

openolat LMS - JMeter Loadtests

IPA Dokumentation

Copyright (C) frentix GmbH.

MITWIRKENDE

	<i>TITEL :</i> openolat LMS - JMeter Loadtests		
<i>AKTION</i>	<i>NAME</i>	<i>DATUM</i>	<i>UNTERSCHRIFT</i>
VERFASST DURCH	Joël Krähemann	26. März 2013	

VERSIONSGESCHICHTE

NUMMER	DATUM	BESCHREIBUNG	NAME

Inhaltsverzeichnis

1	Einleitung	1
1.1	Ausgangslage	1
1.2	Auftrag	1
1.2.1	Aufgabenstellung	1
1.2.2	Kriterienkatalog	2
1.2.2.1	Muss-Kriterien	2
1.2.3	Zeitplan	4
1.3	Vorkenntnisse	4
1.4	Journal	5
2	Umgebung	10
2.1	Debian GNU/Linux	10
2.2	Apache Tomcat	11
2.3	MySQL	15
2.4	openolat LMS	15
2.5	JMeter	18
3	JMeter Loadtests	21
3.1	Threads (Users)	21
3.1.1	Thread-Gruppe	21
3.2	Logik Controller	21
3.2.1	If-Controller	22
3.2.2	Loop-Controller	22
3.3	Konfigurations Elemente	22
3.3.1	Benutzer Definierte Variablen	22
3.3.2	CSV Einstellungen	23
3.3.3	HTTP Managers	23
3.3.4	Zähler (Counter)	24
3.4	Sampler	24
3.4.1	HTTP Request	24
3.5	Präprozessoren	25

3.5.1	JSR223 Präprozessor	25
3.6	Postprozessoren	26
3.6.1	XPath Extractor	26
3.7	Überprüfung	26
3.7.1	Versicherte Antwort	27
3.8	Listeners	27
3.8.1	View Results Tree	27
3.9	Funktionsbausteine	28
3.9.1	Random	28
4	Auswertung	29
4.1	JConsole	29
4.2	Resttool	30
4.3	Test 1: 50 Benutzer	31
4.4	Test 2: 100 Benutzer	32
4.5	Test 3: 200 Benutzer	32
4.6	Test 4: 200 Benutzer à 50 Sekunden	33
5	Reflexion	35
5.1	Verlauf	35
5.2	Arbeitsmethodik	35
5.3	Kommunikation	35
A	Zeitraffer - Status der Arbeiten	36
B	Glossar	38
B.1	Glossar	38
C	Literatur	40
C.1	Literatur	40

Tabellenverzeichnis

1.1	Muss-Kriterien	3
1.2	Donnerstag 7. März	5
1.3	Montag 11. März	6
1.4	Dienstag 12. März	6
1.5	Mittwoch 13. März	6
1.6	Donnerstag 14. März	7
1.7	Samstag 16. März	7
1.8	Montag 18. März	8
1.9	Dienstag 19. März	8
1.10	Mittwoch 20. März	8
1.11	Donnerstag 21. März	9
1.12	Montag 25. März	9
1.13	Dienstag 26. März	9
3.1	Suffix Übersicht	26
4.1	openolat LMS - Java Parameter	29
4.2	Auslastung mit 50 Benutzer	31
4.3	Auslastung mit 100 Benutzer	32
4.4	Auslastung mit 200 Benutzer	33
4.5	Auslastung mit 200 Benutzer à 50 Sekunden	34

Vorwort

Dieses Dokument kam auf Grund meines Lehrabschlusses zustande, der eine Praktische Abschlussarbeit vorsieht. Das Kapitel 1 gibt eine detaillierte Einsicht in deren Umfang.

Meine Java Karriere begann Ende 2001 mit JavaSE-1.2, indem ich das Buch Goto Java 2 las und umsetzte. Damals war auf meinem Computer Microsoft Windows 98 vorinstalliert und die Classpath Probleme schienen unendlich. Zudem waren die Ressourcen sehr begrenzt mit einer 6 GB Festplatte. Deshalb blieb ich bei einfachen Texteditors mit Syntax-Highlighting für einen längeren Zeitraum und compilierte meine Programme vom Terminal aus.

Gut zwölf Jahre später darf ich für eine renommierte E-Learning Plattform openolat Workload-Tests schreiben. Diese sind insofern wichtig um Engpässe im System zu erkennen und somit für die verfügbaren Ressourcen zu optimieren. Des weiteren können allfällige Memory-Leaks oder Dead-Locks erkannt werden. Ich bewege mich also immer noch am Limit der Leistung.

Kapitel 1

Einleitung

In diesem Kapitel wird einen Überblick auf die Thematik verschafft und auf die Aufgabenstellung detailliert eingegangen. Problemstellungen und Lösungsansätze sollen ersichtlich sein.

1.1 Ausgangslage

Openolat ist ein eLearning Management System, welches über verschiedene Schnittstellen verfügt. RESTful Webservice, webdav Dateitransfer, AJAX und statisches HTML sind die Schnittstellen mit denen der Benutzer unter Umständen konfrontiert ist. Es läuft als Cluster-fähige Webanwendung auf einem Java Servlet Container, getestet wurden Apache Tomcat und Oracle Glassfish.

Diese verteilte Anwendung gilt es zu optimieren. Damit dies erreicht werden kann, benötigt es Last auf der entsprechenden Schnittstelle, um eventuelle Engpässe zu evaluieren. Apache JMeter scheint das ideale Werkzeug zu sein, da es für frühere Versionen lauffähige JMeter Tests gab, die jedoch nicht mehr aktuell sind.

1.2 Auftrag

Die Aufgabenstellung ist vorgegeben und klar definiert. Jedoch lässt sie sich zeitlich nicht abschliessen, weil Änderungen an Openolat können dazu führen, dass die Tests unbrauchbar werden. Des weiteren wird gewünscht, dass weitere Elemente getestet werden, als in der Aufgabenstellung definiert sind, bekannte Tests sind als Kann-Kriterien in denn Kriterienkatalog eingeflossen.

Der Login Test wurde bereits vor der IPA von einem anderen Mitarbeiter portiert, dies wurde dementsprechend im Kriterienkatalog gestrichen.

1.2.1 Aufgabenstellung

Lieferobjekte / Resultate (welche Arbeit soll getan werden?)

- Installation und Konfiguration der Software Apache JMeter und OpenOLAT
- Login Testcase: Portierung eines bestehenden OLAT-Login JMeter Testcases auf die aktuellste Version von OpenOLAT
- eAssessment Testcase: Aufbau eines neuen Testcases zur Simulation einer elektronischen Prüfungssituation: Login, Aufruf eines Kurses, Aufruf eines eAssessments, eAssessment durchführen, Logout
- Skript um Testbenutzer und Testressourcen (eAssessment, Kurs) vor Testdurchführung anzulegen

Prüfbaren/messbaren Ziele

- Der eAssessment Testcase läuft vollständig und fehlerfrei mit einem Testbenutzer durch

- Der eAssessment Testcase kann wiederholt und mit mehreren Testbenutzern gleichzeitig aufgerufen werden

Durchzuführende Tests

- Testdurchlauf mit 50 gleichzeitigen Benutzern, Überwachen von CPU- und Memoryauslastung
- Schrittweise Erhöhung der Last bis Testsystem beginnt Fehler zu produzieren oder die Hardware vollständig ausgelastet ist

1.2.2 Kriterienkatalog

Der Kriterienkatalog zeigt die Aufgabenstellung in Teilaufgaben gegliedert dar. Dabei wird zwischen Kann- und Muss-Kriterien unterschieden. Die Zeit zum Dokumentieren ist nicht mit eingerechnet, ausgeschlossen API-Referenz und Quellcode Kommentare. Sie beträgt etwa 40 bis 50 Prozent und ist somit deckungsgleich mit den verrichteten Arbeiten. Auf die Kann-Kriterien wurde verzichtet da sie nicht zur eigentlichen Arbeit gehören.

1.2.2.1 Muss-Kriterien

In der folgenden Tabelle werden Ziele aufgeführt, die zwingend erreicht werden müssen.

Schlüssel	Aufwand[h]	Tätigkeit	Beschreibung
M01	2.0	Zeitplan erstellen und nach verfolgen.	Es soll ein 10 Arbeitstage dauernder Zeitplan erstellt werden. Mit dessen Hilfe der Abgabetermin eingehalten werden soll. Des weiteren sollen die Arbeitszeiten nachvollziehbar sein. Arbeitszeiten werden im Zeitplan und Journal festgehalten, als auch im Intervall-Zeitplan.
M02	5.0	Journal führen.	Die verrichteten Arbeiten sollen nachvollziehbar sein und bei nicht Erreichen der Zielsetzung soll es Aufschluss über die Ursache geben.
M03	6.0	In JMeter einarbeiten.	JMeter Dokumentation lesen und gegen statische Inhalte testen. Die Tests sind kurz zu halten und werden später nicht mehr benötigt, sie können jedoch als Hilfestellung verwendet werden.
M04	2.0	Arbeitsumgebung aufsetzen und konfigurieren.	Die Installation geschieht idealerweise einmalig. Gewisse Konfigurationen müssen allenfalls nachträglich erledigt werden oder im Test Kontext angepasst werden.
M05.01	3.0	Java Programm für das Erstellen von Benutzern.	Es soll der vorhandene REST-Client für das Erstellen von Benutzer erweitert werden. Die erstellten Benutzer und deren Passwörter müssen in eine CSV Datei exportiert werden.
M05.02	2.0	Java Programm für das importieren eines Kurses.	Es soll der vorhandene REST-Client für das Importieren von Kursen erweitert werden. Die Repository Entry ID der importierten Kurse soll in eine CSV Datei gespeichert werden, damit man sie mit JMeter auslesen und verwenden kann.
M06	2.0	Umfangreicher eAssessment Test erstellen.	Eigens für den Loadtest soll ein eAssessment Test erstellt werden, der alle interaktiven und Layout Komponenten eines Tests beinhalten.
M07	14.0	JMeter Loadtest für eAssessment Test erstellen.	Der Test wird Programmierelemente beinhalten müssen, wie Bedingungen, Schleifen und Parser.
M08	10.5	JMeter Testdurchläufe und Evaluieren mittels JConsole.	Das Verhalten des Systems mit unterschiedlichen Konfigurationen testen und dokumentieren.

Tabelle 1.1: Muss-Kriterien

1.2.3 Zeitplan

	07.03.	11.03.	12.03.	13.03.	14.03.	18.03.	19.03.	20.03.	21.03.	25.03.
M01										
M02										
M03										
M04										
M05.01										
M05.02										
M06										
M07										
M08										

Grobe Zeitplanung im Raster.

1.3 Vorkenntnisse

Mit den folgenden Technologien kenne ich mich bereits aus.

- maven
- eclipse
- JavaSE
- RESTful

- HTML/CSS
- XML/XPath
- openolat LMS

Es folgen Technologien von denen ich keine besondere Vorkenntnisse besitze.

- JMeter
- JMX/JConsole

1.4 Journal

Arbeiten, Zeitaufwand und Reflexion sind hier aufgeführt.

