Last.forward

Entwickler Dokumentation

Endlich, Christian	**853223**
Knoll, Thomas	**853856**
Scherer, Heike	**854907**

Fachhochschule Kaiserslautern, Sommersemester 2007, Prof. Hendrik Speck
http://lastforward.sourceforge.net

Inhalt

1 Einleitung 4
1.1 Was ist Last.forward ? 4
1.2 Lizenz und Disclaimer 4
1.3 Weiterentwicklung 5
1.4 Entwicklerdokumentation 5
1.5 Installationsanleitung unter Eclipse oder anderen IDEs 5
2 Entwicklungsumgebung 7
2.1 Systemvoraussetzungen 7
2.2 Programmiersprachen 7
2.3 Entwicklungsumgebung 7
2.4 Hardwareanforderungen 7
2.5 Softwareanforderungen 7
3 Datenbankstruktur Last.forward 8
3.1 Datenbankdesign 9
4 Packages 11
5 Klassenbeschreibungen 12
5.1 Package de.lastfm.gui 12
5.1.1 About.java 12
5.1.2 ActionEventHandler.java 12
5.1.3 GraphPanel 13
5.1.4 TreeDisplay 14
5.1.5 Gui.java 14
5.1.6 Help.java 15
5.1.7 Profile.java 16
5.1.8 Searchfield.java 16
5.1.9 JFilterPanel.java 16
5.2 Package de.lastfm.listener 17
5.2.1 Agecomboboxlistener.java 17
5.2.2 CenteringControl.java 17
5.2.3 ComboxListener.java 17
5.2.4 CountryComboboxListener.java 17
5.2.5 PlaycountComboBoxListener.java 17

5.2.6 ZoomControl.java 18
5.3 Package de.lastfm.entity 18
5.3.1 ProfileEntity.java 18
5.4 Package de.lastfm.analysis 18
5.4.1 JAnalysisPanel.java 18
5.4.2 Pie3D.java 18
5.5 Package de.lastfm.db 19
5.5.1 DB_Friends.java 19
5.5.2 DB_User.java 19
5.5.3 DB_Neighbour.java 19
5.5.4 DB_Groups.java 19
5.5.5 Dbconnector.java 19
5.5.6 DBReader.java 19
5.5.6 CountryComperator.java 19
5.5.7 HSQLDBCreator.java 19
5.6 Package de.lastfm.crawler 20
5.6.1 Crawl.java 20
5.6.2 Friends.java 20
5.6.3 Neighbours.java 20
5.6.4 Groups.java 20
5.7 Package Images 20
6 Klassendiagramm 21
7 Kontakt 22

1 Einleitung

1.1 Was ist Last.forward ?

Last.forward ist eine Open Source Software zum Analysieren und Visualisieren der der Musikplattform lastfm.de. Die hierfür benötigten Daten werden zunächst mit einer entsprechenden Applikation (Crawler) gesammelt. Anschließend werden diese Daten gesplittet und aufbereitet in eine MYSQL-Datenbank geschrieben. Das Visualisierungs-Tool verarbeitet die Daten und stellt sie in verschiedenen 2 verschiedenen Graphen dar.
Zum Einen bieten wir eine Sicht als Baum Diagramm an. Die Hauptvisualisierung besteht bei unserem Projekt in Form eines gerichteten Graphen, Es werden die Beziehungen zwischen dem einzelnen Nutzern und seinem Netzwerk bei Lastfm.de sowie seine Gruppenzugehörigkeit angezeigt. Weiterhin ermitteln wir die musikalischen Verwandten, bei lastfm.de Nachbarn genannt, und stellen auch diese in unserer Visualisierung dar. Unsere Anwendung ermöglicht es sich die Profil-Details, jeden im Graphen dargestellten User, anzuzeigen. Zudem ist es möglich, durch das Ein- und Ausschalten von verschiedenen Filtern, die Graphen zu modifizieren. Weiterhin bieten wir 3 statistische Auswertungen im Form von Kuchendiagrammen an. Eine davon wertet die Herkunft der musikalischen Freunden aus. Eine Zweite die Altersverteilung der direkten Freunde. Weiterhin werten wir den Anteil von Frauen und Männern über alle sozialen Beziehungen aus.
Hier erzeugen wir einen entscheidenden Mehrwert gegenüber der Homepage von Last.fm.de.
Ein weiterer großer Vorteil unserer Anwendung ist die übersichtliche Darstellung der sozialen Beziehungen eines registrierten lastfm-User, die so auf der Homepage des Betreibers nicht ersichtlich ist.

