Praxis der Softwareentwicklung: Visualizing Trends. Was verrät uns Twitter?

Implementierung

Maximilian Awiszus Holger Ebhart Lidia Grigoriev
Paul Jungeblut Philipp Kern Matthias Schimek

WS 2014/15

Inhaltsverzeichnis

1 Einleitung

Diese Dokument gibt einen Überblick über den Implementierungsprozess unseres Projekts. Der Hauptbestandteil ist die Dokumentation von Änderungen, die an dem in der vorherigen Projektphase erstellten Entwurf vorgenommen wurden. Diese Änderungen werden im nächsten Kapitel aufgeführt und erläutert. Das zweite Kapitel dieses Dokuments listet Komponententest auf, die schon während der Implementierungsphase durchgeführt wurden. Allerdings ist deren Anzahl begrenzt.

2 Änderungen gegenüber dem Entwurf

In diesem Abschnitt werden die Änderungen gegenüber dem Entwurf beschrieben.

2.1 Änderungen an der Datenbank

2.1.1 (mysql)

Auch in diesem Paket gab es nur geringfügige Änderungen. Es wurden noch einzelne weitere Methoden hinzugefügt, um Daten unter einem speziellen Gesichtpunkt bei der Datenbank anzufragen. Weitere kleinere Änderungen sind im Folgenden aufgelistet:

DBConnection executeStatmentUpdate(Statement) führt ein 'Update' in der DB durch.

closeResult(Result) schließt ein mysql-result

closeStatement(Statement) schließt ein mysql-statement

result

Hier wurden nur einige Getter- und Setter-Methoden für die einzelnen Klassen bzw. ihre Attribute hinzugefügt, bzw. deren interne Struktur (z.B. Liste statt Array) leicht modifiziert.

2.2 Änderungen am Crawler

Am Crawler musste in Bezug auf die im Entwurf festgelegte Programmlogik ein großer Teil angepasst werden. Diese Umstellung des Entwurfs wurden durch die im ersten Entwurf zu ineffiziente Lokalisierung der Tweets, Retweets und Accounts nötig. Der ursprüngliche Entwurf sieht vor, dass die von Twitter gelieferten Statusobjekte (Tweets,Retweets) in mehreren Warteschlangen einsortiert werden und diese von einer Instanz der Klasse 'Locator' lokalisiert werden. Der 'Locator' ruft dazu einen Webservice auf, der im Erfolgsfall den Ländercode des entsprechenden Statusobjekts zurücklieferte. Da ein Aufruf des Webservices deutlich länger als eine eine Sekunde benötigt, und auch das Aufrufen des Webservice von mehreren parallel arbeitenden Threads nur bedingt schneller Ergebnisse liefert, musste hier der Entwurf überarbeitet werden. Eine wesentliche Änderung besteht darin, dass die Ergebnisse der Lokalisierung gespeichert werden. Dazu wird die Ortinformation des Statusobjekts und der vom Webservice zugeordnete Ländercode in der einer Hashtabelle gespeichert und der Inhalt dieser Hashtabelle regelmäßig in die Datenbank geschrieben. So können schon einmal abgefragte Ortsinformationen ohne den zeitaufwendigen Umweg über den Webdienst lokalisiert werden. Weiterhin sieht die

jetzige Implementierung vor, dass die vom Twitter-Stream gelesenen Statusojekte in eine Warteschlange geschrieben werde, aus welcher sieh parallel von mehreren Worker-Threads entnommen werden. Diese prüfen, ob die Ortsinformation des Statusobjekts schon in der Hashtabelle vorhanden ist. Im Erfolgsfall wird das Statusobjekt sofort in die Datenbank geschrieben. Andernfalls wierd es in eine Warteschlange weitergereicht, aus welchem mehrere Lokalisierer-Threads die Objekte entnehmen und dem Webservice zur Lokalisierung senden. Das folgende Sequenzdiagramm verdeutlicht noch einmal den Ablauf des Vorgangs:

Neben dem 'Crawler' wurde auch der 'Locator' bzw. das Paket 'locate' etwas umstrukturiert, er besteht nun aus zwei (Haupt-)Klassen, welche einmal die Anfragen mittels Hashtabelle und Anfragen an den Webservice kapseln, sowie zwei weiteren (Hilfs-)Klassen. Diese sind im folgenden Klassendiagramm dargestellt.