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
-	0.75	Besprechung über den Ablauf der IPA mit Florian Gnägi. Es wurde der ist und soll Zustand der Testabdeckung besprochen. Der eAssessment Test soll umfangreich sein und der Einschreibebaustein für einen Kurs soll getestet werden.	Der Login Test wurde bereits portiert und es steht allenfalls noch dessen Reimplementierung aus. Der Login Test benützt die Full-Page Refresh Schnittstelle von openolat er soll jedoch in Zukunft mit der Web 2.0 Schnittstelle funktionieren. Doch Priorität hat der eAssessment und der Einschreibe Test.
M01	0.5	Erstellen eines Zeitraffers auf einem A3 Blatt für interne Verwendung.	Dieser soll auf Wunsch immer die nächsten zwei Tage geplant werden, damit der Verlauf besser Nachvollziehbar ist.
-	0.25	Scrum.	-
M04	1.0	Ordner Struktur erstellen und vorhandene Ressourcen kopieren. Installieren von JMeter und die Konfiguration der Datenbank und des Servers festhalten, mit diverse READMEs als Produkt.	Die READMEs werden in die Dokumentation miteinbezogen.
-	3.0	Dokumentenstruktur anlegen und Docbook XML Kenntnisse auffrischen.	Ich habe mich für das Verfassen der Dokumentation für Docbook XML entschieden, da der Quick-Start Guide in einem internettauglichem Format gespeichert werden soll und um mich nicht mit zwei verschiedenen Textverarbeitungsprogrammen auseinandersetzen zu müssen.
-	0.5	Gespräch mit Herr U. Niederhäuser und Austausch betreffend Wegleitung.	Allfällige Fragen mit Herr A. Knöpfli klären.
-	0.5	Smalltalk mit Anwesenden.	-
-	1.0	Gespräch mit Herr U. Niederhäuser, F. Gnägi und A. Knöpfli.	Während den geplanten 80 Stunde sollte auch Zeit für die Präsentation da sein.
-	0.5	Journal führen.	-

Tabelle 1.2: Donnerstag 7. März

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
M01	1.0	Zeitplan Diagramm erstellen.	Ich habe die Zeitplan Grafik mit SVG erstellt. Das ist zwar eine aufwendige Methode, aber bietet mir das Optimum an gestalterischer Freiheit.
-	3.5	Erstes Kapitel fertig stellen.	-
M04	2.0	Kapitel 2 Struktur anlegen, Konfigurationen einbinden sowie Shell Kommandos festhalten.	Weil die Planung noch nicht fertig war wurde parallel installiert und konfiguriert. Damit meine Erfahrungen getreu wiedergegeben werden können, habe ich die Dokumentation ebenfalls vorgezogen.
M03	0.5	Lesen der relevanten Seiten aus der JMeter Dokumentation.	Ist sehr knapp beschrieben und ist viel weniger als erwartet.
M03	1.0	JMeter mit statischem Inhalt testen.	Dafür verwendete ich meine private Homepage weedlight.ch.

Tabelle 1.3: Montag 11. März

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
M03	2.0	JMeter mit statischem Inhalt testen.	Ich habe mit XPath erfolgreich einen Link ausgelesen und diesen in einer Variable weiterverwendet. Die XPath Implementation scheint sich aber zu bisher Bekannten zu unterscheiden - evtl. Versionsunterschied. Hat sich bis jetzt nur beim Gruppieren bemerkbar gemacht.
M04	2.0	Dokumentation ausführen und Konfiguration anpassen.	-
-	3	Stromausfall!	Wegen Stromausfall war das Arbeiten unmöglich, um 17.15 Uhr Abbruch weil Behebung nicht absehbar schien.

Tabelle 1.4: Dienstag 12. März

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
M04	2.0	Dokumentation ausführen und Konfiguration anpassen.	Im Hinblick zu meinem nächsten Arbeitsschritt fiel mir ein, dass in <code>olat.local.properties</code> der RESTful Webservice eingeschaltet sein muss. Das Glossar habe ich in diesem Arbeitsschritt begonnen.
M02	0.5	Journal führen.	Obwohl ich für gestern das Journal nachträglich führen muss, ist es authentisch wiedergegeben.
-	0.25	Kommunikation.	Mail schreiben wegen Arbeitsausfall und Stand der IPA übermitteln.
M05.01 & M05.02	5.75	Restclient implementieren in Java. Skeleton der Klassen sowie API-Dokumentation erstellen.	Ich habe noch nie so ausführlich dokumentiert, aber das Ausführen macht Spass. Des weiteren habe ich mir das javadoc Doclet dbdoclet angeschaut. Dies ist ein Softwarepaket, das mit einer GUI kommt, aber die nicht funktioniert.

Tabelle 1.5: Mittwoch 13. März

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
M05.01 & M05.02	3.5	API-Dokumentation ergänzen, Properties Datei anwenden und die <code>generateUser</code> sowie <code>importCourse</code> Methoden implementieren. <code>CsvHelper</code> Klasse implementieren.	Mit dem Aufrufen von javadoc und dem dbdoclet war ich heute erfolgreich und habe diese meiner Dokumentation hinzugefügt.
M06	1.25	Umfangreicher eAssessment Test erstellen.	Ich habe in der angegebenen Zeit nur 18 Fragen verteilt auf 6 Sektionen geschafft. Dafür habe ich etwas menschenleserliches, was mir bei der Fehlersuche im JMeter Test wahrscheinlich hilfreich sein könnte.
M05.01 & M05.02	1.5	REST-API Pfad Probleme ausfindig machen und beheben.	Die <code>RestConnection</code> Klasse erhielt kein Security Token gab, aber beim Login Status Code 200 zurück. Das machte das Aufspüren des Fehlers enorm aufwendig. Die Fehlerursache lag in einer Methode, die ich nicht geschrieben habe. Der Hauptentwickler wies mich jedoch darauf hin, dass ich auf einer richtigen Openolat Instanz andere Pfade und Ports habe, als in den Selenium Tests.
M07	1.25	Loadtest erstellen für eAssessment Test.	Erfolgreicher Import des eAssessment Tests über RESTful Webservice, CSV Datei scheint in Ordnung zu sein. Generieren der Benutzer und deren Attribute in eine CSV Datei schreiben verlief ohne Probleme. Der Login mit JMeter und den 100 Benutzer war erfolgreich.
M02	0.5	Journal führen.	-

Tabelle 1.6: Donnerstag 14. März

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
-	1.5	Nachforschungen bezüglich Konvertierungstool für Docbook XML Publikation.	Das Evaluieren fiel auf dlatex als PDF Generierungswerkzeug. Ich bitte darum diese Zeit nicht für die IPA anzurechnen, da ich Docbook XML auch für ein anderes FOSS Projekt einsetze.

Tabelle 1.7: Samstag 16. März

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
M07	7.75	JMeter Loadtest implementieren.	Ich hatte heute diverse XPath Probleme, die ich jedoch lösen konnte. Diese war sehr Zeitraubend, dafür habe ich etwas dazu gelernt. Des weiteren musste ich herausfinden, dass im For-Controller die HTTP-Requests des vorangehenden Http-Sampler nicht verfügbar sind. Dies schliesse ich darauf zurück, dass ich bei jedem XPath-Postcontroller keine Daten erhielt. Morgen werde ich viel dokumentieren können.
M02	0.25	Journal führen.	-

Tabelle 1.8: Montag 18. März

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
M07	6.75	JMeter Loadtest implementieren.	Es gab weitere Probleme mit XPath, deshalb werde ich einen Patch für JMeter schreiben.
-	1.0	Treffen mit A. Knöpfli	Inputs betreffend Dokument Aufbau nach Wegleitung zur Kenntnis genommen. Es machte Spass einem Aussenstehenden die Arbeit zu zeigen.
M02	0.25	Journal führen.	-

Tabelle 1.9: Dienstag 19. März

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
M07	7.75	JMeter Loadtest implementieren.	Beim Erstellen des Patches war es unumgänglich den Quellcode anzuschauen, dabei viel mir auf, dass mittels einem bestimmten Suffix auch die Anzahl Treffer ausgelesen werden können. Also war die Mühe nicht umsonst, weil vorhergegangene Recherchen erfolglos gewesen sind. Ich habe heute das Ausfüllen aller Fragetypen fertig gestellt. Weiteres habe ich mit der Dokumentation von JMeter begonnen.
M02	0.25	Journal führen.	-

Tabelle 1.10: Mittwoch 20. März

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
M07	4.75	JMeter Loadtest dokumentieren.	Ich habe die Dokumentation am Stück geschrieben. Danach hatte ich die Mittagspause absolut nötig. Ich habe jemanden der mit JMeter Erfahrungen hat, angefragt ob er das darauf bezogene Kapitel gegenlesen würde.
M08	3.0	Auswerten der JMeter Tests.	Ich machte mir Gedanken über eine automatisierte Generierung von Testdaten und habe Recherchen bezüglich Überwachungswerkzeuge angestellt. Dabei musste ich feststellen, dass ich ein weiteres Tool benötige, um die Last des Betriebssystems zu überwachen. Ich traf die Entscheidung eine kleine RMI Anwendung zu programmieren, die über das Wochenende Server Neustarts durchführt, bei einem Crash das System neu aufsetzt und die Testbedingungen anpasst.
M02	0.25	Journal führen.	-

Tabelle 1.11: Donnerstag 21. März

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
M08	8.0	Auswerten der JMeter Tests.	Ideen von einer verteilten Rekonfigurationssoftware haben wir vorerst verworfen. Vier Tests durchführen und dokumentieren.
-	1.5	Dokumentation bereinigen.	Rechtschreibfehler beheben und Glossar fertigstellen.
M02	0.25	Journal führen.	-

Tabelle 1.12: Montag 25. März

Schlüssel	Aufwand[h]	Tätigkeit	Reflexion
-	1.5	Quick-Start Guide schreiben.	Anhand der Dokumentation eine Kurzübersicht auf Englisch verfassen.

Tabelle 1.13: Dienstag 26. März

Kapitel 2

Umgebung

Es wird Vorausgesetzt, dass eine funktionsfähige Installation eines Unixartigen-Betriebssystem vorhanden ist. Hier wird anhand einer Debian GNU/Linux "stable" Distribution erklärt, wie man die Arbeitsumgebung aufsetzt.

2.1 Debian GNU/Linux

Bevor wir das Testing Repository hinzufügen legen wir fest, dass das Standard Repository "stable" ist. Falls dies unterlassen wird, werden Sie nach dem nächsten Upgrade eine "testing" Distribution haben.

Beispiel 2.1 /etc/apt/apt.conf

```
APT::Default-Release "stable";
```

Nachdem wir das Standard Repository definiert haben, ist es sicher das "testing" Repository hinzuzufügen.

Beispiel 2.2 /etc/apt/sources.list

```
#

# deb cdrom:[Debian GNU/Linux 6.0.7 _Squeeze_ - Official amd64 CD Binary-1 ↔
20130223-14:06]/ squeeze main

# deb cdrom:[Debian GNU/Linux 6.0.7 _Squeeze_ - Official amd64 CD Binary-1 ↔
20130223-14:06]/ squeeze main

deb http://ftp.ch.debian.org/debian/ stable main contrib non-free
deb-src http://ftp.ch.debian.org/debian/ stable main contrib non-free

deb http://security.debian.org/ stable/updates main contrib non-free
deb-src http://security.debian.org/ stable/updates main contrib non-free

# testing repo
deb http://ftp.ch.debian.org/debian/ testing main contrib non-free
deb-src http://ftp.ch.debian.org/debian/ testing main contrib non-free
```

Wenn jetzt in der Ausgabe ersichtlich ist, dass das testing Repository abgerufen wird, dann können wir mit der Installation fortfahren.

