01.07.2013

Fachhochschule
Brandenburg
University of
Applied Sciences
Fachbereich
Informatik und Medien

Dokumentation
Live-Share-Chart

Entwurf und Implementierung eines prototypischen Systems zur automatischen Anzeige von Aktiendaten

Team:
B.Sc. Andy Klay
B.Sc. Frank Mertens

Betreuer:
M.Sc. Jonas Brüstel
Ilonka Wolpert

Inhalt

Dokument-Historie

Version	Datum	Autor	Erläuterung
0.1	20.06.2013	Klay	Initial Dokumentation mit Struktur
0.5	24.06.2013	Mertens	Mehrere Kapitel angefangen zu schreiben
0.6	25.07.2013	Klay	Einführung, Funktionale Sicht
0.7	27.07.2013	Mertens	Architektur
0.8	28.06.2013	Mertens	Systemvorraus. , Bekannte Fehler
0.9	31.06.2013	Klay, Mertens	Nochmalige Überarbeitung aller Kapitel
1.0	01.07.2013	Klay, Mertens	Klassendiagramm, Final Check

1. Einführung

LiveShareChart ist ein Client-Programm(Desktop-Applikation) zur Anzeige von minutengenauen, aktuellen Aktienkursen. Die Anzeige der Aktienkurse erfolgt in einem Diagramm und die entsprechenden Daten werden aus einer Datenbank gezogen. Zu einem einzeln ausgewählten Aktienkurs werden die dazu am aktuellen Tag erschienenen News (Nachrichten) angezeigt. Weiterhin läuft im unteren Teil der Programmoberfläche ein Newsticker (Nachrichten-Laufband), der die jeweils neusten News zu allen bekannten Aktienkursen enthält. Die News-Links werden zusammen mit den Kursdaten aus der Datenbank heruntergeladen. Durch einen Klick auf einzelne News wird der News-Content in einem Web-Panel angezeigt.

Funktionaler Umfang der Idee:

- Client-Programm mit
  - Drop-Down-Box
    - Wählt und Ändert den anzuzeigenden Aktienkurs aus
    - Aktualisiert anzuzeigende News aus
  - Diagramm
    - Zeigt den Kursverlauf des heutigen Tages an
  - Update-Button
    - Erzwingt eine Aktualisierung der Aktien-Daten des Client
  - News-Liste
    - Zeigt die neusten bekannten News an
  - Web-Panel zur Anzeige von News
    - Zeigt die gewählte News in einem inline-Browser an
  - Newsticker
    - Zeigt die neuste News zu jeder Aktie(letzte jeder Aktie)
    - Zeigt heutige relative Veränderung aller Aktienkurse an
- Hilfsprogramm
  - Generierung neuer Kurswerte
  - Abfrage von aktuellen RSS-News
  - Speicherung aller Daten in der DynamoDB
- Amazon DynamoDB
  - NoSQL-Datenbank, Zugang von der FHB gestellt

Verwendete Technologien:

- Java und JavaFX
- Amazon DymnamoDB

2. Client-Funktionen

2.1 Funktionale Sicht/User-Sicht auf den Client

Der Nutzer kann, nachdem er LiveShareChart gestartet hat, folgende Aktionen tätigen:

- Auswahl einer anzuzeigenden Aktie in einer Auswahl-Box zur Anzeige von Diagramm und News der Aktie
- Anklicken der dazugehörigen News in einer Liste zur Anzeige der News
- Anklicken der News im Newsticker zur Anzeige der News
- Anklicken der Aktiennamen im Newsticker zur Anzeige der Aktie im Diagramm

Schlussendlich wurde die GUI mit einem Fenster mit zwei Tabs umgesetzt. Über den Tabs liegt unten am Rand des Fensters der News-Ticker mit News und aktuellen Kurs-Bewegungen aller Aktien, siehe Abbildung 1. Der Newsticker ist animiert und bewegt sich von rechts nach links durch das Fenster. In dem Tab „Kurse“ werden das Aktien-Diagramm, links, und rechts die dazugehörigen News zu der aktuell ausgewählten Aktie dargestellt. In dem Tab „Geöffneter Feed“ wird eine News geöffnet, ausgelöst durch den Klick auf einen News-Eintrag aus dem News-Ticker oder den Aktien-News rechts. Durch Auswahl der Choice-Box links oben wird eine Aktie gewählt. Diese Auswahl ändert dann das Diagramm und die aktuellen News rechts.