2.3 Änderungen am Kategorisierer

2.4 Änderungen an der GUI

An der GUI wurden einige Änderungen zum Entwurf vorgenommen, dies lag u.A. an einer Vielzahl von Details, welche während der Entwurfsphase noch nicht bekannt oder bewusst auf die konkrete Implementierung verschoben wurden. Die Grobstruktur des Entwurfs blieb jedoch erhalten, wie das folgende Paketdiagramm (2.5) zeigt. Eine wichtige Änderung in Bezug auf das Gesamtpaket 'gui' ist, dass es sich bei den 'View' - Klassen, welche die Darstellung der grafischen Komponenten beinhalten, um XML Dateien handelt. Dies war im Entwurf so nicht vermerkt, allerdings implizit angedacht. Die im Entwurf beabsichtigten Idee der Trennung von Darstellung und Programmlogik wird dadurch noch stärker erfüllt, da die XML-Dateien keinerlei Programmlogik enthalten. Desweiteren existiert in jedem Unterpaket von 'gui' nicht nur eine XML-Datei mit Darstellungslogik, sondern u.U. mehrere, da so teilweise größere grafische Komponenten in einzelne XML-Dateien gekapselt werden können.

Allgemein gilt, dass im folgenden nur 'wichtige' Änderungen dokumentiert werden.

2.4.1 Änderungen im Paket gui

Hier sollen die Änderungen beschrieben werden, die in Klassen vorgenommen wurden, welche unmittelbar im Paket 'gui' angesiedelt sind.

GUIController In der Hauptkontroller-Klasse sind einige Methoden hinzugekommen. Größtenteils waren diese im Entwurf schon angedacht nur noch nicht hinreichend konkret geplant worden, die Änderungen im GUIController werden ausführlicher als für den Rest der (Unter-)Pakete besprochen, da es sich um die zentrale Klasse der GUI handelt.

getAccount(int) liefert Account von DB zurück.

close() beendet Programm.

addUserToWatch(twitter4j.User, int) fügt Twitter-Account zur Datenbank hinzu.

getCategory(int) liefert Kategorie von Datenbank
getCategory(String) liefert Kategorie von Datenbank (nach Zeichenkette im Namen)
getCategoryRoot(int[]) baut Baum aus Kategorie-Items auf
getSummedData() liefert alle Daten von Datenbank zu einer Suchanfrage
getDataByAccount() liefert Daten aufbereitet nach Account
getDisplayValueProperty() 'Berechnungsformel' liefert relativen Wert zum Einfärben der Karte
getLocations() liefert Orte von DB zurück
getMapDataillnformation() liefert Detailinformationen zu Orten, auf welche geklickt wird
getSelectedAccounts() liefert in Query ausgwählte Accounts
getSelectedCategories() lieft in Query ausgewählte Kategorien
getSelectedLocations() liefert in Query auswählte Orte
isConnected() indiziert ob Verbindung zu DB besteht
isReady()
main() startet Programm
setCategory(int, int) fügt Kategorie zu DB hinzu
setMapDetailInformation(MyDataEntry) setzt Detailinformationen zu Orten, auf welche geklickt wird
setSelectedAccount(int, boolean) fügt Account zu Query hinzu
setSelectedCategory(int, boolean) fügt Kategorie zu Query hinzu
setSelectedLocation(int, boolean) fügt Ort zu Query hinzu

SelectionHashList Diese neu hinzugekommene Klasse stellt eine Ergänzung zum Entwurf dar, welche aus Gründen der Effizienz gewählt wurde. Mit implemtiert eine Hashstruktur, über die mittels Getter- und Settermethoden entschieden werden kann, ob eine Kategorie oder ein Account für ein Query an die Datenbank ausgewählt ist, bzw. diese zu einem Query hinzuzufügen.