Beispiel 2.3 Aktualisiere Paketdatenbank

```
#!/bin/bash

apt-get update
```

Beispiel 2.4 Installiere Software

```
#!/bin/bash

apt-get -t testing install java7-jdk mysql-server mysql-client tomcat7 maven2
```

2.2 Apache Tomcat

Apache Tomcat ist ein Java Servlet Container, das ist ein Webserver der unter anderem dynamische Seiten enthalten kann. Dabei stehen einem die volle Funktionalität des JRE zur Verfügung.

Wir konfigurieren Apache Tomcat so, dass er auf Port 80 horcht. Damit das möglich ist muss in der Konfigurationsdatei `/etc/default/tomcat7` der Parameter `AUTHBIND` auf `yes` gestellt sein. Desweiteren ändern wir die RAM-Belegungsparameter, damit die Java Virtual Machine mehr Arbeitsspeicher zur Verfügung hat.

Beispiel 2.5 `/etc/default/tomcat7`

```
# Run Tomcat as this user ID. Not setting this or leaving it blank will use the
# default of tomcat7.
TOMCAT7_USER=tomcat7

# Run Tomcat as this group ID. Not setting this or leaving it blank will use
# the default of tomcat7.
TOMCAT7_GROUP=tomcat7

# The home directory of the Java development kit (JDK). You need at least
# JDK version 1.5. If JAVA_HOME is not set, some common directories for
# OpenJDK, the Sun JDK, and various J2SE 1.5 versions are tried.
#JAVA_HOME=/usr/lib/jvm/openjdk-6-jdk

# You may pass JVM startup parameters to Java here. If unset, the default
# options will be: -Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC
#
# Use "-XX:+UseConcMarkSweepGC" to enable the CMS garbage collector (improved
# response time). If you use that option and you run Tomcat on a machine with
# exactly one CPU chip that contains one or two cores, you should also add
# the "-XX:+CMSIncrementalMode" option.
JAVA_OPTS="-Djava.awt.headless=true -XX:MaxPermSize=128m -Xmx512m -Xms256m -XX:+ ↵
    UseConcMarkSweepGC"

# To enable remote debugging uncomment the following line.
# You will then be able to use a java debugger on port 8000.
#JAVA_OPTS="${JAVA_OPTS} -Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server ↵
    =y,suspend=n"

# Java compiler to use for translating JavaServer Pages (JSPs). You can use all
# compilers that are accepted by Ant's build.compiler property.
```

```
#JSP_COMPILER=javac

# Use the Java security manager? (yes/no, default: no)
#TOMCAT7_SECURITY=no

# Number of days to keep logfiles in /var/log/tomcat7. Default is 14 days.
#LOGFILE_DAYS=14

# Location of the JVM temporary directory
# WARNING: This directory will be destroyed and recreated at every startup !
#JVM_TMP=/tmp/tomcat7-temp

# If you run Tomcat on port numbers that are all higher than 1023, then you
# do not need authbind. It is used for binding Tomcat to lower port numbers.
# NOTE: authbind works only with IPv4. Do not enable it when using IPv6.
# (yes/no, default: no)
AUTHBIND=yes
```

In der Datei `/etc/tomcat7/server.xml` können wir nun das Attribut `port` auf `80` setzen.

Beispiel 2.6 `/etc/tomcat7/server.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<!-- Note: A "Server" is not itself a "Container", so you may not
define subcomponents such as "Valves" at this level.
Documentation at /docs/config/server.html
-->
<Server port="8005" shutdown="SHUTDOWN">
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
  <!--APR library loader. Documentation at /docs/apr.html -->
  <!--
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="
on" />
  -->
  <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper
-howto.html -->
  <Listener className="org.apache.catalina.core.JasperListener" />
  <!-- Prevent memory leaks due to use of particular java/javax APIs-->
```

```
<Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" ↵
/>
<Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" ↵
"/>
<Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" ↵
/>

<!-- Global JNDI resources
Documentation at /docs/jndi-resources-howto.html
-->
<GlobalNamingResources>
  <!-- Editable user database that can also be used by
  UserDatabaseRealm to authenticate users
  -->
  <Resource name="UserDatabase" auth="Container"
    type="org.apache.catalina.UserDatabase"
    description="User database that can be updated and saved"
    factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
    pathname="conf/tomcat-users.xml" />
</GlobalNamingResources>

<!-- A "Service" is a collection of one or more "Connectors" that share
a single "Container" Note: A "Service" is not itself a "Container",
so you may not define subcomponents such as "Valves" at this level.
Documentation at /docs/config/service.html
-->
<Service name="Catalina">

  <!--The connectors can use a shared executor, you can define one or more named ↵
  thread pools-->
  <!--
  <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
    maxThreads="150" minSpareThreads="4"/>
  -->

  <!-- A "Connector" represents an endpoint by which requests are received
  and responses are returned. Documentation at :
  Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
  Java AJP Connector: /docs/config/ajp.html
  APR (HTTP/AJP) Connector: /docs/apr.html
  Define a non-SSL HTTP/1.1 Connector on port 8080
  -->
  <Connector port="80" protocol="HTTP/1.1"
    connectionTimeout="20000"
    URIEncoding="UTF-8"
    redirectPort="8443" />
  <!-- A "Connector" using the shared thread pool-->
  <!--
  <Connector executor="tomcatThreadPool"
    port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443" />
  -->
  <!-- Define a SSL HTTP/1.1 Connector on port 8443
  This connector uses the JSSE configuration, when using APR, the
  connector should be using the OpenSSL style configuration
```

```
        described in the APR documentation -->
<!--
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
        maxThreads="150" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS" />
-->

<!-- Define an AJP 1.3 Connector on port 8009 -->
<!--
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
-->

<!-- An Engine represents the entry point (within Catalina) that processes
every request. The Engine implementation for Tomcat stand alone
analyzes the HTTP headers included with the request, and passes them
on to the appropriate Host (virtual host).
Documentation at /docs/config/engine.html -->

<!-- You should set jvmRoute to support load-balancing via AJP ie :
<Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">
-->
<Engine name="Catalina" defaultHost="localhost">

    <!--For clustering, please take a look at documentation at:
        /docs/cluster-howto.html (simple how to)
        /docs/config/cluster.html (reference documentation) -->
    <!--
    <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
    -->

    <!-- Use the LockOutRealm to prevent attempts to guess user passwords
        via a brute-force attack -->
    <Realm className="org.apache.catalina.realm.LockOutRealm">
        <!-- This Realm uses the UserDatabase configured in the global JNDI
            resources under the key "UserDatabase". Any edits
            that are performed against this UserDatabase are immediately
            available for use by the Realm. -->
        <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
            resourceName="UserDatabase"/>
    </Realm>

    <Host name="localhost" appBase="webapps"
        unpackWARs="true" autoDeploy="true">

        <!-- SingleSignOn valve, share authentication between web applications
            Documentation at: /docs/config/valve.html -->
        <!--
        <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
        -->

        <!-- Access log processes all example.
            Documentation at: /docs/config/valve.html
            Note: The pattern used is equivalent to using pattern="common" -->
        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="↵
            logs"
            prefix="localhost_access_log." suffix=".txt"
```

```
        pattern="%h %l %u %t &quot;%r&quot; %s %b" />
    </Host>
</Engine>
</Service>
</Server>
```

2.3 MySQL

Openolat LMS benützt die Java Bibliothek Hibernate um mit den verschiedenen unterstützten Datenbankimplementationen zu arbeiten. Für einen einfachen Setup verwenden wir jedoch MySQL.

Als erstes starten wir die interaktive Kommandoeingabe von mysql.

Beispiel 2.7 Client starten

```
#!/bin/bash

mysql -uroot -p
```

Im MySQL-Client legen wir den Benutzer openolat an, den wir später für die Konfiguration von openolat LMS verwenden.

Beispiel 2.8 openolat Datenbank Benutzer

```
grant all privileges on openolat.* to openolat@localhost identified by 'openolat';
create database openolat CHARSET 'utf8';
```

2.4 openolat LMS

Openolat LMS ist OSS und kann dementsprechen an die individuellen Bedürfnisse angepasst werden.

Um eine WAR Datei mit der gesamten Webapplikation zu erhalten klonen wir das Mercurial Repository. Danach können wir sie compilieren und verpacken.

Beispiel 2.9 Mercurial Repository klonen

```
#!/bin/bash

cd
hg clone http://hg.openolat.org/openolat openolat
```

Um die Tabellen der openolat LMS Datenbank anzulegen, führen wir das heruntergeladene SQL Script aus.

Beispiel 2.10 MySQL Script ausführen

```
#!/bin/bash

mysql -uopenolat -popenolat openolat < ~/openolat/src/main/resources/database/mysql/ ↵
    setupDatabase.sql
```

Diese Konfigurationsdatei unterscheidet sich zur Beispiel Konfiguration indem, dass der Loadtest Modus und der RESTful Webservice eingeschalten sind. Diese Konfigurationsdatei muss anschliessend nach `src/main/java/` kopiert werden.

Beispiel 2.11 olat.local.properties

```
#####
#
# To start, copy this file to olat.local.properties and modify it to fit your ↵
# needs. Please have a look
# at the file src/main/resources/serviceconfig/olat.properties to get the full ↵
# list of configuration
# options.
#
#####
allow.loadtest.mode=true
restapi.enable=true

#####
# Application data directory.
#####

# runtime application data directory. Tomcat user needs R/W permissions here
userdata.dir=/opt/openolat/olatdata

#####
# Database settings
#####

# supported vendors currently include "mysql" and "postgresql"
db.vendor=mysql
# here you have two options: when you set auto.upgrade.database to true
# the alter scripts in /src/main/resources/database/**/alter*.sql are
# executed automatically. For most cases this is fine. If you set it to
# false however, you must execute those scripts yourself BEFORE starting
# OpenOLAT after an update.
auto.upgrade.database=true

# the name of the application database
db.name=openolat
# the name of the OLAT database user
db.user=openolat
# the password of the OLAT database user
db.pass=openolat
# JDBC options (e.g., to set character channel behavior etc.)
db.jdbc.options=useUnicode=true&characterEncoding=UTF-8

#####
# Web application container (e.g., Tomcat) settings
#####

# hosted application fully qualified domain name (e.g., DNS CNAME)
# omit references to protocol/scheme (e.g., HTTP(S))
server.domainname=localhost
# the port on which the container is listening
server.port=80
```

```
#####
# SMTP (mail) settings
#####

# mail support can be disabled by leaving the following entry blank or
# setting it to the keyword 'disabled' (without quotes!)
smtp.host=localhost
# if required by your local SMTP you may need to provide credentials
smtp.user=openolat
smtp.pwd=openolat
# system mails will be sent from this address (from local domain with valid ↵
# reverse dns):
fromemail=no-reply@your.domain
# set this email to a mail address in your domain (used as reply-to address)
adminemail=webmaster@your.domain
# set this email to a mail address in your domain (used for 'ask for help here' ↵
# type of messages that do not have special address)
supportemail=${adminemail}
# set this email to a mail address in your domain (used to tell users how to apply ↵
# for more quote disk space)
quotaemail=${supportemail}
# set this email to a mail address in your domain (used to notify when users are ↵
# deleted from the system)
deleteuseremail=${supportemail}
# set this email to a mail address in your domain (used for red-screen error ↵
# reports)
erroreemail=${adminemail}

#####
# OLAT identity settings
#####

# OLAT instance ID (effects a unique namespace for addressable items)
# ID should be no longer than 10 characters! This needs to be unique
# amongst nodes within a cluster, if you are not clustering then you
# can leave this value as is.
instance.id=myopenolat