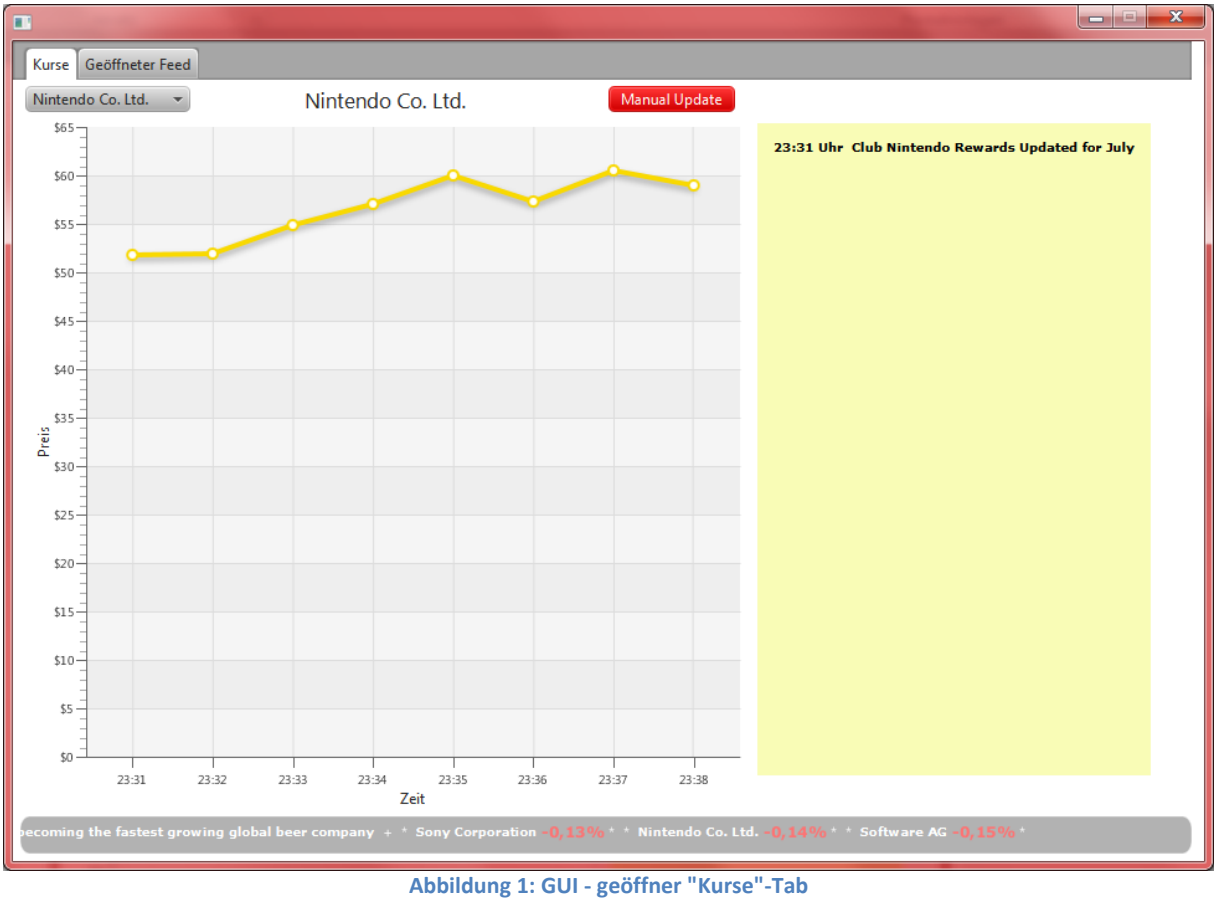

Abbildung 1: GUI - geöffner "Kurse"-Tab

3. Architektur

In Abbildung 2 ist das Schichtendiagramm von LiveShareChart zu sehen. Das Architekturmuster entspricht im Wesentlichen dem MVVM-Model[1]. In Abbildung 3 ist die Klassen-Übersicht mit Package-Struktur zu sehen, welche bei den folgenden Ausführungen behilflich sein soll. Weitere Erklärungen und Erläuterungen zu den Komponenten geben die folgenden Unterkapitel.