Labels Diese Klasse enthält zentral alle Beschriftungen, welche in der GUI verwendet werden.

RunnableParameter Neu hinzugekommene Klasse, die das Interface 'Runnable' so implementiert, dass dem Runnable ein Parameterübergeben werden kann.

InfoRunnable Zusammen mit der gerade geschriebenen Klasse, wird mit der neu hinzugekommenen InfoRunnable ein Thread gestartet, der Nachrichten wie 'Verbinde mit Datenbank ...', 'Lade Daten' für eine begrenzte Zeit anzeigt. Die folgende Grafik zeigt das Klassendiagramm des 'GUIControllers' zusammen mit verwandten Klassen. Allerdings sind die Klassen nicht mit all ihren Methoden und Attributen aufgeführt.

Im Folgenden werden die Änderungen in den Subpaketen von 'gui' beschrieben.

databaseOpts

In diesem Paket wurde, wie oben vermerkt, die XML-Datei für den grafischen Inhalt in 4 Separate XML Dateien aufgeteilt, um die Lesbarkeit und Übersichtlichkeit zu

erhöhen. Weiter wurden in 'DatabaseOptController' einige Methoden und innere Klassen hinzugefügt:

DatabaseOptController

intialize Diese Methode wird aufgerufen, wenn die GUI gestartet wird. Hier werden alle grafischen Elemente mit den jeweiligen EventHandler verknüpft.

MyAcctionEventHandler

MyLocEventHandler, MyAccEventHandler, MyCatEventHandler Behandelt alle Events die auf den verschiedenen grafischen Elementen aufgerufen werden.

Desweiteren wurden einige private Hilfmethoden hinzugefügt, die beispielsweise einen Baum aus Kategorien aufbauen.

table

Diese Paket blieb relativ unverändert, lediglich die Hilfsklasse 'InternAccount' ist aus Gründen der Effizienzsteigerung hinzugekommen.

standardMap

In diesem Paket ergaben sich einige Änderungen. Diese waren durch die relative Inkompatibilität des verwendeten Map-Frameworks 'unfolding' mit 'javafx'. Daher musste eine die Klasse 'StandardMapDialog' hinzugefügt werden:

StandardMapDialog Diese Klasse erbt von JFrame einem Java Swing-Typ, da nur in Java Swing das verwendete Map-Framework fehlerfrei dargestellt werden kann. Sie enthält unter anderem eine Referenz auf den 'GUIController' und auf eine Instanz von 'MyUnfoldingMap'.

update() diese Methode wird aktualisiert die auf der Karte dargestellten Daten. Die Daten werden vom 'GUIController' bezogen und an die eigentliche Karte in 'MyUnfoldingMap' weitergereicht.

timeSliderMap

Die im Paket 'standardMap' gemachten Änderungen waren hier im selben Maße nötig.

unfolding

In diesem Paket wurden nur geringfügige Änderungen durchgeführt.

MyUnfoldingMap mouseClicked(MouseEvent) Methode berechnet zu jedem Klick auf die Karte, das angeklickte Land und liefert Informationen zum jeweiligen Retweet-Verhalten.