#####
# Misc / Developer settings
#####

# for developers set to true - velocity pages are NOT cached
olat.debug=false
# for developers set to false - i18n files are not cached
localization.cache=true
# path to the source code. Use this together with olat.debug=true in a
# development setup. If set properly, velocity templates, i18n files and
# static files like CSS will be reloaded from the source code on each request
# without redeployment of the entire webapp.
#project.build.home.directory=/Users/srosse/workspace/OpenOLAT

user.generateTestUsers=false
# disable full text indexer at startup for development environment
generate.index.at.startup=false
```

```
# only set this if you do not want unit tests to be performed during the build
skip.unit.tests=true
# set to false if you do not require special fonts
unpack.fonts=false
# when running in eclipse use "INFO, syslog, A1", for production use "INFO, syslog ↵
"
log.rootCategory = INFO, syslog, A1
# if enabled then the IM Server must be running before OLAT is started!
instantMessaging.enable=false
```

Nun müssen wir nur noch openolat LMS compilieren und als Webarchive verpacken. Maven nimmt uns da sehr viel Arbeit ab. Die WAR Datei muss dann von target/openolat-lms-8.4-SNAPSHOT.war nach /var/lib/tomcat7/webapps kopiert werden, damit sie entfaltet wird.

Beispiel 2.12 Maven Buildprozess

```
#!/bin/bash

cd openolat
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
mvn clean package
```

2.5 JMeter

Im Start Script von JMeter passen wir noch das Maximum der RAM-Belegung an für einen reibungslosen Ablauf. Bitte beachten Sie, dass man für den Loadtest noch groovy-all.jar benötigt und dieses in den lib Unterordner von JMeter kopiert.

Beispiel 2.13 JMeter Start Script

```
#!/bin/sh

## Licensed to the Apache Software Foundation (ASF) under one or more
## contributor license agreements. See the NOTICE file distributed with
## this work for additional information regarding copyright ownership.
## The ASF licenses this file to You under the Apache License, Version 2.0
## (the "License"); you may not use this file except in compliance with
## the License. You may obtain a copy of the License at
##
## http://www.apache.org/licenses/LICENSE-2.0
##
## Unless required by applicable law or agreed to in writing, software
## distributed under the License is distributed on an "AS IS" BASIS,
## WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
## See the License for the specific language governing permissions and
## limitations under the License.

## =====
## Environment variables:
## JVM_ARGS - optional java args, e.g. -Dprop=val
## =====

# The following should be reasonably good values for most tests running
```

```
# on Sun JVMs. Following is the analysis on which it is based. If it's total
# gibberish to you, please study my article at
# http://www.atg.com/portal/myatg/developer?paf\_dm=full&paf\_gear\_id=1100010&detailArticle= ↵
#   true&id=9606
#
# JMeter objects can generally be grouped into three life-length groups:
#
# - Per-sample objects (results, DOMs,...). An awful lot of those.
#   Life length of milliseconds to a few seconds.
#
# - Per-run objects (threads, listener data structures,...). Not that many
#   of those unless we use the table or tree listeners on heavy runs.
#   Life length of minutes to several hours, from creation to start of next run.
#
# - Per-work-session objects (test plans, GUIs,...).
#   Life length: for the life of the JVM.
#
# This is the base heap size -- you may increase or decrease it to fit your
# system's memory availability:
HEAP="-Xms2048m -Xmx2048m"
#
# There's an awful lot of per-sample objects allocated during test run, so we
# need a large eden to avoid too frequent scavenges -- you'll need to tune this
# down proportionally if you reduce the HEAP values above:
NEW="-XX:NewSize=128m -XX:MaxNewSize=128m"
#
# This ratio and target have been proven OK in tests with a specially high
# amount of per-sample objects (the HtmlParserHTMLParser tests):
# SURVIVOR="-XX:SurvivorRatio=8 -XX:TargetSurvivorRatio=50"
#
# Think about it: trying to keep per-run objects in tenuring definitely
# represents a cost, but where's the benefit? They won't disappear before
# the test is over, and at that point we will no longer care about performance.
#
# So we will have JMeter do an explicit Full GC before starting a test run,
# but then we won't make any effort (or spend any CPU) to keep objects
# in tenuring longer than the life of per-sample objects -- which is hopefully
# shorter than the period between two scavenges):
#
TENURING="-XX:MaxTenuringThreshold=2"
#
# This evacuation ratio is OK (see the comments for SURVIVOR) during test
# runs -- not so sure about operations that bring a lot of long-lived information into
# memory in a short period of time, such as loading tests or listener data files.
# Increase it if you experience OutOfMemory problems during those operations
# without having gone through a lot of Full GC-ing just before the OOM:
# EVACUATION="-XX:MaxLiveObjectEvacuationRatio=20%"
#
# Avoid the RMI-induced Full GCs to run too frequently -- once every ten minutes
# should be more than enough:
RMIGC="-Dsun.rmi.dgc.client.gcInterval=600000 -Dsun.rmi.dgc.server.gcInterval=600000"
#
# Increase MaxPermSize if you use a lot of Javascript in your Test Plan :
PERM="-XX:PermSize=64m -XX:MaxPermSize=128m"
#
# Finally, some tracing to help in case things go astray:
#DEBUG="-verbose:gc -XX:+PrintTenuringDistribution"
#
# Always dump on OOM (does not cost anything unless triggered)
DUMP="-XX:+HeapDumpOnOutOfMemoryError"
#
SERVER="-server"
```

```
ARGS="$SERVER $DUMP $HEAP $NEW $SURVIVOR $TENURING $EVACUATION $RMIGC $PERM"  
java $ARGS $JVM_ARGS -jar "`dirname "$0"`/ApacheJMeter.jar" "$@"
```

Kapitel 3

JMeter Loadtests

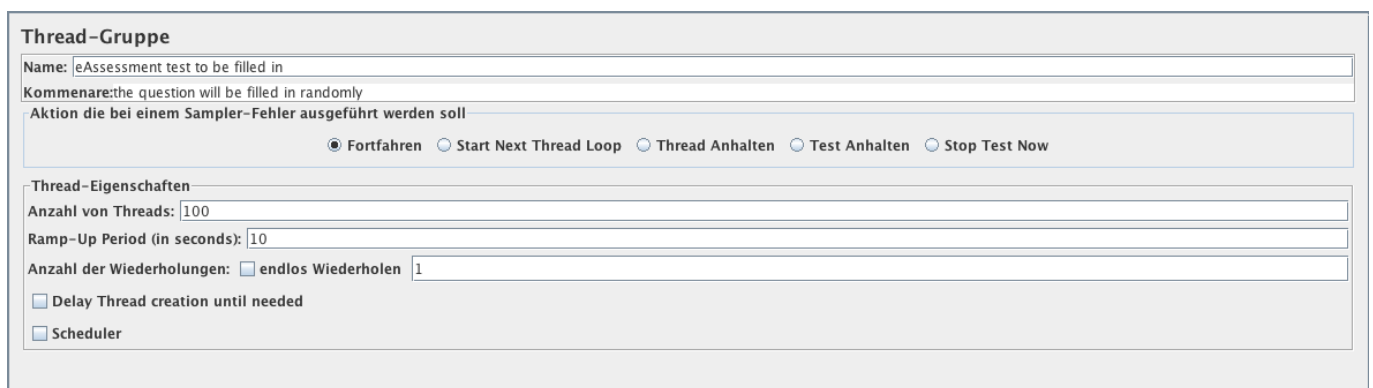
JMeter ist eine Open Source Software veröffentlicht unter Apache License Version 2.0 und ist somit frei verfügbar. Mit JMeter kann man automatisierte Tests von Netzwerk Anwendungen durchführen. Es bietet für verschiedene Problemstellungen das passende Werkzeug, wie zum Beispiel Konstrukte die aus der Software Programmierung kommen. Namentlich Variablen, Bedingungen, Schleifen und Zähler. Man hat auch die Möglichkeit diverse Scriptsprachen zu verwenden.

Es würde den Rahmen der Abschlussarbeit sprengen alle Komponenten von JMeter zu dokumentieren. Deshalb gehe ich speziell auf die im eAssessment Loadtest verwendeten Elemente ein. Der Einfachheit wegen verzichtete ich die unvollständig übersetzten Komponenten der GUI einzudeutschen.

3.1 Threads (Users)

Threads sind gleichzeitig ablaufende Handlungen, dabei hat jeder Thread seinen eigenen Geltungsbereich.

3.1.1 Thread-Gruppe



The screenshot shows the 'Thread-Gruppe' configuration window in JMeter. It includes fields for 'Name' (eAssessment test to be filled in), 'Kommentare' (the question will be filled in randomly), and 'Aktion die bei einem Sampler-Fehler ausgeführt werden soll'. Below these are radio buttons for 'Fortfahren' (selected), 'Start Next Thread Loop', 'Thread Anhalten', 'Test Anhalten', and 'Stop Test Now'. The 'Thread-Eigenschaften' section contains 'Anzahl von Threads' (100), 'Ramp-Up Period (in seconds)' (10), 'Anzahl der Wiederholungen' (1, with 'endlos Wiederholen' unchecked), 'Delay Thread creation until needed' (unchecked), and 'Scheduler' (unchecked).

Bildschirmfoto JMeter Thread Gruppe

Thread-Gruppen bietet einem die Möglichkeit Threads zu gruppieren, um diese anschliessend mit Verzögerungen zu starten. Das *Ramp up period* Feld gibt dabei an nach wie vielen Sekunden alle Threads gestartet sein sollen.

3.2 Logik Controller

Die Logik Bausteine bilden die in der Programmierung üblichen Sprachkonstrukte ab.

3.2.1 If-Controller

If-Controller

Name:

Kommentare:

Bedingung (Javascript)

☐ Interpret Condition as Variable Expression? ☐ Für alle Unterelemente auswerten?

Bildschirmfoto JMeter If-Controller

Der If-Controller braucht man um Ausschlüsse nach Bedingungen umzusetzen. Das das Feld *Bedingung (JavaScript)* muss eine gültiger Boolescher JavaScript Ausdruck sein. Es lassen sich JMeter Variablen in diesem Feld verwenden, welche jedoch mittels doppeltem Hochkomata als Escape-Sequenz umschlossen sein muss.

3.2.2 Loop-Controller

Schleifen-Controller (Loop Controller)

Name:

Kommentare:

Anzahl der Wiederholungen: ☐ endlos Wiederholen

Bildschirmfoto JMeter Loop-Controller

Der Loop Controller bildet eine spezielle For-Schleife ab, wo man lediglich die Anzahl Iterationen angeben kann. Sie können JMeter Variablen ohne Escape-Sequenz verwenden.

3.3 Konfigurations Elemente

Mit Konfigurations Elementen kann man Standard Werte und Variablen definieren. Diese werden am Anfang des Gültigkeitsbereiches verarbeitet, bevor irgendetwas verarbeitet oder generiert wird. Es gibt verschiedene Namenskonzepte, aber am naheliegendersten ist es das von Java zu übernehmen. Global sichtbare Variablen in Upper-Case und kontextbezogene in Camel-Case.

3.3.1 Benutzer Definierte Variablen

Benutzer definierte Variablen

Name:

Kommentare:

Name:	Wert	Description
host	debian-openolat	IP address or hostname
port	80	the port number to connect to
protocol	http	the default protocol to be used
contextPath	openolat-lms-8.4-SNAPSHOT	the context path where the web application was deployed to
baseURI	http://openolat-debian:80	the base URI of the web server
contextURI	http://openolat-debian:80/openolat-lms-8.4-SNAPSHOT	the URI of the web application

Detail Hinzufügen Add from Clipboard Löschen Up Down

Bildschirmfoto JMeter Benutzer Definierte Variablen

In diesem Konfigurations Element gibt man Konstanten an, die sich während der Laufzeit nicht ändern. Es eignet sich Ideal um Textbausteine zu erstellen. Diese Variablen werden analog zu Referenzen aufgerufen. `${host}` wäre demnach der Wert *debian-openolat*. Manche Felder von JMeter erlauben keine Variablen oder verlangen das Umschliessen mittels doppeltem Hochkomata.

3.3.2 CSV Einstellungen

CSV Einstellungen

Name: user attributes from user.csv

Kommentare: you may want to generate the user.csv file manually by invoking com.frentix.Resttool

Einstellungen der CSV Quellen

Dateiname: user.csv

Zeichensatz der Datei: UTF-8

Variablenname (getrennt durch Komma): userKey,userLogin,userPassword,userFirstname,userSurname,userEmail,userProfile

Trennzeichen ('\t' für Tab): ,

Allow quoted data?: True

Datei erneut einlesen?: True

Thread stoppen?: False

Sharing mode: Current thread group

Bildschirmfoto JMeter CSV Einstellungen

CSV Dateien eignen sich gut um Tabellen auszulesen. Dabei kann man den *Sharing Mode* auf *Current Thread Group* setzen, was dazu führt, dass jeder Thread eine Zeile erhält. Die Spalten werden durch das Feld *Variablenname* ausgelesen und den angegebenen Variablen zugewiesen.

3.3.3 HTTP Managers

HTTP Cookie Manager

Name: HTTP Cookie Manager

Kommentare: defaults for cookies

Options

☐ Cookies bei jedem Durchgang löschen?

Cookie Richtlinie: netscape Implementation: HC4CookieHandler

Anzahl der gespeicherten Cookies im Cookie Manager

Name:	Wert	Domain	Pfad:	Sicher

Hinzufügen Löschen Laden Speichern

Bildschirmfoto JMeter HTTP Cookie Manager

Die diversen HTTP Managers die vorhanden sind, bieten einem die Möglichkeit das Standardverhalten des HTTP Clients zu steuern. Es stehen folgende HTTP Managers zur Verfügung:

- HTTP Authorisierungs Manager
- HTTP Cache Manager
- HTTP Cookie Manager
- HTTP Header Manager

3.3.4 Zähler (Counter)

Zähler (Counter)

Name: counter

Kommentare: index of available questions

Start: 1

Zunahme: 1

Maximum: \${QUESTION_REF_matchNr}

Zahlenformat: 0

Reference Name: NTH_QUESTION_REF

☒ Zähler (Counter) für jeden Benutzer einzeln führen

☒ Reset counter on each Thread Group Iteration

Bildschirmfoto JMeter Zähler

Mit Zähler inkrementiert man Variablen um die angegebene Zunahme. Die Option *Zähler für jeden Benutzer einzeln führen* hat den Effekt, dass jeder Benutzer (Thread) eine eigene Variable erhält. *Reset counter on each Thread Group Iteration* führt dazu, dass der Zähler nicht immer weiter erhöht wird.

3.4 Sampler

Unter einem Sampler versteht man ein Werkzeug, das die eigentliche Netzwerk Kommunikation durchführt. JMeter bietet für verschiedene Protokolle Samplers an.

3.4.1 HTTP Request

HTTP Request

Name: open openolat LMS

Kommentare:

Web Server

Server Name oder IP: \${host} Port Number: \${port} Timeouts (milliseconds) Connect: Response:

HTTP Request

Implementation: HttpClient4 Protokoll (http): protocol Methode: GET Content Kodierung: UTF-8

Pfad: /\${contextPath}/dmz

☐ Automatisch Redirects folgen ☒ Folge Redirects ☒ Benutze KeepAlive ☐ Use multipart/form-data for POST ☐ Browser-compatible headers

Parameters Post Body

Parameter die mit dem Request gesendet werden:

Name:	Wert	Encodieren?	Gleichheitszeichen mit einbeziehen?

Detail Hinzufügen Add from Clipboard Löschen Up Down

Datei mit dem Request senden:

Dateiname: Wert des "name"-Attributes: MIME Type:

Hinzufügen Datei laden... Löschen

Proxy Server

Server Name oder IP: Port Number: Benutzername: Passwort:

Optionale Aufgaben

☐ Hole alle Bilder und Java Applets (nur HTML Dateien) ☐ Use concurrent pool. Size: 4 ☐ Als Überwacher (Monitor) benutzen ☐ Save response as MD5 hash?

Embedded URLs must match: Source IP address:

Bildschirmfoto JMeter HTTP Request

Der HTTP Request kann sehr vielseitig konfiguriert werden. Es lässt sich die HTTP Methode definieren, die benutzt werden soll und Parameter die mit dem Request verschickt werden sollen. Parameter entsprechen den Formularfelder einer Webpage.

3.5 Präprozessoren

Diese Verarbeitungswerkzeuge werden vor den eigentlichen Samplern ausgeführt. Sie dienen dazu Variablen aufzubereiten und um allfällige Berechnungen durchzuführen, die dann für den Sampler zur Verfügung gestellt werden.

3.5.1 JSR223 Präprozessor

Bildschirmfoto JMeter JSR223 Präprozessor

JSR223 ist eine Scripting Engine für Java. Die Groovy Erweiterung erlaubt es auch Java Snippets auszuführen.

Beispiel 3.1 GapTextProcessor.java