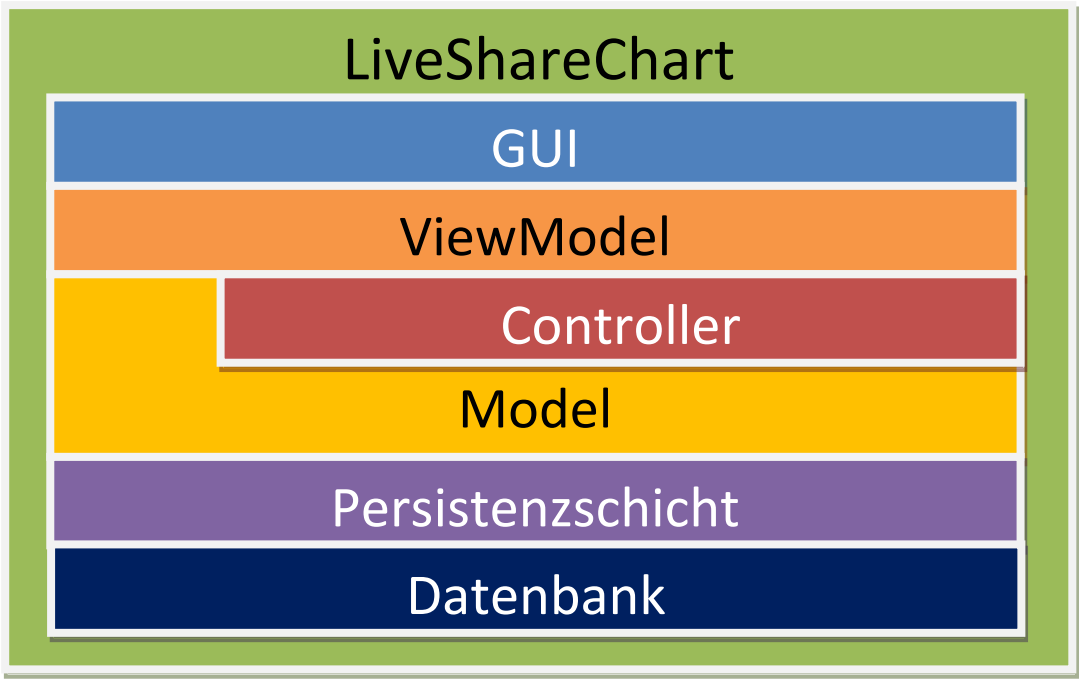

Abbildung 2: LiveShareChart MVVM-Model

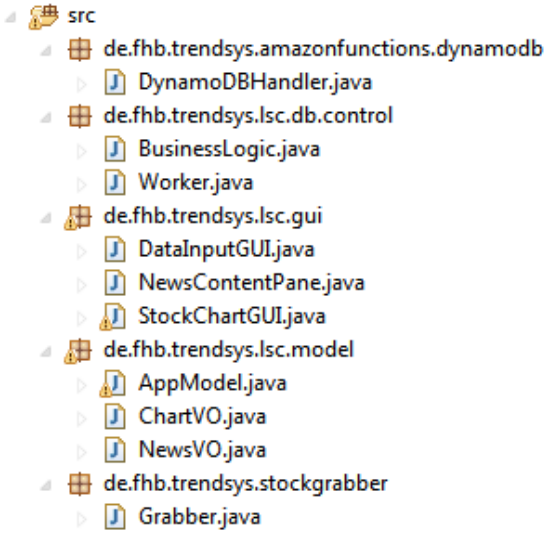

Abbildung 3: Packages und Klassenübersicht

[1] Model-View-ViewModel

3.1 GUI

Die GUI stellt die Schnittstelle für den User zum System dar und hält eine Referenz auf den Controller. Die GUI bekommt die Daten über das View-Model.