Das Last.forward Projekt wurde im Rahmen der Lehrveranstaltung Medienkonzeption- und Produktion Sommersemester 2007 an der Fachhochschule Kaiserslautern, Standort Zweibrücken, gestartet. Aufgabe war es, eine Software zur Visualisierung eines sozialen Netzwerkes zu entwickeln. Die Hauptmerkmale wurden hierbei auf die folgenden drei Punkte gelegt:

- Sammeln der Informationen mittels Crawler
- Speichern der Daten in einer Datenbank
- Auswertung und Anzeigen der geforderten Informationen mittels eines entsprechenden Algorithmus

Um eine Weiterentwicklung der Software zu gewährleisten, wurde das Projekt unter der Open Source Lizenz realisiert und auf der SourceForge.net Seite zugänglich gemacht.

1.2 Lizenz und Disclaimer

Last.forward wird unter der GNU General Public License (GPL) entwickelt. Mehr Informationen hierzu finden Sie unter http://www.opensource.org. Last.forward übernimmt keine Haftung für eventuelle Schäden, die durch Nutzung bzw. Weiterentwicklung des Crawlers oder der Visualisieung entstehen könnten. Das Programm kann nicht nur einfach benutzt werden, sondern der Programmcode ist durch die OpenSource Lizenz auch für jeden zugänglich, kann erweitert und ins eigene OpenSource-Programm eingebaut werden.

1.3 Weiterentwicklung

Durch eine Zusammenarbeit mit lastfm.de wäre eine einfachere Auswertung der Daten aus deren Datenbank möglich. Dadurch würde die Crawler-Komponente erheblich entlastet und man könne eine Mehrzahl an zusätzlichen Informationen anbieten. Weiterhin war das Ziel dieses Projektes es unter dem Motto eines
Open-Source-Projekt laufen zu lassen, damit die Weiterentwicklung auch durch Mitglieder der Open-Source-Gemeinde geschehen kann. Dadurch wollen wir eine performante Weiterentwicklung erzielen.

1.4 Entwicklerdokumentation

Ziel dieser Dokumentation ist es, Entwicklern bei der Weiterentwicklung bzw. bei der Nutzung bestehenden Quellcodes zu unterstützen. Dieses Dokument dient zum besseren Verständnis des Aufbaus unserer entwickelten Anwendung. Als Grundvoraussetzung wären Grundkenntnisse in Java 5.0 von Vorteil. Darüber hinaus empfehlen wir, sich in das Open Source Java Visualisierungstoolkit Prefuse (www.prefuse.org) einzuarbeiten.

1.5 Installationsanleitung unter Eclipse oder anderen IDEs

Nachdem das Projekt über CVS herunter geladen und in den Eclipse – Workspace eingebunden wurde, müssen zusätzliche Libraries bekannt gegeben werden. Diese befinden sich auf der Hauptebene des Projektes. Beim Anlegen des Projektes ist darauf zu achten dieses als Java – Projekt zu deklarieren.
Folgende Libraries finden bei uns Verwendung:

- Prefuse.jar
- Htmlparser.jar
- Jfreechart101.jar
- Mysqlconnector-java.jar
- BareBonesBrowserLaunch.jar
- Quaqua.jar
- Gnujaxp.jar
- Servlet.jar
- Hsqldb.jar

Zum Einbinden einer eigenen Datenbank müssen im package de.lastfm.db die Datei dbconnector.java folgende 4 Werte abändern:

```
private static final String PASSWORT = "";
private static final String USER = "sa";
private static final String URL = "jdbc:hsqldb:file:lastfm";
private static final String JDBC_DRIVER = "org.hsqldb.jdbcDriver";
```

oder

```
private static final String PASSWORT = "";
private static final String USER = "root";
private static final String URL = "jdbc:mysql://localhost/lastfm";
private static final String COM_MYSQL_JDBC_DRIVER ="com.mysql.jdbc.Driver";
```

Es wird dringend empfohlen, die Datei „lastfm.sql“ aus dem CVS Repository CVS\lastforward\lastfm in Ihre Datenbank zu importieren. Diese enthält ein Skript zum Anlegen der Datenbank sowie die Grundstruktur der benötigten Tabellen. In dieser Mysql-Exportdatei sind schon verschiedene User vorhanden. Unter anderem „Angelsson“, „-Sayonara-“, „asmodi“ und „caitlinm“.

Diese Angaben basieren noch auf der Verwendung von MY-SQL. Die Anwendung wurde jedoch im Laufe des Projektes auf die userfreundliche Datenbank HSQLDD um dem User eine Einrichtung einer DB auf seinem Rechner zu ersparen. Gecrawlte Ergebnisse unsere Anwendung bleiben somit konsistent bestehen. Mehr Infos findet man hierzu unter http://www.hsqldb.org/.