```
import java.lang.Math;
import java.lang.String;
import java.net.URLEncoder;
import org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase;
import org.apache.jmeter.protocol.http.util.HTTPArgument;

HTTPSamplerBase httpSamplerBase = (HTTPSamplerBase) sampler;

for(int i = 1; i <= Integer.valueOf(vars.get("GAP_TEXT_REF_matchNr")); i++){
    final String answer = vars.get("GAP_TEXT_REF_" + i);

    String name = "name=\"";
    int nameStart = answer.lastIndexOf(name) + name.length();
    int nameEnd = answer.indexOf("\"", nameStart);

    httpSamplerBase.addEncodedArgument(URLEncoder.encode(answer.substring(nameStart, ↵
        nameEnd), "UTF-8"), URLEncoder.encode(Double.toString(Math.floor(Math.random() ↵
        *100000)), "UTF-8"));
}

httpSamplerBase.addEncodedArgument(URLEncoder.encode("olat_fosm", "UTF-8"), "Save+answer+") ↵
;
```


3.6 Postprozessoren

Diese Prozessoren werden nach dem Samplerdurchgang ausgeführt. Diese Prozessoren erhalten die Antwort auf die vom Sampler ausgeführte Anfrage.

3.6.1 XPath Extractor

XPath Extractor

Name:

Kommentare:

Apply to:

☐ Main sample and sub-samples ☒ Main sample only ☐ Sub-samples only ☐ JMeter Variable

XML Parsing Options

☒ Use Tidy (tolerant parser) ☒ Quiet ☒ Report errors ☒ Show warnings

☐ Use Namespaces ☐ Validate XML ☐ Ignore Whitespace ☐ Fetch external DTDs

☐ Return entire XPath fragment instead of text content?

Reference Name:

XPath query:

Vorgabe-Wert:

Bildschirmfoto JMeter XPath Extractor

XPath dient zum Auslesen von Elementen und Attributen von XML-Konformen Baumstrukturen. Dieser Postprozessor hat die Option *Use tidy*, was dazu führt einen tolerante Verarbeitung von XML durchzuführen und ist somit anwendbar auf HTML.

Der XPath Extractor erzeugt mehrere Referenzen die mittels dem entsprechendem Suffix angesprochen werden. Nachstehende Tabelle zeigt die Suffixes anhand der angegebenen Referenz *TEXT_INPUT_NAME_REF* und dem aufgeführten XPath Ausdruck. Die Indizien können weitergeführt werden und hören nicht bei 2 auf.

Beispiel 3.2 XPath Ausdruck

```
//input[@type='text']/@name
```

Referenz	Bedeutung
TEXT_INPUT_NAME_REF_matchNr	Anzahl gefundener Elemente, die auf den XPath Ausdruck passen.
TEXT_INPUT_NAME_REF_1	Erstes Element, das auf den XPath Ausdruck passt.
TEXT_INPUT_NAME_REF_2	Zweites Element, das auf den XPath Ausdruck passt.
...	...

Tabelle 3.1: Suffix Übersicht

3.7 Überprüfung

Mit Überprüfungen kann man Annahmen machen über Inhalte. Falls eine Annahme fehlschlägt erhält man ein entsprechendes Feedback.

3.7.1 Versicherte Antwort

The screenshot shows the 'Versicherte Antwort' (Assured Response) configuration window in JMeter. It includes fields for 'Name' (search replay ID) and 'Kommentare' (assert search replay ID to be present). The 'Apply to' section has radio buttons for 'Main sample and sub-samples', 'Main sample only' (selected), 'Sub-samples only', and 'JMeter Variable'. The 'Zu testendes Antwort-Feld (Response-Feld)' section has radio buttons for 'Text-Antwort (Text-Response)' (selected), 'Document (text)', 'URL gesampled', 'Antwort-Code (Response-Code)', 'Antwort-Message', 'Antwort-Header (Response-Header)', and 'Status ignorieren'. The 'Regeln für passende Muster' section has radio buttons for 'Enthält' (selected), 'Entsprechungen', 'Gleicht', 'Teilzeichenkette (Substring)', and 'Nicht'. The 'Zu testende(s) Muster' section contains a list with the pattern 'o.fi900030001'. At the bottom are 'Hinzufügen' and 'Löschen' buttons.

Bildschirmfoto JMeter Versicherte Antwort

3.8 Listeners

Listeners arbeiten im globalen Gültigkeitsbereich. Sie werden dazu verwendet Auswertungen durchzuführen.

3.8.1 View Results Tree

The screenshot shows the 'View Results Tree' listener window in JMeter. It includes fields for 'Name' (View Results Tree) and 'Kommentare'. The 'Schreibe alle Daten in eine Datei' section has a text field for 'Dateinamen eingeben, oder eine existierende Datei auswählen.' and buttons for 'Datei laden...', 'Nur Loggen/Anzeigen' (with checkboxes for 'Fehler' and 'Erfolge'), and 'Konfigurieren'. The left pane shows a tree of test elements, with 'open current single choice question' selected. The right pane shows the 'Sampler result' tab with details for the selected element: Thread Name: eAssessment test to be filled in 1-25, Sample Start: 2013-03-20 17:26:36 MEZ, Load time: 535, Latency: 295, Size in bytes: 51287, Headers size in bytes: 305, Body size in bytes: 50982, Sample Count: 1, Error Count: 0, Response code: 200, Response message: OK. Below this are 'Response headers' and 'HTTPSampleResult fields'. At the bottom are 'Raw' and 'Parsed' tabs, and a 'Scroll automatically?' checkbox.

Bildschirmfoto JMeter View Results Tree

Visualisiert die Ergebnisse der Sampler und Annahmen als Baum.

3.9 Funktionsbausteine

Die meisten Felder von JMeter erlauben den Einsatz von Funktionen, die auch in Referenzen schreiben können. In der JMeter Dokumentation wird Upper-Case für Referenzen verwendet sowie der Suffix *_REF*. Funktionen können verschachtelt werden.

3.9.1 Random

`${__Random(mindestWert, maximalWert, [referenz])}` ist die Syntax der Random Funktion. Ersichtlich ist, dass die Referenz optional ist.

Kapitel 4

Auswertung

Ziel der Auswertung ist es zwei Situationen nachzustellen. Die eine wo der Betrieb von openolat LMS genug Ressourcen zur Verfügung hat und eine andere wo sie nicht genügen. Dies mit der Absicht für eine Hilfestellung zum Evaluieren von Fehlern.

Es gelten für beide Tests die gleich Rahmenbedingungen, es variiert lediglich die Benutzeranzahl und das verzögerte Einloggen in das System. Beide Tests laufen auf einem leeren System - die Datenbank wird vor jedem Durchlauf gelöscht und neu angelegt sowie das Datenverzeichnis gelöscht, zusätzlich das Tomcat 7 Logfile neu angelegt. Es wird vor und nach dem Test der Garbage Collector laufen gelassen.