Die GUI besteht aus den Klassen StockChartGUI und NewsContentPane, siehe Abbildung 4. StockChartGUI stellt das Hauptfenster der GUI und hält eine Referenz auf die Business-Logik. NewsContentPane ist eine eigen Klasse für den Inhalt des Tabs „geöffneter Feed“, welcher ein Panel zur Ansicht von Web Content bereitstellt. Die Klasse DataInputGUI, nicht fertiggestellt, stellt ein Fenster zur Bearbeitung der DB-Einträge und ist ein Entwicklertool und nicht für den Endanwender gedacht.

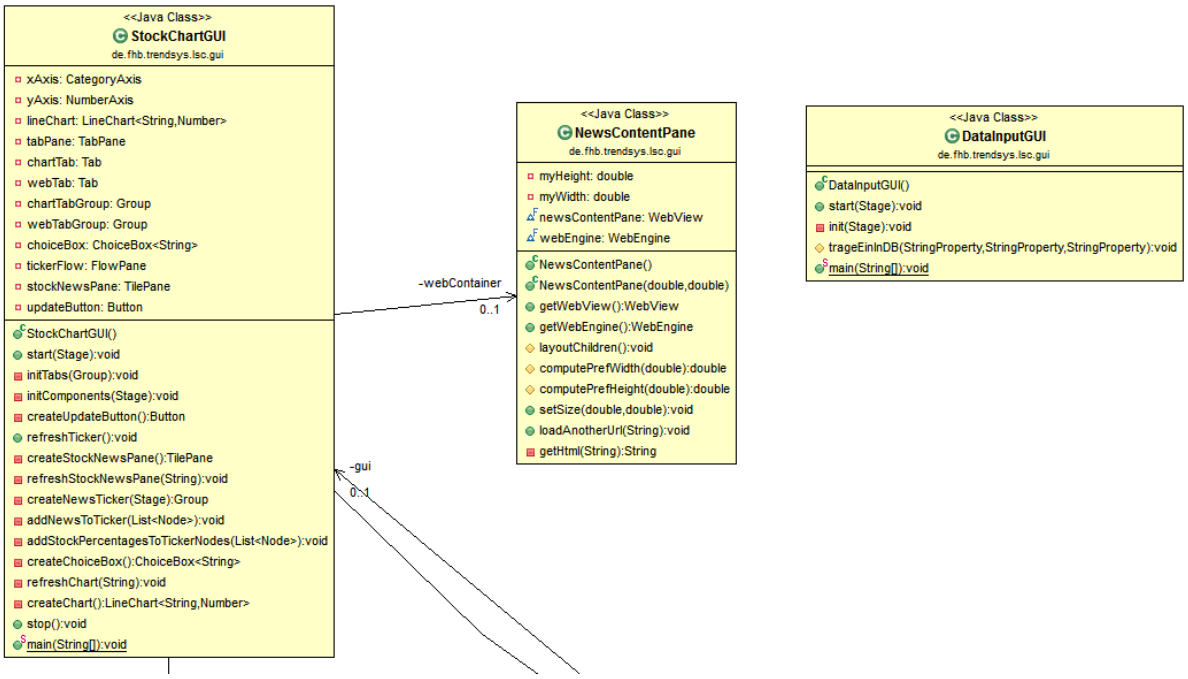

Abbildung 4: Klassendiagramm GUI

3.2 ViewModel

Das ViewModel stellt das Bindeglied zwischen dem Model und der View dar. Es dient zur Vorbereitung der Daten aus dem Model, damit sie einfach in der View angezeigt werden können. Weiterhin übernimmt es die automatische Benachrichtigung an die GUI-Elemente, wenn sich veränderte Daten auf das Model oder die View auswirken[2].

