Hacking-Lab Mobile Event App | Backend Installation Guide

Oussama Zgheb

7. Juni 2015

Datum	Version	Änderung	Autor
14.04.15	1.0	Initial	Oussama Zgheb
16.04.15	1.1	Weitere Settings erfasst	Oussama Zgheb
29.05.15	1.2	Facebook & Twitter	Oussama Zgheb
01.06.15	1.3	Installation, Unterhalt	Oussama Zgheb

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument soll einer mit der Materie vertrauten Person eine Hilfestellung zur Konfiguration und Installation des HLM-NG Backend geben.

2 Voraussetzungen

Die Software wurde auf unter Ubuntu 12.04 LTS, Lubuntu 14.10 & 15.04 sowie Debian 7 entwickelt und getestet , weitere Betriebssysteme sind höchst wahrscheinlich auch möglich einzusetzen.

Folgende Software wird auf dem Zielgerät benötigt:

- Tomcat 7
- MySQL 5.6
- Eclipse EE (Kepler oder neuer)

2.1 Google Cloud Messaging

Google Cloud Messaging [1] wird eingesetzt, um Push Meldungen auf die Mobile Apps zu verteilen. Um den Dienst benutzen zu können, braucht man ein Google Konto [2] sowie einen ApiKey. Um den ApiKey zu erhalten muss man unter `https://console.developers.google.com/project` ein neues Project erstellen. Nun findet man unter »APIs & Auth« den Dienst »Google Cloud Messaging for Android«, dieser mussa aktiviert werden. Unter »APIs & Auth -> Credentials« kann ein »Key for server applications« (API Key) erstellt werden. Diese Daten werden später benötigt.

2.2 Facebook

2.2.1 App erstellen

Um auf eine Facebook Seite zu Posten wird eine App benötigt, diese kann unter `https://developers.facebook.com/` erstellt werden. Zuerst muss jedoch ein Facebook Konto bestehen sowie auf Facebook angemeldet sein. Wichtig: Der App Typ muss dabei »Website« sein und die Site URL richtig gesetzt sein, z.B. »https://fix.confoxy.com/hlmng/frontend/«. Nun findet man im Dashboard der App folgende Daten: »App ID« und »App Secret«.

2.2.2 Konfiguration

Nun müssen die Parameter der Facebook4J Library [3] hinterlegt werden. Dies geschieht in der Datei facebook4j.properties im Rootverzeichnis »src«. Die Datei sollte also wie folgt aussehen:

```
oauth.appId=****************
oauth.appSecret=********************************
oauth.permissions=publish_actions, manage_pages, publish_pages
```

Diese Tokens laufen nie ab, es Bedarf also keiner nachträglichen Änderungen mehr. [4]. Nach Bedarf kann auch »debug=true« hinzugefügt werden, diese liefert wertvolle Informationen auf System.out.

2.3 Twitter

2.3.1 App erstellen

Um die Twitter API zu benutzen muss man unter `https://apps.twitter.com/` ein App erstellen. Dafür wird ein Twitter Account benötigt, bei bedarf sollte dieser zuerst erstellt und angemeldet sein. Wichtig ist, dass beim erstellen

[1] `https://developer.android.com/google/gcm/gs.html`
[2] `https://accounts.google.com/signup`
[3] `http://facebook4j.org/`
[4] `https://developers.facebook.com/docs/facebook-login/access-tokens`

der App das Feld »Website« mit der richtigen URL ausgefüllt wird, also z.B. »https://fix.confoxy.com/hlmng/frontend/«. Nach dem erstellen der App sollte man nun unter »Keys and Access Tokens« die benötigten Daten finden. Folgende Parameter benötigt, »Consumer Key«, »Consumer Secret«, »Access Token«, »Access Token Secret«.

2.3.2 Konfiguration

Nun müssen die Parameter der Twitter4J Library [5] hinterlegt werden. Dies geschieht in der Datei twitter4j.properties im Rootverzeichnis »src«. Die Datei sollte also wie folgt aussehen:

```
oauth.consumerKey=**********************
oauth.consumerSecret=**************************************************
oauth.accessToken=******************************************
oauth.accessTokenSecret=******************************************
```

Diese Tokens laufen nie ab, es Bedarf also keiner nachträglichen Änderungen mehr, ausser das App wird durch den Twitter Account entfernt. [6]. Nach Bedarf kann auch »debug=true« hinzugefügt werden, diese liefert wertvolle Informationen auf System.out.