```
drop database if exists openolat;
```

```
#!/bin/bash

# remove OpenOLAT directory and create it again.
rm -rf /opt/openolat/olatdata
mkdir /opt/openolat/olatdata
chown tomcat7.tomcat7 /opt/openolat/olatdata

# remove tomcat7 log file and create empty file
rm /var/lib/tomcat7/logs/catalina.out
touch /var/lib/tomcat7/logs/catalina.out
```

Der Server wird während einem Durchlauf mit Formulardaten von eAssessment Tests belastet.

Für die Gemeinsamen Bedingungen gelten folgende Java Parameter für Apache Tomcat:

Java Parameter	Beschreibung	Wert
-Xmx	Maximum Arbeitsspeicher begrenzung	128m
-Xms	Initiale Arbeitsspeicher Anlegung	256m
-XX:MaxPermSize	Maximum des permanenten Generierheaps	128m
-Xss	Thread Stackgrösse	1m (default)

Tabelle 4.1: openolat LMS - Java Parameter

Die JMeter Benutzer werden innerhalb von 50 Sekunden mit 50 Benutzern aufgeschaltet sein. Der Einfachheit halber passen wir das Aufschalten der Threads entsprechen der Benutzeranzahl an. Es wird ein Durchlauf durchgeführt.

4.1 JConsole

Damit wir Apache Tomcat über das Netzwerk überwachen können, müssen wir JMX aktivieren. Dazu bearbeiten wir `/etc/init.d/tomcat7` und setzen die Umgebungsvariable `CATALINA_OPTS` wie folgt - WARNUNG Authentifizierung abge-

schaltet:

```
JAVA_OPTS="-Djava.rmi.server.hostname=debian-openolat -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=1099 -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false"
```

Um die Authentifizierung einzuschalten, lesen Sie bitte folgenden Artikel: <http://tomcat.apache.org/tomcat-7.0-doc/monitoring.html>. Hinweis Änderungen an den Konfigurationsdateien oder Startscript erfordert einen Neustart.

```
#!/bin/bash

/etc/init.d/tomcat7 restart
```

4.2 Resttool

Ausführbare JAR Datei erstellen:

```
#!/bin/bash

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
mvn clean package
mvn assembly:assembly
```

Generieren der Testbenutzer:

```
#!/bin/bash

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
/usr/lib/jvm/java-7-openjdk-amd64/bin/java -jar target/resttool-1.0-SNAPSHOT-jar-with-dependencies.jar -g user 500 -o user.csv
```

Import des eAssessment Kurses:

```
#!/bin/bash

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
/usr/lib/jvm/java-7-openjdk-amd64/bin/java -jar target/resttool-1.0-SNAPSHOT-jar-with-dependencies.jar -i eAssessment_Course.zip -o course.csv
```

Da das Resttool nicht automatisiert die Benutzungsbedingungen annehmen kann, müssen wir noch in der Administration Site dies mittels der Bulkaufgabe machen. Dabei stellen wir die Standardsprache zudem auf Deutsch. Ausserdem muss in JMeter im *open eAssessment test* Sampler die Repository Entry ID auf den importierten Kurs angepasst werden, welche Sie aus course.csv entnehmen können.