Abbildung 2.1: Klassendiagramm Crawler

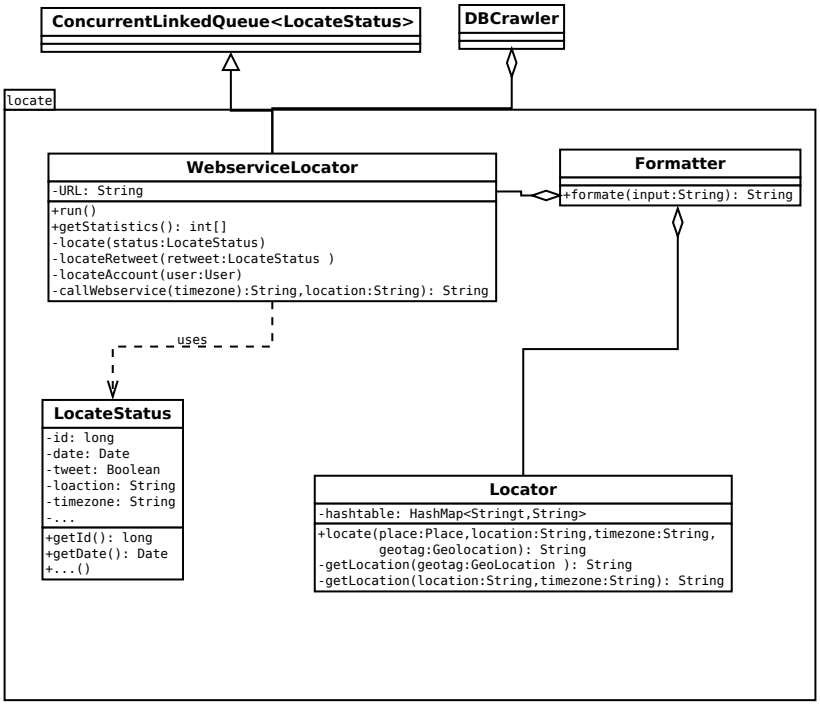

Abbildung 2.2: Klassendiagramm locate

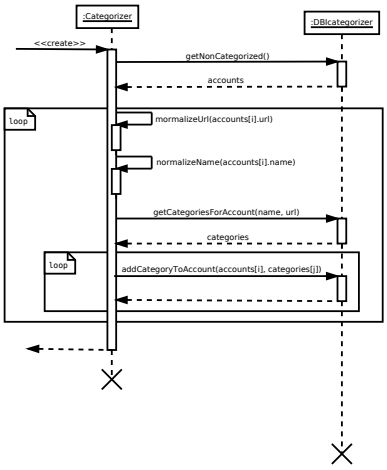

Abbildung 2.3: Sequenzdiagramm Kategorisierer