Ihr ProjektOrdner sollte unter anderem die Datei lastfm.sql enthalten, wenn sie sich entschließen sollten mit HSQLDB zu arbeiten. In dieser Datei stehen alle INSERTS die bei Programmstart eine HSQLDB anlegt und mit dieser arbeitet. Auch in dieser sind die vorher genannter Nutzer verfügbar.

2 Entwicklungsumgebung

2.1 Systemvoraussetzungen

Zur Weiterentwicklung von Teilen des Quellcodes von Last.forward sind folgende Komponenten notwendig:

- Java Development Kit JDK 5.0 der Java Standard Edition J2SE
- My-SQL Datenbank oder
- HSQLDB (früher Hypersonic)
- Prefuse
- JFreeChart
- http-Parser

2.2 Programmiersprachen

Die gesamte Anwendung wurde in Java 5.0 entwickelt, wobei für die Haupt-Visualisierungskomponente das Prefuse-Toolkit verwendet wurde. Der Code ist ausreichend kommentiert. Spezielle Anmerkungen werden in dieser Dokumentation näher erläutert.

2.3 Entwicklungsumgebung

Das Projekt wurde komplett mit Eclipse inklusive CVS Plug-In entwickelt, wodurch ein ständiger Versionsabgleich möglich war. Weiterhin nutzen wir TortoiseCVS zur zusätzlichen Versionskontrolle. Die SQL-Queries wurden mit phpMyAdmin erstellt und anschließend in die entsprechenden Java Klassen als statische PreparedStatements übernommen.

2.4 Hardwareanforderungen

Die Hardwareanforderungen legen wir auf Grund der starken leistungszehrenden Visualisierung wie folgt fest:

- Pentium oder ähnliches mit mindestens 2GHz oder Centrino
- 1024 MB Arbeitsspeicher
- DSL-Internetzugang (damit last.forward genügend Ressourcen aufbauen kann)
- Grafikkarte mit einer Mindestauflösung von 1024x768Pixel

2.5 Softwareanforderungen

Damit unsere Software genutzt oder weiterentwickelt werden kann, empfehlen wir folgende Software:

- Betriebssystem: Windows, Linux MacOS oder Ähnliches
- Java Software Developer Kit (Java SDK),
- Java Runtime Environment (JRE) Version 1.5
- Entwicklungsumgebung Eclipse

3 Datenbankstruktur Last.forward

Tabellenstruktur für Tabelle groupname

Feld	Typ	Null
name	Varchar (120)	nein
bildurl	Varchar (120)	nein

Tabellenstruktur für Tabelle is_friend

Feld	Typ	Null
uid (ID des Users)	Varchar (120)	nein
fid (ID des Freundes)	Varchar (120)	nein

Tabellenstruktur für Tabelle is_neighbour

Feld	Typ	Null
uid	Varchar (120)	nein
Nid (ID des Nachbarn)	Varchar (120)	nein

Tabellenstruktur für Tabelle is_member

Feld	Typ	Null
gid (Gruppenname)	Varchar (120)	nein
uid (ID des Users)	Varchar (120)	nein

Tabellenstruktur für Tabelle userprofile

Feld	Typ	Null	Standard
Id	Varchar (8)	Nein	
realname	Varchar (120)	Ja	*NULL*
nickname	Varchar (120)	Nein	
age	Varchar (3)	Ja	*NULL*
gender	Varchar (1)	Ja	*NULL*
playcount	Varchar (50)	Ja	*NULL*
country	Varchar (120)	Ja	*NULL*
registered	Varchar (20)	Ja	*NULL*
url	Varchar (120)	Ja	*NULL*
bildurl	Varchar (120)	Ja	*NULL*
crawlflag	Varchar (1)	Nein	0

Diese DB kann mit dem File : lastfm_mysql.sql aus dem CVS angelegt werden.

3.1 Datenbankdesign

Tabelle User (Speicherung der Userprofile)

```
CREATE TABLE `userprofile` (
  `id` varchar(8) collate latin1_general_ci NOT NULL,
  `realname` varchar(120) collate latin1_general_ci default NULL,
  `nickname` varchar(120) collate latin1_general_ci NOT NULL,
  `age` varchar(3) collate latin1_general_ci default NULL,
  `gender` varchar(1) collate latin1_general_ci default NULL,
  `url` varchar(120) collate latin1_general_ci default NULL,
  `bildurl` varchar(120) collate latin1_general_ci default NULL,
  `country` varchar(120) collate latin1_general_ci default NULL,
  `registered` varchar(20) collate latin1_general_ci default NULL,
  `playcount` varchar(50) collate latin1_general_ci default NULL,
  `crawlflag` varchar(1) collate latin1_general_ci NOT NULL default '0',
  PRIMARY KEY  (`id`)
)
```