```
#!/bin/bash

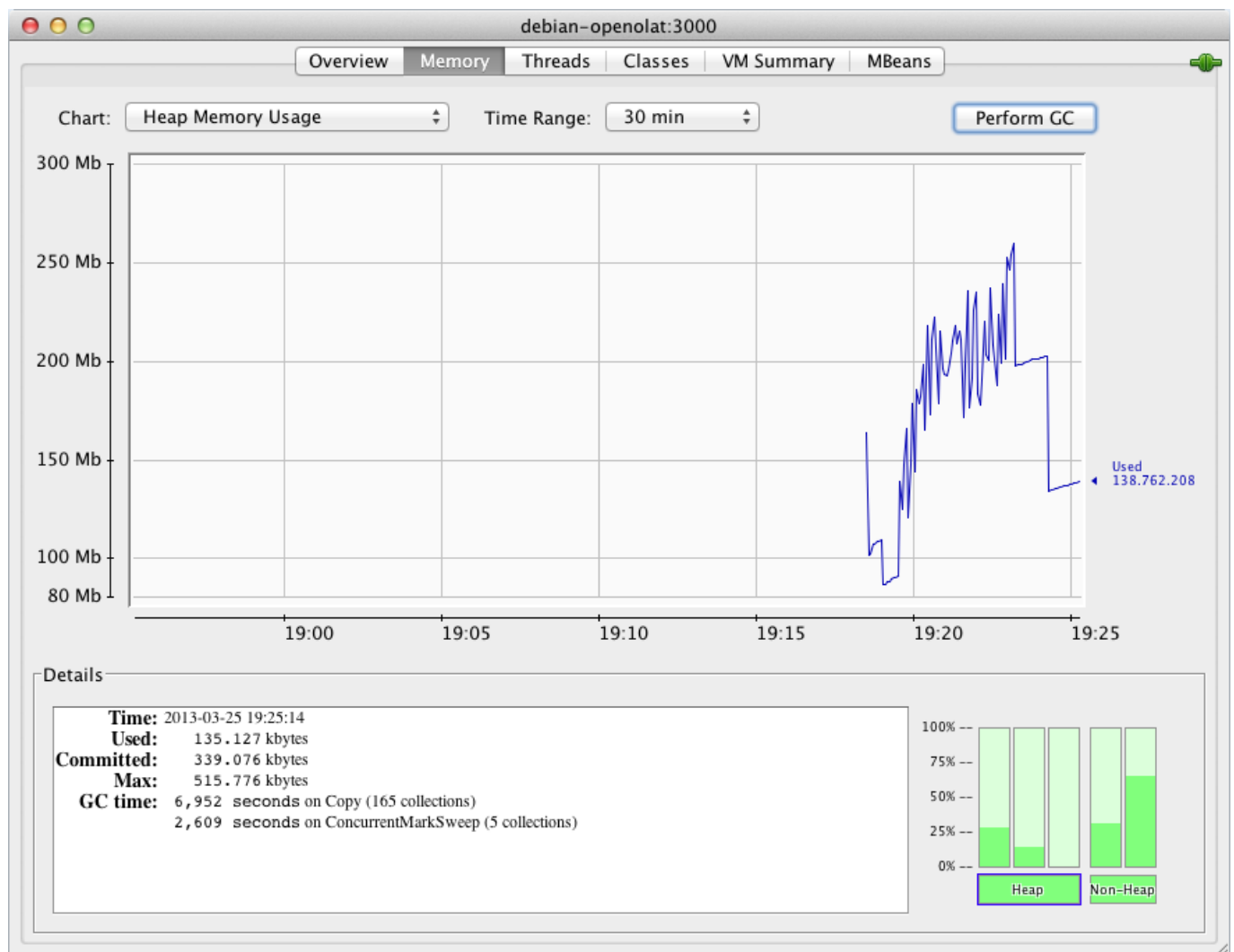
# zweite Spalte ausgeben - Benutzernamen auflisten
awk -F ',' '{print $2}' user.csv
```

Kondition	Ergebnis
ein Durchlauf, alle Threads innerhalb 50 Sekunden gestartet.	Erfolg 100%, Dauer 3min 30s
Auslastung RAM vor dem Test	85MB
Auslastung RAM nach dem Test	134MB
Spitzenauslastung RAM	260MB

Tabelle 4.2: Auslastung mit 50 Benutzer

4.3 Test 1: 50 Benutzer

Ich erwarte, dass der Test erfolgreich verläuft.



Bildschirmfoto Test 1 - JConsole Heap Memory Usage

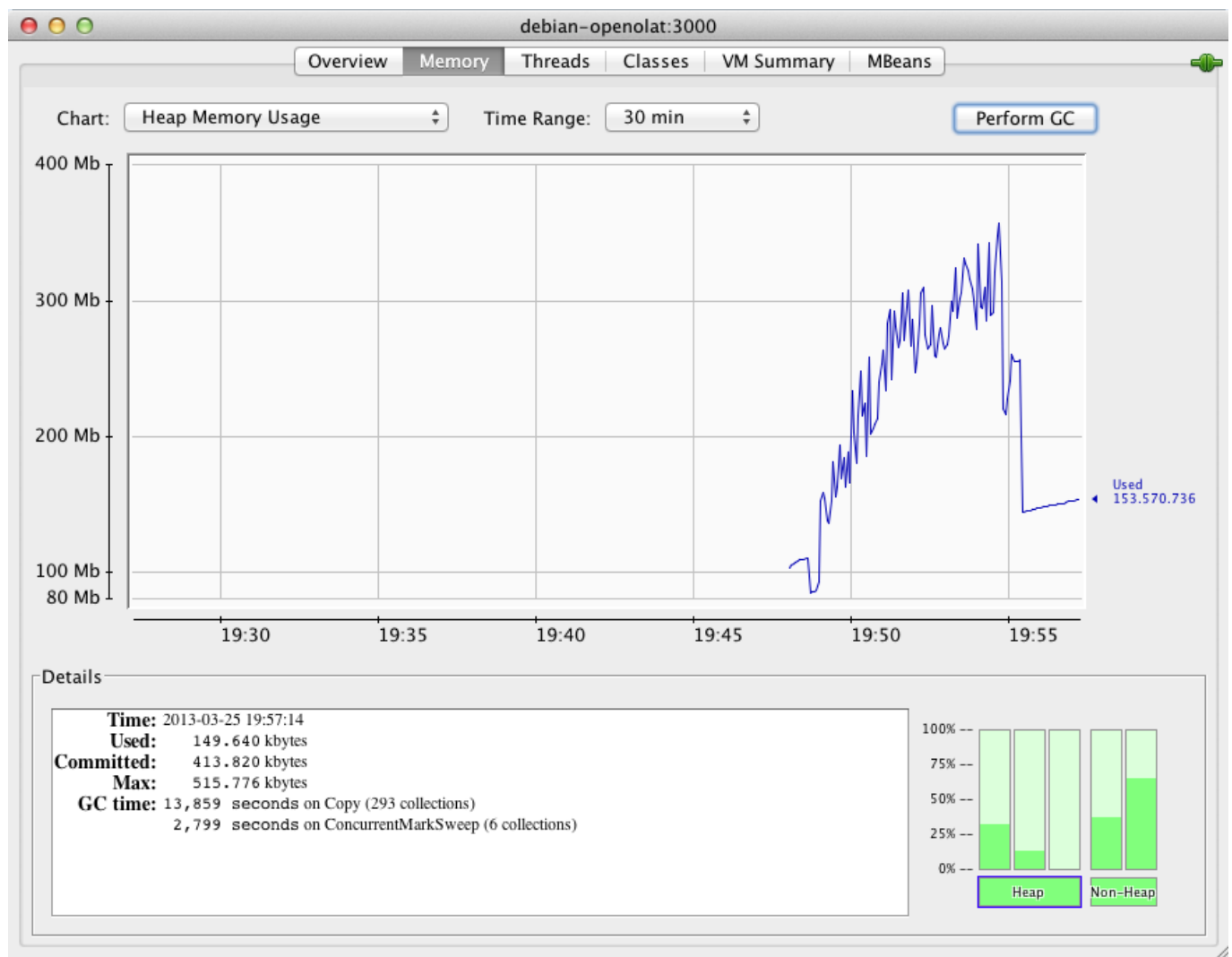
Der Test lief ohne Fehler durch. Jedoch ist der verfügbare Speicher nach dem Test geringer, diesem Phänomen sollten wir nachgehen - eventuel auf ein wachsendes Logfile zurückzuführen. Nach der Durchführung erkannte ich, dass der Speicher auch für Tests mit 100 Benutzer durchlaufen wird.

4.4 Test 2: 100 Benutzer

Da die Threads sich einholen können, erwarte ich Engpässe.

Kondition	Ergebnis
ein Durchlauf, alle Threads innerhalb 100 Sekunden gestartet.	Erfolg 99%, Dauer 6min
Auslastung RAM vor dem Test	86MB
Auslastung RAM nach dem Test	144MB
Spitzenauslastung RAM	358MB

Tabelle 4.3: Auslastung mit 100 Benutzer



Bildschirmfoto Test 2 - JConsole Heap Memory Usage

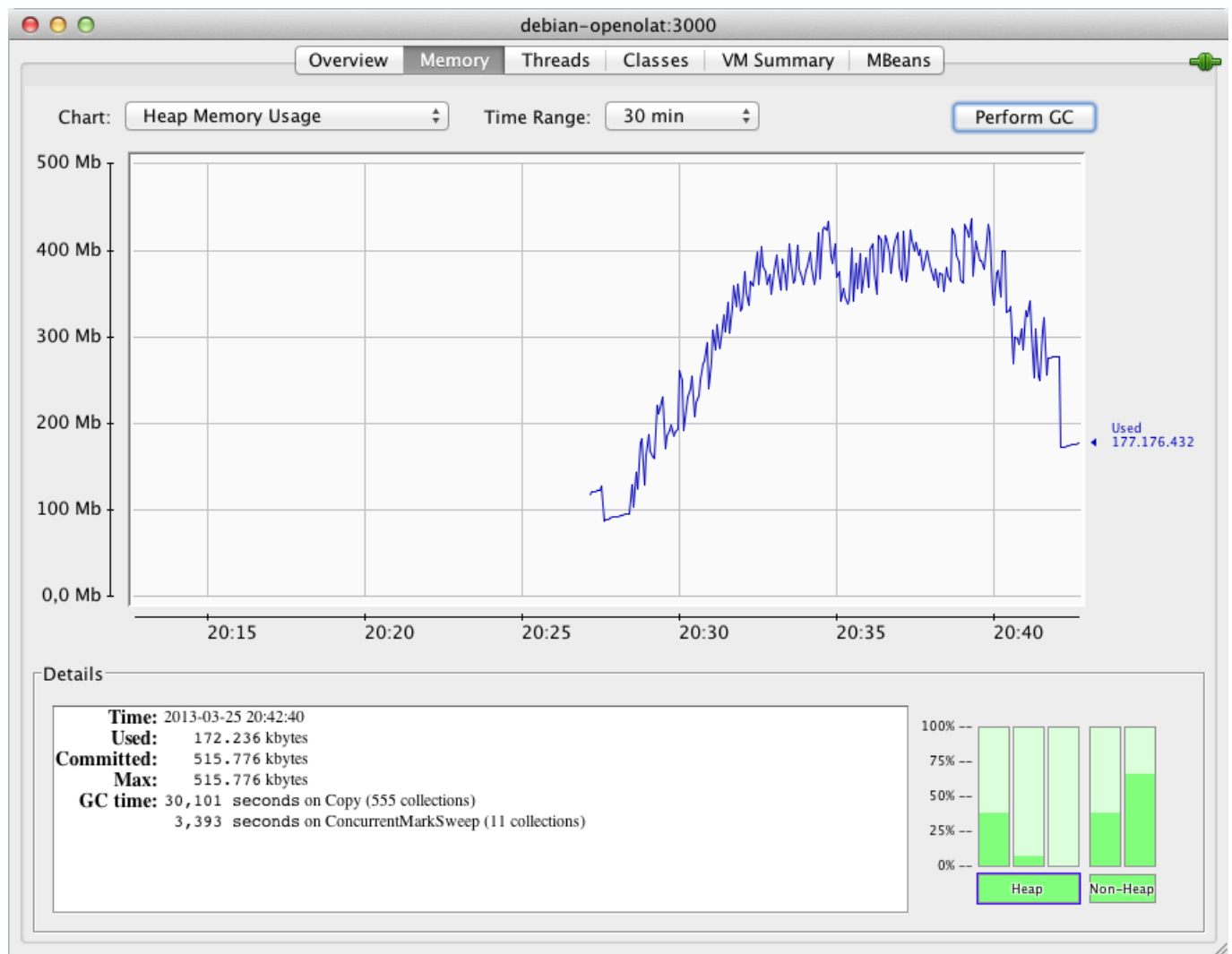
Der Fehler ist kritisch, weil der eAssessment Test nicht abgeschickt werden konnte. Er hat aber nichts mit der Speicherauslastung zu tun, sondern mit der Synchronisation von gleichzeitigen Abläufen.

4.5 Test 3: 200 Benutzer

Meine Meinung ist, dass der Test zu lange dauert und die Verzögerung nicht einer realen Belastung entspricht.

Kondition	Ergebnis
ein Durchlauf, alle Threads innerhalb 200 Sekunden gestartet.	Erfolg 98%, Dauer 14min
Auslastung RAM vor dem Test	87MB
Auslastung RAM nach dem Test	172MB
Spitzenauslastung RAM	437MB

Tabelle 4.4: Auslastung mit 200 Benutzer



Bildschirmfoto Test 3 - JConsole Heap Memory Usage

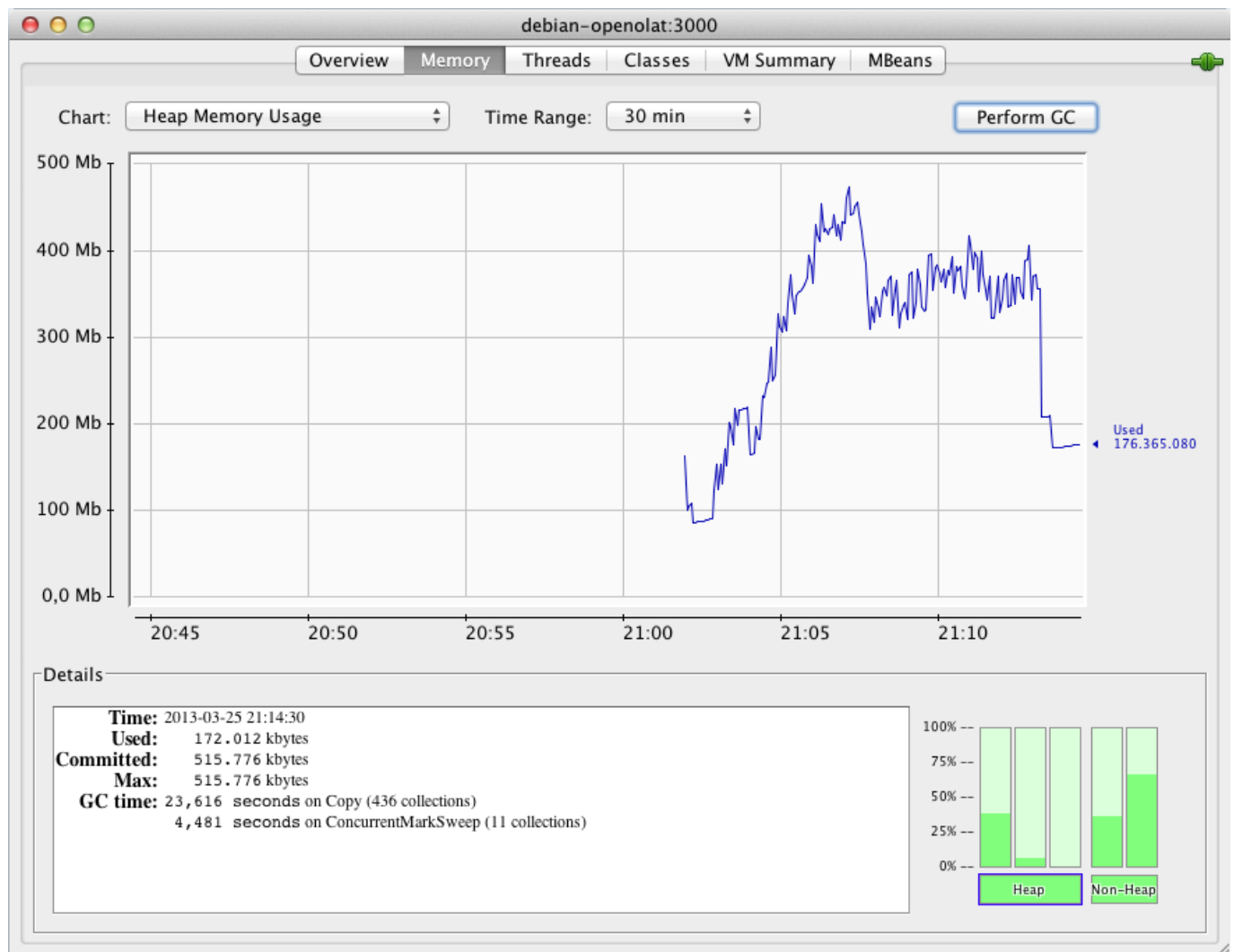
Es traten viermal synchronisations Probleme auf, die selben wie oben.

4.6 Test 4: 200 Benutzer à 50 Sekunden

Die Auslastung des Systems betreffend Geschwindigkeit wird wahrscheinlich erreicht.

Kondition	Ergebnis
ein Durchlauf, alle Threads innerhalb 50 Sekunden gestartet.	Erfolg 72%, Dauer 10min
Auslastung RAM vor dem Test	87MB
Auslastung RAM nach dem Test	172MB
Spitzenauslastung RAM	437MB

Tabelle 4.5: Auslastung mit 200 Benutzer à 50 Sekunden



Bildschirmfoto Test 4 - JConsole Heap Memory Usage

Es traten dreimal Synchronisations Probleme auf, die selben wie oben. Zudem konnten 52 Benutzer den Test erst gar nicht starten. Dieses Verhalten ist wahrscheinlich auf Leistungsprobleme der CPU zurückzuführen, da die Exception `org.olat.logging.DBRuntimeExce` geworfen wird, aufgrund einer Datenbank Timeout. Das heisst, ich bin an die Grenzen des Computers gekommen und nicht an die von OpenOLAT LMS.

Kapitel 5

Reflexion

5.1 Verlauf

Aufgrund meiner Programmiererfahrung konnte ich schnell den Einstieg in JMeter finden. Jedoch war ich zuerst fixiert auf bekannte Strukturen und Geltungsbereiche. Worauf ich mich trotzdem gut einstellen konnte.

Das XPath Problem musste ich auf die harte Tour lernen, wie diese Ausdrücke realisiert wurden. Ich habe JMeter praktisch fertig gepatcht, als ich in der letzte Zeile bemerkte, dass die Anzahl in generierte Referenzen gespeichert wird. Ich verlor dabei etwa drei bis vier Stunden.

Meine Arbeit ist auf jedenfall ein Erfolg, da ich alles Umsetzen konnte. Zudem läuft der JMeter Loadtest unabhängig vom eAssessment Test den ich während des Programmieren und Evaluieren verwendet habe.

5.2 Arbeitsmethodik

Ich verwendete Docbook XML zur Formatierung der gesamten IPA. Recherchen führten mich zu hilfreichen Werkzeuge, wie z.B. das javadoc Doclet dbdoclet, mit dem es sich ganz einfach aus Java Kommentare eine API Referenz generieren lässt. Mittels dblatex generierte ich PDF Dateien.

Keinen WYSIWYG Editor zu verwenden gab mir Kontrolle über den Aufbau des Dokumentes und ersparte mir Mehraufwand. Dies nicht zuletzt, da der mich begleitende Chefexperte mir Inputs betreffend Wegleitung gab. Ich konnte mit Verschieben einer Zeile das ganze Journal ins erste Kapitel übernehmen. Ich würde diese Technologie wieder einsetzen, da ich nur Vorteile sehe.

5.3 Kommunikation

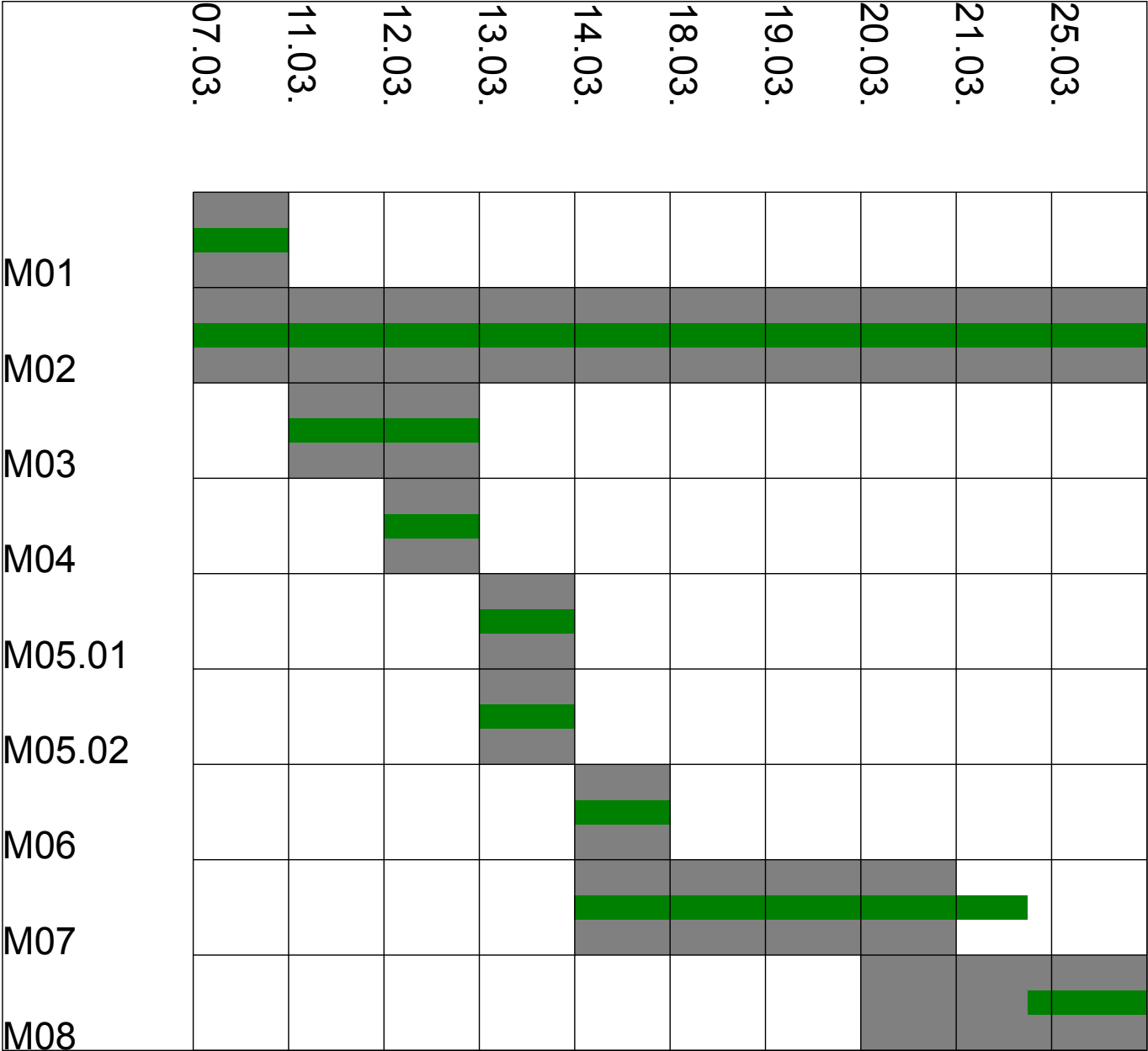
Ich hielt Rücksprache mit dem Hauptentwickler bezüglich Auswertung des JMeter Tests für kritische Programmparameter anzupassen. Während des Verlaufs der Arbeit informierte ich meinen Fachvorgesetzten über den Stand der Arbeit.

Anhang A

Zeitraffer - Status der Arbeiten