In JavaFX ist das ViewModel in Form von Collections, die ein spezielles Observable-Interface implementieren, innerhalb der GUI-Elemente umgesetzt. Diese Collections müssen ausgelesen bzw. gesetzt werden, um das Data-Binding herzustellen.

3.3 Controller

Der Controller umfasst die Klasse BusinessLogic, siehe Abbildung 5. Aufgrund des MVVM-Models hat er in diesem Programm relativ wenig Funktionalität. Das BusinessLogic-Objekt startet allerdings den Worker-Thread der Persistenzschicht.

[2] Sogenanntes Data-Binding: Automatischer uni- oder bidirektionaler Abgleich von Daten auf Basis von Events.

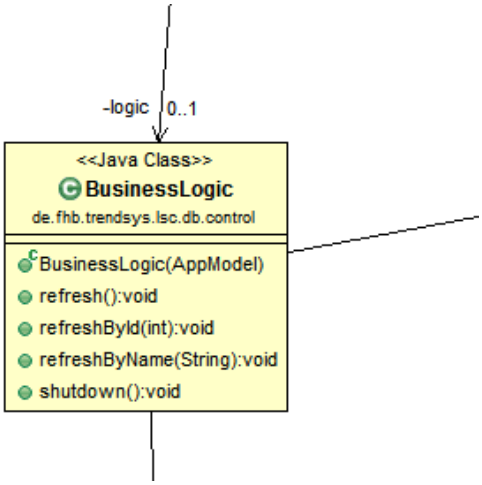

Abbildung 5: Klassendiagramm BusinessLogic

3.4 Model

Das Model hält Daten strukturiert in Form von VO[3] vor. Das Model umfasst das AppModel, welches eine Liste von ChartVO-Objekten hält, siehe Abbildung 6. ChartVO besitzt wiederum eine Liste von NewsVO-Objekten. Ein ChartVO repräsentiert eine Aktie und ein NewsVO einen NewsFeed zu einer Aktie. Der Worker liest sie von der Datenbank aus und transformiert die Daten der Datenbank durch Mapping zu VO. Die Persistenzschicht arbeitet direkt mit diesen VO.

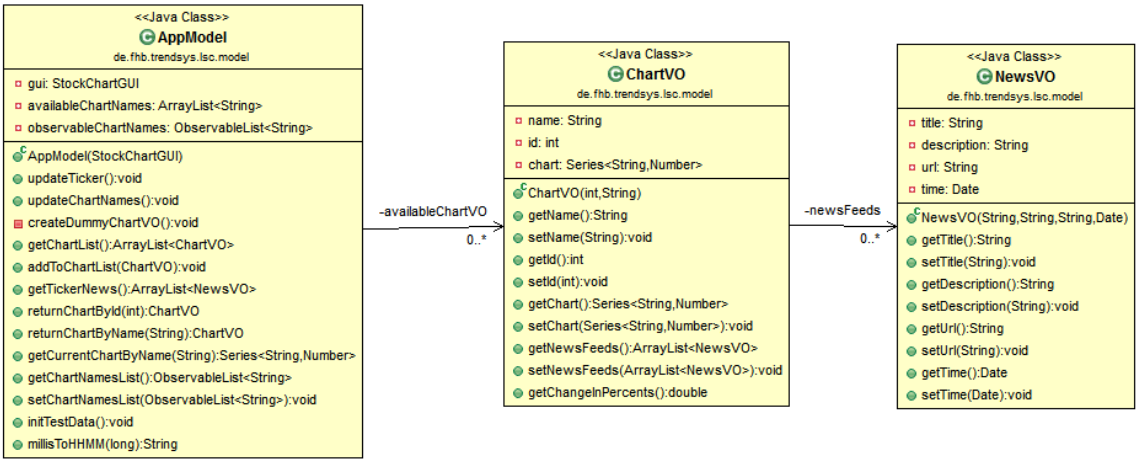

Abbildung 6: Klassendiagramm Model

3.5 Persistenzschicht

Die Persistenzschicht unterteilt sich wesentlich in zwei Bereiche: Zum einen gibt es die Klasse DynamoDBHandler, siehe Abbildung 7, die alle Anfragen an die DynamoDB formuliert, ausführt und das Ergebnis empfängt. Dazu wurden die CRUD-Operationen realisiert. Der Zugriff auf die Datenbank erfolgt über das AWS-SDK[4]. Zum zweiten gibt es einen Worker-Thread, der den DynamoDBHelper regelmäßig beauftragt, die Datenbank auf neue Daten hin zu überprüfen. Die Ergebnisse werden im