2.4 Tomcat

Um sicherzustellen, dass genügend Ressourcen vorhanden sind, wird empfohlen den Java Heap Space etwas grösser als die Default Einstellung festzulegen. Dies geschieht im File »/etc/default/tomcat7«. Dort sollte die Zeile »JAVA_OPTS« wie folgt aussehen:

```
JAVA_OPTS="-Djava.awt.headless=true -Xmx768m -Xms384m -XX:+UseConcMarkSweepGC"
```

Xmx ist dabei die maximale Heap Size, Xms die initiale Grösse. Bei grosser Benutzeranzahl muss die maximale Heap Size eventuell erhöht werden.

3 Installation Server

3.1 Ordner erstellen

Das Backend muss Daten auf dem Filesystem ablegen können. Empfohlen ist, die folgenden Ordner unter **»/var/lib/hlmng«** abzulegen. Es werden die Unterordner »logs«, »media«, und »qr« benötigt. Wichtig dabei ist, dass der User tomcat7 owner ist. Dies kann wie folgt übernommen werden:

```
chown tomcat7 /var/lib/hlmng/*
```

3.2 HLMNGSettings

Unter **»src/settings/HLMNGSettings.java«** können die Parameter für das Backend vorgenommen werden. Nachfolgend werden alle erklärt,

3.2.1 Pfade

- jdbcPath (String) = Der JDBC Pfad zur Datenbank
  z.B. »jdbc:mysql://127.0.0.1/hlmng«, wobei hlmng der Datenbank Schema Name ist
- pubURL = Der Pfad unter welchem der Public Rest API Teil sein soll
  z.B. »/pub«
- admURL = Der Pfad unter welchem der Admin Rest API Teil sein soll
  z.B. »/adm«

[5] http://twitter4j.org/
[6] https://dev.twitter.com/oauth/overview/faq

- qrFileRootDir (String) = Der Pfad unter welchem alle die gerenderten Qr Codes abgelegt werden z.B. »/var/lib/hlmng/qr/«
- mediaFileRootDir (String)= Der Pfad unter welchem alle heraufgeladenen Dateien abgelegt werden z.B. »/var/lib/hlmng/media«
- logFileRootDir (String) = Der Pfad unter welchem die Logfiles abgelegt werden (falls aktiv) z.B. »/var/lib/hlmng/logs«
- restAppPath (String) = Der Pfad unter welchem die Schnittstelle erreichbar sein soll z.B. »https://fix.confoxy.com/hlmng/rest«
- gcmURL (String) = Die URL der Google Cloud Messaging API um Nachrichten zu senden z.B: »https://android.googleapis.com/gcm/send«
- Der Subpfad zur Rest API wird im File »WebContent/web.xml« festgelegt:

```
<servlet-mapping>
        <servlet-name>
                Jersey REST Service
        </servlet-name>
        <url-pattern>
                /rest/*       <---- */
        </url-pattern>
</servlet-mapping>
```

3.2.2 Login Daten

- jdbcUser (String) = Der JDBC User für die Datenbank
- jdbcPassword (String) = Das JDBC Passwort für den User
- apiKey (String) = Der vom Google erhaltene Key für ihre API's

3.2.3 Facebook

- facebookAppId (String) = Die Facebook App ID
- facebookPageId (String) = Die Facebook Page ID auf welche gepostet werden soll

3.2.4 Weiteres

- cacheTime (int) = Cache Max-Age in Sekunden
- qrCodeWidth , qrCodeHeight (int)= Die Dimension der gerenderten Qr Codes in Pixel
- maxMediaImageSizeMB (Double) = Die maximale Dateigrösse für die Datei uploads in Megabyte
- logSysErr (boolean) = Falls Wahr wird jeglicher Log Output auf System.Error umgeleitet anstatt direkt in das Logfile geschrieben zu werden
- selectLimit (int) = Die Anzahl von Elementen die bei limitieren Queries geladen werden sollen (z.B. bei /newest calls)
- maxActionsAllowed (int) = Die Anzahl Aktionen (siehe API Dokumentation -> Spam Schutz) welche in »actionGraceTime« durchgeführt werden dürfen
- actionGraceTime (int) = Die Aktionen Zeitfenster (in ms)
- Angestrengtheit (int) = Die Länge der QR Code Payload in Bit
- mediaUploadThumbnailPixel (int) = Die maximale Höhe oder Länge eines Thumbnails

3.3 MySQL

3.3.1 Konfiguration

Auf der MySQL Server Instanz muss ein Schema mit dem Namen »hlmng« erstellt werden (andere Namen sind möglich, diese müssen dann auch so in die Konfiguration übernommen werden). In dieses Schema müssen alle sich unter **»src/doc/6_SQL_Creates«** befindende SQL Dateien importiert werden. Dies ist möglich mit dem »source« Befehl oder einer GUI wie MySQL Workbench [7]. Nun sollte ein Benutzer mit den benötigten Berechtigungen (delete,insert,select,update) angelegt werden, dies sollte etwa wie folgt aussehen:

```
CREATE USER 'user'@'localhost' IDENTIFIED BY '************';
GRANT DELETE,INSERT,SELECT,UPDATE ON hlmng.* TO 'user'@'localhost';
```