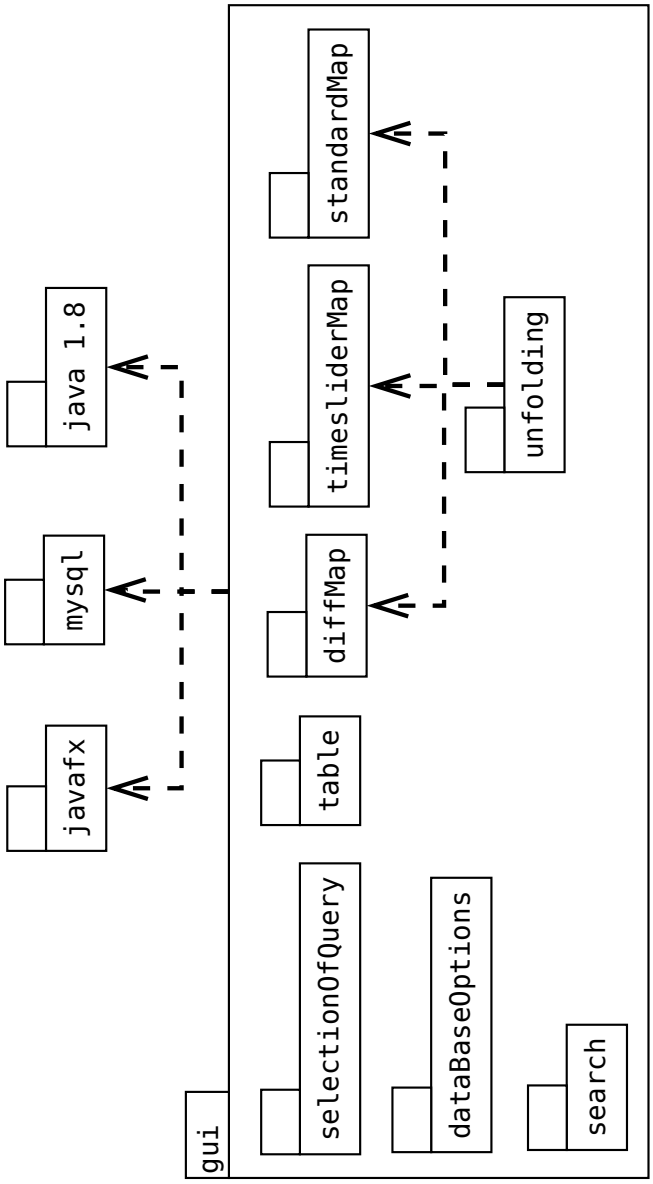

Abbildung 2.4: Paketdiagramm der GUI

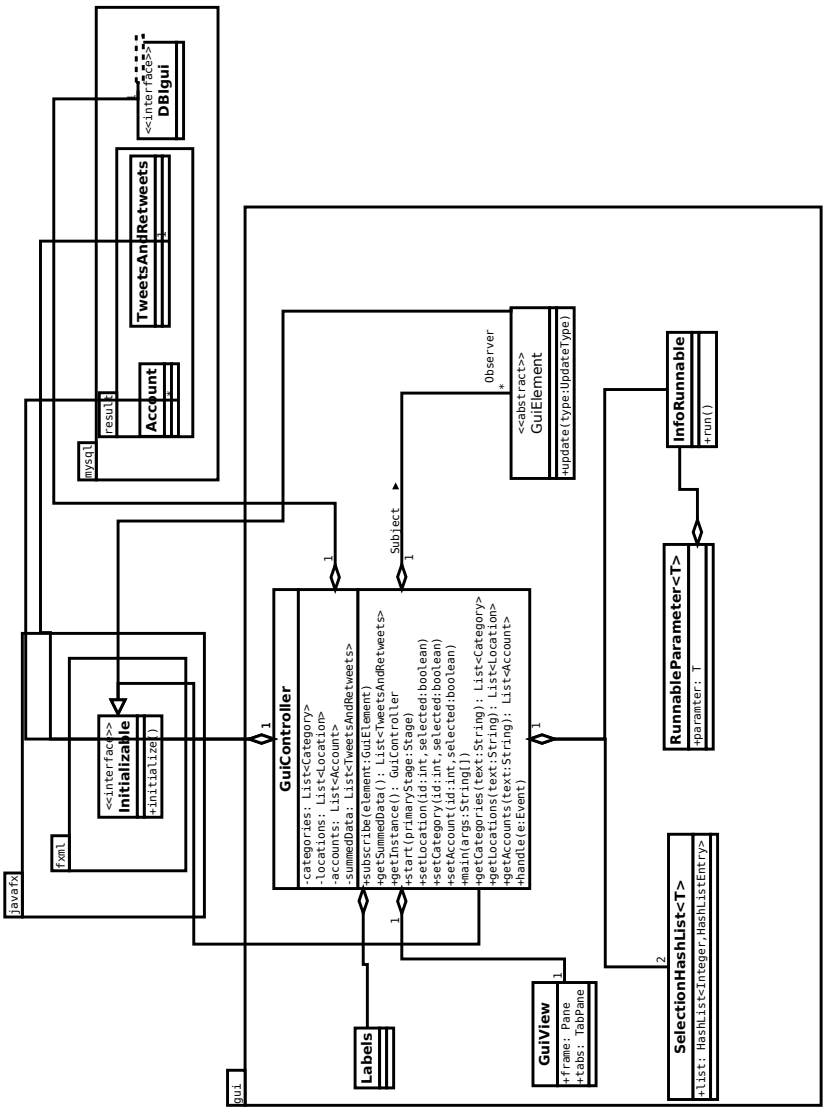

Abbildung 2.5: Klassendiagramm der GUI

3 Tests

3.1 Tests