Tabelle is_friend (Speicherung der Freundschaftsbeziehungen)

```
CREATE TABLE `is_friend` (
  `uid` varchar(120) collate latin1_general_ci NOT NULL,
  `fid` varchar(120) collate latin1_general_ci NOT NULL,
  PRIMARY KEY  (`uid`,`fid`)
)
```

Tabelle is_neighbour (Speicherung der Nachbarschaftsbeziehungen)

```
CREATE TABLE `is_neighbour` (
  `uid` varchar(120) collate latin1_general_ci NOT NULL,
  `nid` varchar(120) collate latin1_general_ci NOT NULL,
  `matchvalue` varchar(6) collate latin1_general_ci NOT NULL,
  PRIMARY KEY  (`uid`,`nid`)
)
```

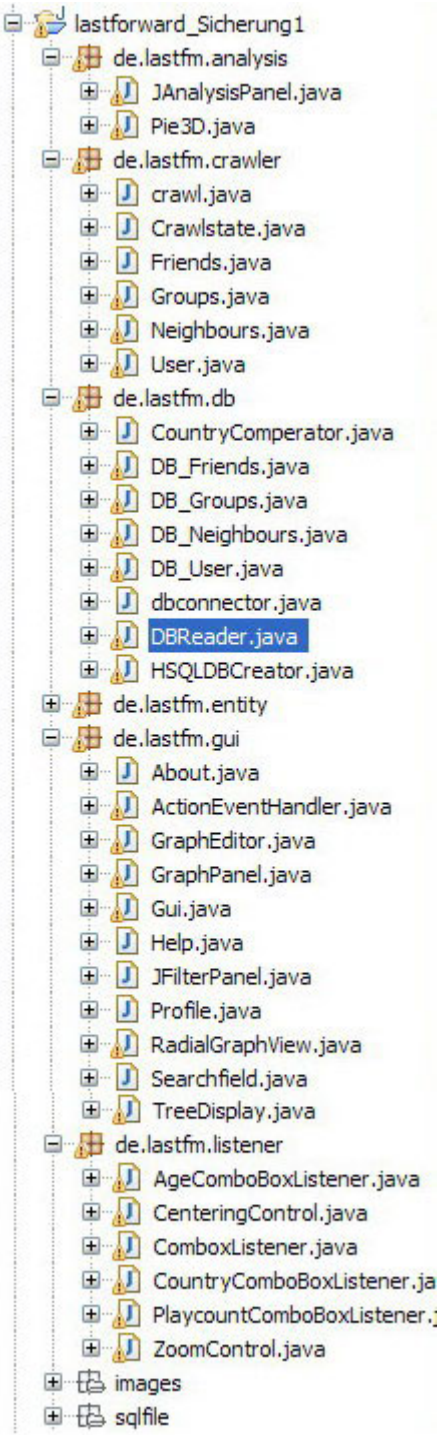
Tabelle is_member (Speicherung der Gruppenmitgliedschaften)

```
CREATE TABLE `is_member` (
  `gid` varchar(120) collate latin1_general_ci NOT NULL,
  `uid` varchar(120) collate latin1_general_ci NOT NULL,
  PRIMARY KEY  (`gid`,`uid`)
)
```

Tabelle group (Speicherung der Gruppen)

```
CREATE TABLE `groupname` (
  `name` varchar(120) collate latin1_general_ci NOT NULL default 'lastfm',
  `bildurl` varchar(120) collate latin1_general_ci NOT NULL,
  PRIMARY KEY  (`name`)
)
```

4 Packages


lastforward_Sicherung1
de.lastfm.analysis
JAnalysisPanel.java
Pie3D.java
de.lastfm.crawler
crawl.java
Crawlstate.java
Friends.java
Groups.java
Neighbours.java
User.java
de.lastfm.db
CountryComperator.java
DB_Friends.java
DB_Groups.java
DB_Neighbours.java
DB_User.java
dbconnector.java
DBReader.java
HSQLDBCreator.java
de.lastfm.entity
de.lastfm.gui
About.java
ActionEventHandler.java
GraphEditor.java
GraphPanel.java
Gui.java
Help.java
JFilterPanel.java
Profile.java
RadialGraphView.java
Searchfield.java
TreeDisplay.java
de.lastfm.listener
AgeComboBoxListener.java
CenteringControl.java
ComboxListener.java
CountryComboBoxListener.ja
PlaycountComboBoxListener.
ZoomControl.java
images
sqlfile

5 Klassenbeschreibungen

Es folgen nun kurze Klassenbeschreibungen, um Ihnen den Einstieg in den Code zu erleichtern. Es wird allerdings nicht zu detailliert auf die verschiedenen Klassen eingegangen, um die Übersicht zu wahren. Wollen Sie mehr über bestimmte Klassen erfahren, öffnen Sie diese und werfen Sie einen Blick auf die ausreichende Kommentierung. Wir werfen hier auch nur einen Blick auf die wichtigsten Klassen und Methoden die Last.forward anbietet und die zur Weiterentwicklung wichtig sind.