[3] Value Objects
[4] Amazon Web Services Software Development Kit

Worker verarbeitet und anschließend in die VOs gespeichert. Der Worker hat eine interne Warteschlange, über die er erkennt, wann er welche Daten abfragen muss. Wann welche Daten abgefragt werden, wird über eine Priorität geregelt. Der Aktienkurs, der gerade in der GUI angezeigt wird, wird jede Minute aktualisiert. Alle anderen Kurse werden nur alle 10 Minuten aktualisiert.

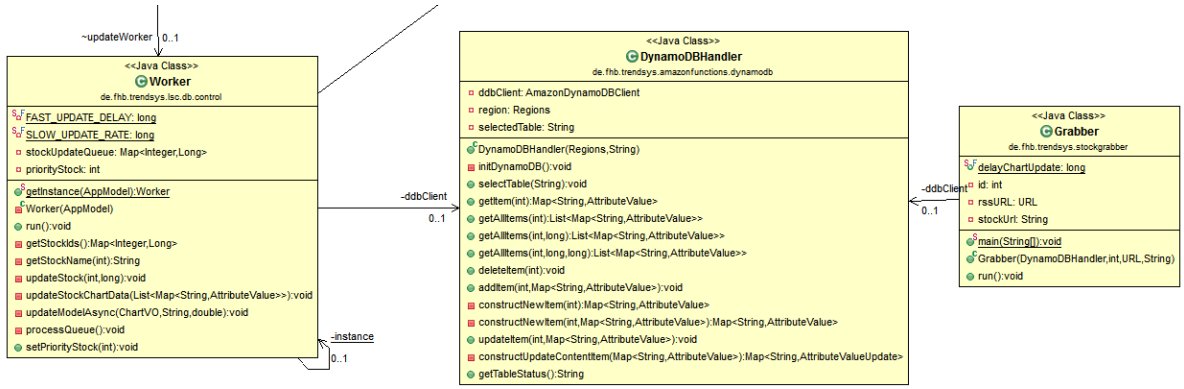

Abbildung 7: Klassendiagramm Persistenz

3.6 Datenbank

In der Datenbankschicht wird die Amazon DynamoDB verwendet. Sie ist eine NoSQL-Datenbank, die verteilte Hash-Tabellen für hohe Performanz, Skalierung und Datensicherheit bietet. Die Datenintegrität wird über ein Node-Netz sichergestellt, bei der alle für begrenzte Teile der Datenbank zuständigen Nodes eine Anfrage gemeinsam beantworten. Nur die Antwort mit den meisten „Stimmen“ wird als Ergebnis zurückgesendet. Nodes, die nicht die richtige Antwort geliefert haben, werden von den anderen Nodes aktualisiert. Ein Node dient dabei als Master-Node für einen begrenzten Teil der Daten und verfügt über Slave-Nodes. Sollte der Master-Node ausfallen, über der nächste Slave-Node in der Reihe seine Funktion, bis er wieder online geht. Anschließend wird er über die Slave-Nodes aktualisiert.

Die Hash-Tabelle der Datenbank kann Tupel mit beliebig vielen Attributen speichern. Vorausgesetzt ist nur, dass ein Primary und ein Range Key vorhanden sind. Über den Range Key sind sehr schnelle Abfrage, in diesem Fall über den Zeitstempel, möglich. In der Datenbank werden vier verschiedene Tupel verwendet. Alle verwenden mindestens die Attribute „id“ als Primary Key und „timestamp“ als Range Key.

1. Index-Tupel

Dieser Tupel hat die ID 0 und speichert in seinen Attributen alle IDs von Aktien-Tupeln, die gleichzeitig auch die IDs der Kurs-Tupel sind, siehe Tabelle 1.