3.3.2 Verbindungsdaten

Unter **»hlmmng/WebContent/META-INF/context.xml«** werden die Informationen für die MySQL Verbindung festgelegt, siehe »url« sowie »username« und »password«.

```
<Resource name="jdbc/hlmng" auth="Container" type="javax.sql.DataSource"
driverClassName="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/hlmng?autoReconnect=true"
username="user" password="password" />
```

4 Installation Software

4.1 Kompilation

Die neuste Version kann per GitHub [8] bezogen werden.

4.1.1 Eclipse

Unter Eclipse kann man das Projekt wie folgt exportieren:
Project Explorer -> Rechtsklick -> Export -> War File

4.1.2 Commandline

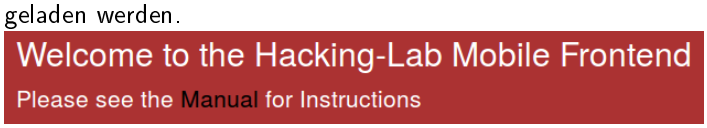
```
cd /path/to/project
jar cvf hlmng.war *
```

4.2 Installation

Das erstellte WAR File muss in den »WebApps« Ordner von Tomcat kopiert werden. Dieser befindet sich bei Debian unter »/var/lib/tomcat7/webapps/«. Die Anwendung sollte nun automatisch deployed werden und sogleich erreichbar sein.

4.2.1 Überprüfung

Falls alles erfolgreich installiert wurde, sollte unter `https://localhost:8443/hlmng/frontend/` das Frontend geladen werden.



Als Test sollte erfolgreich ein »Speaker« erstellt werden können. Unter `https://localhost:8443/hlmng/rest/adm/speaker` sollte der soeben erstellte »Speaker« nun als JSON angezeigt werden. Zudem sollte das Log frei von Warnings oder Errors sein.

[7] `http://www.mysql.de/products/workbench/`
[8] `https://github.com/ozzi-/hlmng`

4.2.2 Common Mistakes

- Falsch Verbindungsdaten in »context.xml«

```
com.mysql.jdbc.exceptions.jdbc4.MySQLNonTransientConnectionException:
Could not create connection to database server. Attempted reconnect 3 times.

oder

Caused by: java.sql.SQLException: Access denied for user 'hlmdng_db' . .
```

- »Seite konnte nicht gefunden werden«
  Sicherstellen, dass die Pfade richtig konfiguriert sind und die Applikation von Tomcat geladen wurde.
- Nicht das Log Konsultieren
  Viele Fehler können durch das konsultieren des Log lokalisiert werden.
- Falsche Systemzeit
  Falls eine falsche / abweichende Systemzeit auf dem Server (sowie auf dem System des Operator Browsers) kann es zu Fehlern bei Countdown und Timer anzeigen geben. Auch Twitter sowie Facebook können bei abweichender Systemzeit die Posts verweigern (respektive deren Oauth Implementiertung).

5 Unterhalt

Folgende Unterhaltsarbeiten können im längerfristigen Betrieb auftreten:

- Expiry der API Token
  Neue Token generieren und in der Applikation einfügen.
- Festplattenspeicher
  QR Codes auf dem Dateisystem können gelöscht werden, bei Bedarf werden diese neu angelegt.
  Alte Log Dateien können migriert oder gelöscht werden.
- Arbeitsspeicher
  Da kein langfristiger Testlauf mit mehren hunderten Benutzern im Rahmen der Entwicklung der Bachelorarbeit simuliert werden konnte, sollte beim ersten produktiven Einsatz die Arbeitsspeicher Auslastung überwacht werden, z.B. mit JStat [9].
  - Datenbank
    Der übliche Unterhalt der Datenbank sollte gewährleistet sein.
- Libraries
  Es werden einige Libraries verwendet, diese sollten periodisch oder bei bekannten Sicherheitslücken aktualisiert werden und das Projekt neu kompiliert, getestet und deployed werden.
- ui-Router
  Da das Modul ui-Router noch in Entwicklung ist, kann es durchaus sein, dass es noch grössere Änderungen geben wird. Diese müssten bei einem Update des Moduls eventuell nachgeführt werden.

[9] https://docs.oracle.com/javase/7/docs/technotes/tools/share/jstat.html

6 Fussnoten Index