Nachdenkzettel Logging

….....
Vorname, Name, Matrikelnummer

1. Kennzeichnen Sie in der Config die Stellen wo über das
- was geloggt wird
- wieviel geloggtwird
- wo geloggtwird (ref in logger verweißt auf target)
- wie geloggtwird
entschiedenwird

```
<Configuration>
   <Appenders>
      <File name="A1" fileName="A1.log"  append="false">
         <PatternLayout pattern="%t %-5p %c{2}  - %m%n"/>
      </File>
      <File name="bLogic" fileName="businessLogic.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>
      <Logger name="se2examples.core.businessLogic" level="error">
         <AppenderRef ref="bLogic"/>
      </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?

Indem man unter den <Logger>...</Logger> einen Logger für das ganze Package definert, der alles loggt was in den einzelnen Klassen geloggt wird, jedoch natürlich nur das mit dem definierten Level. Als AppenderRef könnte man dann auch ein anderes File nehmen.
[Beispiel oben in italic eingetragen]

2. Geben Sie je ein Beispiel wann Sie den loglevel
- error: *Bei Exceptions die nicht abgefangen werden können oder die nicht „geheilt“ werden können*
- info: *Man will einen Überblick haben wo es "entlanggelaufen" ist*
- debug: *debug wichtige Informationen z.B. wo man normal sysout print genommen hätte* verwenden

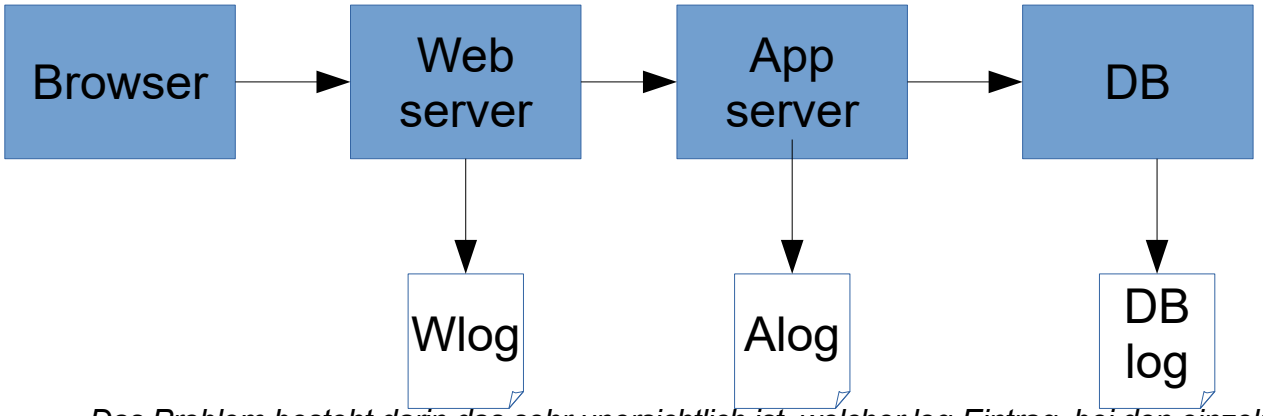
3. Sie verwenden einen FileAppender für das Logging. Jetzt soll Ihre Application im Datacenter laufen. Was machen Sie mit dem FileAppender?

Man sollte einen rolling FileAppender benutzen, der bei einer bestimmten Größe das log file von Anfang an überschreibt und einen skalierbaren Appender benutzen, der je nach Bedarf angepasst wird.

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

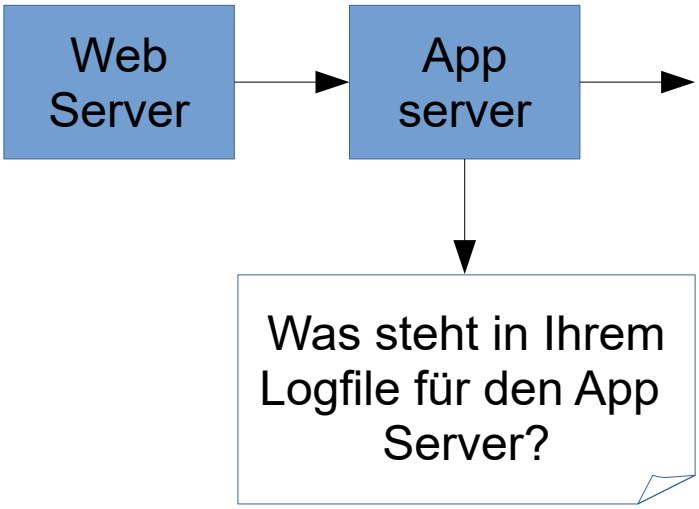
Im Allgemeinen sollte logging die Application nicht sichtlich verlangsamen, kann jedoch durch schlechte Programmierung durchaus vorkommen (z.B. multi-threaded for-schleifen, die jeden Durchgang loggen und so tausende Einträge in ein paar Sekunden erzeugen).
Wenn log.debug("foobar") aufgerufen wird, gibt der Logger "log" mit dem Log-Level debug die message "foobar" an den Appender, welcher in der log4j.xml definiert wurde weiter.
Das Subsystem sollte sich so verhalten, dass genau ersichtlich ist wo, wann und mit welchem Level geloggt wurde.

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?



Das Problem besteht darin das sehr unersichtlich ist, welcher log-Eintrag, bei den einzelnen log-files jetzt genau von meiner Request kam. Das Log system sollte durch Tokens, die übergeben werden, alle Logs zusammenführen können, um so einen guten Überblick zu erschaffen. Zeitpunkte wäre als Tokens schlecht, da bei Servern Synchronisation schwer umzusetzen ist und allgemein Zeit ein schlechter Normfaktor bei Webapplikation ist.

6. Was sollten Sie pro Komponente/Tier loggen?



Alles was mit der Verbindung zu tun hat sollte der Webserver loggen(IP, Protokolle, Status der Verbindung etc. außer PWs)

Alles was mit der Applikation zu tun hat loggt der AppServer(z.B. edit, delete, changed file etc.)

Zeiten sollten auch geloggt werden, z.B. wie lange man gebraucht hat um Daten von der DB abzufragen -> zu lange -> log.warn("...")

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?

Welche sind benuzt? Wer kriegt keine und muss daher warten? Was ist frei? Connectionabbruch?
Wenn/Wann GetConnection und wenn/wann FreeConnection gemacht werden.
Zeitstempel: Für jeden Entwickler im Team: wie lange hat connection gedauert etc.

8. Sie fügen log-statements in die Login-Klasse ein. Was müssen Sie unbedingt beachten???
Tipp: Denken Sie über Userverhalten nach. Und über Mitarbeiter….

*Es sollten **NIEMALS** Passwörter geloggt werden. Man könnte z.B. loggen, wer sich wie lange verbunden hat (Datenschutz technisch auch schwierig) oder ob man versucht hat sich mit falschen Passwort anzumelden (Fehlversuche z.B. wegen DDOS Attacken bei Webservern).*