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 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
- error
=> Wenn eine regex Prüfung fehlschlägt
- info
=> erfolgreich Ausführungen von Programmabschnitten
- debug
=> Methodenaufrufe mit Übergabe werten und results
verwenden

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

=> FileAppender durch SocketLogging oder DataBaseLogging ersetzen

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

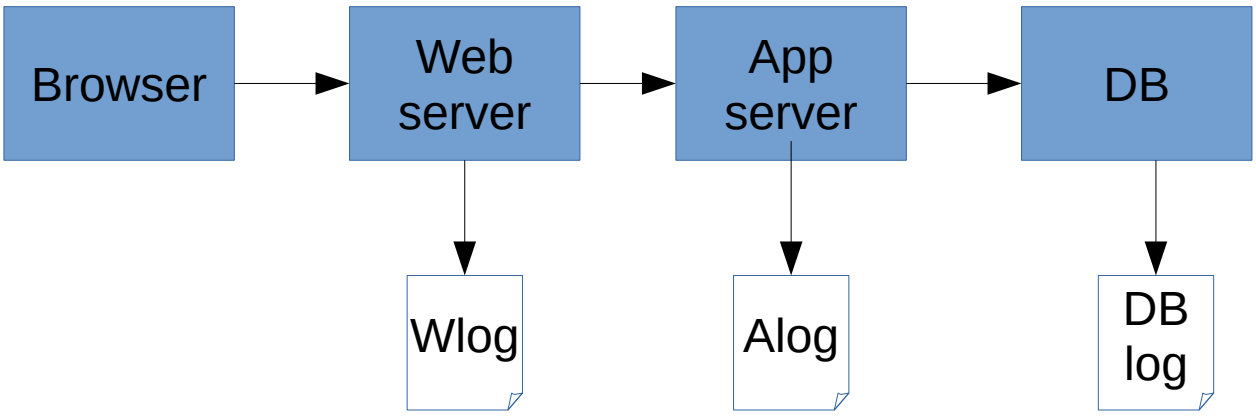
1: Im Falle von log4j: Von der Konfiguration abhängig
Asynchron: Kaum Einfluss
Synchron: minimaler Einfluss

2: Asynchron: Log Entry wird im Queue abgelegt und danach abgearbeitet
Synchron: Log Entry wird sofort geschrieben

3: So wenig Ressourcen wie möglich verschwenden (Hauptspeicher, Ausführungszeit, etc...)

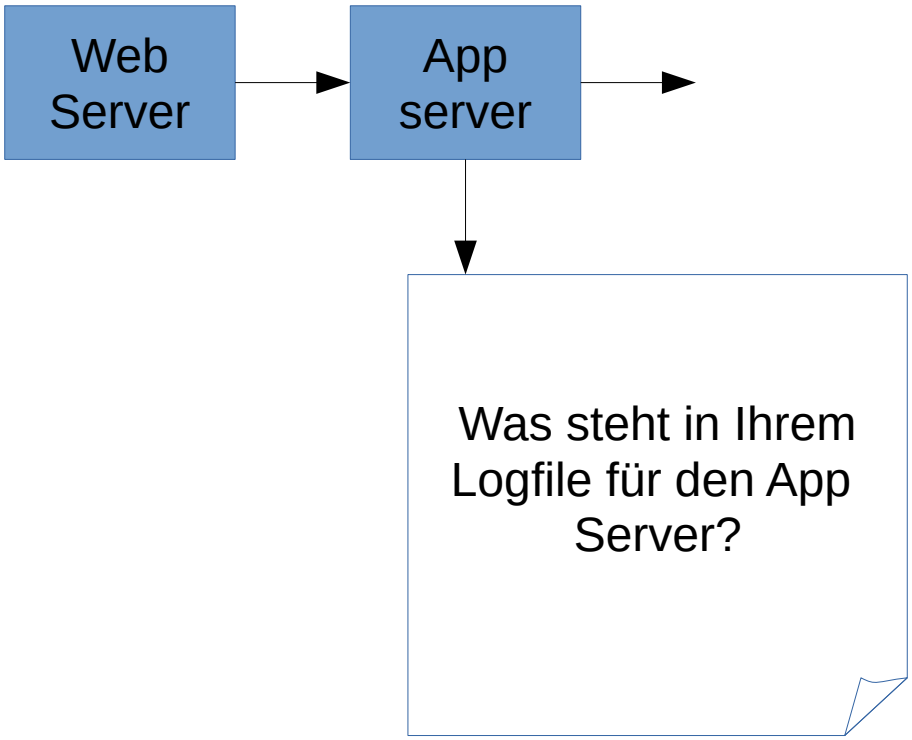
5. Ein Request an Ihre Application durchläuft einen Proxy Server, dann einen Web Server, dann einen Application Server und dann die Datenbank. Auf jedem Server loggen Sie die Requests. Welches Problem tritt auf?

=> Logs sind Dezentralisiert und Assoziation einzelner Entrys ist schwieriger



6. Was sollten Sie pro Komponente/Tier loggen?

=>
- Anfragen der vorherigen/an die nächste Komponente
- Programm relevanten Ereignisse des Webservers



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?

=>
- Wie viele Connections sind belegt/frei
- ggf. ob eine Connection abgebrochen und/oder (wieder)hergestellt wurde
- Performance
- Welche Befehle wurden der Connection mitgegeben