verbindung passiert wann?
- wie viele Elemente sind überhaupt in dem Pool?
- wer nutzt den Pool?