Wegen halbtägigem Stromausfall und Fehlkalkulation von 2 Stunden hatte ich für M07 länger als geplant. Die Fehleinschätzung ist auf Komplikationen betreffend Verständnis des XPath Konzeptes zurückzuführen.

Nicht ersichtlich ist der halbe Tag Aufschub, wegen dem Stromausfall. Ich habe es nicht mehr angepasst, da der letzte Task Pufferzeit beinhaltet.



Reflexion Zeitplanung im Raster.

Anhang B

Glossar

B.1 Glossar

In diesem Glossar werden Abkürzungen sowie Fachbegriffe ausgeführt.

A

Asynchronous JavaScript and XML (AJAX)

AJAX ist das Feature eines Browsers, das JavaScript ermöglicht zu einem beliebigen Zeitpunkt Daten an einen Webserver zu schicken und die Antwort abzuwarten, um dann z.B. das DOM zu bearbeiten. Zur Kommunikation wird entweder JSON oder XML eingesetzt.

C

Comma Separated Values (CSV)

CSV Dateien zeichnen sich durch ihre einfache und strikte Semantik aus. Sie werden als eine Tabelle interpretiert, wobei ein Feldseparator Zeichen die Datei in Spalten gliedert. Felder mit Leerzeichen werden mittels Hochkomata begrenzt. Die Zeilen der Datei entsprechen der Reihe einer Tabelle.

G

Graphical User Interface (GUI)

Das GUI ist eine Benutzeroberfläche eines Programmes die fensterbasiert aufgebaut ist. Typische Bedienelemente sind Knöpfe, Schieber und Eingabefelder. Im Kontrast dazu gibt es Konsolenanwendungen die eingabebasiert bedient wird.

H

Hyper Text Markup Language (HTML)

HTML ist eine seitenbeschreibungs Sprache, die Layout-, Formatier- und Meta-Elemente definiert. Diese Elemente nennt man Tags und sind vorgegeben. Tags werden mit Spitzen Klammern begrenzt, wobei es Standalone- und Schliessende-Tags gibt. Es ist ein Dateiformat das hauptsächlich seine Anwendung im World Wide Web findet und gilt als Standard.

Hyper Text Transport Protocol (HTTP)

HTTP definiert die Kommunikation zwischen dem Client und Webserver. Es befindet sich auf der OSI-Modell Schicht fünf bis sieben und ist somit ein Anwendungsprotokol. Der wohl verbreiteste Webserver wird von der Apache Software Foundation unterhalten.

J

Java Runtime Environment (JRE)

Zur JRE gehören die Laufzeitumgebung und die Virtuelle Maschine in der assemblierter Java Code abgearbeitet wird. Java Programme lassen sich auf der entsprechende JRE plattformunabhängig ausführen.

L

Learning Management System (LMS)

Ein LMS ist eine Plattform auf der sich typischerweise Lerninhalte erstellen lässt, um sie zu präsentieren. eAssessment Tests ist eines der interaktiven Lernmöglichkeit, die ein LMS implementieren kann.

O

Open Source Software (OSS)

Unter dem Begriff OSS versteht man Software deren Quellcode öffentlich zugänglich ist. Freie Software wird oftmals fälschlicher Weise als Open Source bezeichnet, gilt aber klar zu unterscheiden bezüglich Lizenz. OSS hat Einschränkungen bezüglich Modifizieren und Verteilen.

S

Structured Query Language (SQL)

SQL ist eine Scriptsprache für Datenbank Management Systeme. Mittels SQL lassen sich Datenbanken und Tabellen erstellen oder verändern.

W

Web Archive (WAR)

WAR Dateien enthalten ZIP-komprimierte Dateien. Es sind JAR Dateien die einem vorgegebenen Aufbau folgen. Typisch dafür ist die `Manifest` Datei im Verzeichnis `META-INF` und der Web Applikations Ordner `WEB-INF`.

Anhang C

Literatur

C.1 Literatur

- [1] [Debian Reference](http://www.debian.org). Software in the Public Interest, Inc.. Stand 25.03.2013 <http://www.debian.org>.
- [2] [Tomcat 7](http://tomcat.apache.org). Apache Software Foundation. Stand 25.03.2013 <http://tomcat.apache.org>.
- [3] [MySQL 5.5 Reference Manual](http://www.mysql.com). Oracle. Stand 25.03.2013 <http://www.mysql.com>.
- [4] [OpenOLAT 8 User Manual](http://www.openolat.org). frentix GmbH. Stand 25.03.2013 <http://www.openolat.org>.
- [5] [openolat LMS - RESTful API](http://www.openolat.org). frentix GmbH. Stand 25.03.2013 <http://www.openolat.org>.
- [6] [Maven - The Definitive Guide](http://www.oreilly.com). O'Reilly. September 2008 <http://www.oreilly.com> 978-0-596-51733-5.
- [7] [Java Buch](http://www.addison-wesley.de). Addison-Wesley. Stand 11.03.2012 <http://www.addison-wesley.de> 978-3-8273-2751-2.
- [8] [RESTful Java with JAX-RS](http://www.oreilly.com). O'Reilly. Dezember 2009 <http://www.oreilly.com> 978-0-596-15804-0.
- [9] [XPath Specification](http://www.w3c.org). W3 Consortium. Stand 25.03.2012 <http://www.w3c.org>.
- [10] [JMeter - User's Manual](http://jmeter.apache.org). Apache Software Foundation. Stand 11.03.2012 <http://jmeter.apache.org>.
- [11] [JConsole - Java SE Monitoring and Management Guide](http://www.oracle.com). Oracle. Stand 25.03.2012 <http://www.oracle.com>.