Tabelle 1: Beispiel für Index-Tupel

id	*timestamp*	stockid0	stockid1	stockidn
0	„0“	„1“	„2“	…

2. Aktien-Tupel

Ein Tupel dieser Art speichert allgemeine Informationen zu einer Aktien: Firmenname, URLs zu mehreren RSS-News-Feeds sowie eine URL zur Firmenseite, die immer dann eingesetzt wird, wenn die URL zum RSS-Feed nicht verfügbar ist, siehe Tabelle 2.

Tabelle 2: Beispiel für einen Aktien-Tupel

id	*timestamp*	stockname	stockurl	rss0
1	„0“	„Sony Corporation“	„http://www.sony.net“	…

3. Kurs-Tupel ohne News

Kurs-Tupel enthalten nur den aktuellen Aktienkurs zum Zeitpunkt der Erstellung, siehe Tabelle 3.

Tabelle 3: Beispiel für einen Kurs-Tupel ohne News

id	*timestamp*	stock
1	„1372658392586“	„0,23567882007588792“

4. Kurs-Tupel mit News

Kurs-Tupel mit News enthalten zusätzlich Attribute über eine News: Titel, Beschreibung und die URL zur News, siehe Tabelle 4.

Tabelle 4: Beispiel für einen Kurs-Tupel mit News

id	*timestamp*	stock	newstitle	newsdescription	newsurl
1	„137265…“	„0,235…“	„DSC-RX1R“	„Starke Leistung …“	„http://www.sony...“

4. Systemvoraussetzungen

4.1 Allgemeine Voraussetzungen

Client und der Grabber:
- Windows-, MacOS- oder Linux-Betriebssystem
- Java 1.7
- eine Amazon DynamoDB

4.2 Hinweise zur Installation

Vor der ersten Verwendung muss die DynamoDB eingerichtet werden. Es wird eine Provisioned Throughput Read und Write Capacity von 1 benötigt. Es muss eine Tabelle mit den Namen "stockdata" angelegt werden. Der Primary Key muss „id“ lauten und vom Typ Number sein, der Range Key „timestamp“ und vom Typ String.

Anschließend werden die Aktien-Tupel angelegt. Die ID muss dabei verschieden von 0 sein, der timestamp muss „0“ sein. Alle anderen Attribute müssen ebenfalls vorhanden sein.

Dann wird das Index-Tupel angelegt. Die ID muss 0 sein, der timestamp muss „0“ betragen. Weitere Attribute beginnen mit „stockid“ und müssen als Wert je eine der IDs der Aktientupel haben.

Jetzt kann der Grabber entsprechend eingerichtet werden, in dem in der main-Methode Threads zu den einzelnen Aktien gestartet werden. Der Grabber erzeugt pro Aktien einen Kurs-Tupel mit eventuell vorhandenen News pro Minute und speichert diesen in der Datenbank.

5. Bekannte Fehler

1. Die Newsticker-Animation bricht ab, bevor sie den kompletten Text nach links hinausgescrollt hat.
2. GUI – Die Newsliste auf der rechten Seite vergrößert sich unter dem Fensterrand hinaus, wird aber am Rand geclippt. Verursacht wird das durch lange, nicht umgebrochene News-Titel.
3. DataInputGUI – Für den Grabber wurde eine GUI vorbereitet, die jedoch nicht mit dem Programm verbunden und damit funktionslos ist.
4. Bei Umwandlung der Timestamps für die Oberfläche in hh:mm –Format hält die Observable-List der Unterklasse Series die Reihenfolge der Datensätze scheinbar nicht mehr ein, so dass dieses Feature zur Abgabe kurzfristig deaktiviert werden musste und nun statt Minutenangaben immer noch Millisekunden als Einheit in der X-Achse angeben werden

6. Ausblick

Weitere mögliche Features:

- Einbettung als JNPL-File in ein HTML-Dokument

Tabellenverzeichnis

Abbildungsverzeichnis

A Anhang

A1 Komplettes Klassendiagramm