5.1 Package de.lastfm.gui

Dieses Package enthält folgende Klassen :

- About.java
- ActionEventHandler.java
- GraphPanel.java
- TreeDisplay.java
- Gui.java
- Help.java
- Profile.java
- Searchfield.java
- JFilterPanel

5.1.1 About.java

Diese Klasse ist ein Menu-Item der Menu-Bar "Info" . In der GUI wird dieses JPanel (Null-Layout, d.h. absolute Positionierung der Elemente) mit getAboutPanel() aufgerufen. Es besitzt eine Kurzbeschreibung des Projektes und informiert über die Entwickler sowie die Versionsnummer von Last.forward.

5.1.2 ActionEventHandler.java

Diese Klasse enthält eine statische Methode performSearch(). Diese wird nach bestätigter Eingabe eines Usersnamen in der GUI aufgerufen. Zuerst nimmt sich diese Methode den Input-String im Searchfield und testet auf der DB ob dieser User bereits in einem gecrawlten Format vorliegt. Hierzu dient in der der DB das Crawlflag. Dieses signalisiert ob dieser User komplett mit seinen Daten gecrawlt wurde. Steht dieses Flag auf 0, liegt nur rein das Userprofil vom gesuchten User in der MySQL-DB. Anschließend wird durch DBReader.*searchByNickname*(input) eine Instanz der Klasse ProfileEntity erzeugt. In diesem Objekt speichern wir uns alle Userprofil-Daten aus der DB und setzen damit die im ProfilePanel verwendeten Profil-Daten-Felder. Über die im ProfilEntity gefundene „bildurl“ laden wir aus dem Internet dieses Image ins ProfilePanel. Anschließend wird die TreeView und die GraphView auf „visible = true “ geschaltet und ein Standardbaum erzeugt. Hierzu später mehr bei TreeDisplay.java.

Nach dieser Initialisierung werden 3 HashSets durchlaufen die jeweils die Knoten am Standardbaum anfügen. Zum einen die Gruppenknoten, und zum anderen die Knoten Freunde und Nachbarn.

Im Anschluss wird durch eine Iterierung durch den neu erzeugten Baum die Datasets für die 3D-Menüs generiert. Diese werden später vom StatisticPanel aufgenommen und über Instanzen der Klasse Pie3D.java erzeugt.

5.1.3 GraphPanel

Diese Klasse ist Dreh- und Angelpunkt für die Graph-Visualisierung. In ihr wird der Graph für die Visualisierung erzeugt, beziehungsweise werden hier die Informationen zur Ermittlung der Statistiken erzeugt.

Zuerst stehen in dieser Klasse verschiedene Methoden die von den Filtern benutzt werden um Knoten aus den Ansichten auszublenden. Diese werden über den jeweiligen ActionEventHandler, im weiteren AEH genannt, im Filterpanel aufgerufen. Für die Arbeitsweise in beiden Ansichten, werden diese AEH in beiden Views registriert. Hierzu mehr im Paket „de.lastfm.listener“.

Weiterhin befinden sich hier die Methoden die das Centering und das Zoomen über die Knöpfe am oberen Rand der Anwendung garantieren. Diese Methoden sind mit beiden AEH in den Ansichten verknüpft und auch dort registriert.

Die Methode readCountries(Graph g) kann ein Graph übergeben bekommen, bei dem Sie ermittelt welche Länder in den einzelnen Knoten vorkommen. Diese Methode erstellt eine Vector, der später dazu benutzt wird dem Country-Filter die entsprechenden Werte zu setzen.

Die Methode readGraphAnalysis(Graph g) bekommt auch einen Graph übergeben und ermittelt alle relevanten Daten im bestehenden Graphen. Hierzu zählen die Auswertung der männlichen und weiblichen Personen die an Beziehungen teilnehmen, sowie die Ermittlung der Länderverteilung der „Nachbarn“. Weiterhin ermittelt diese Methode wie viele Personen die sich an direkten Freundesbeziehungen beteiligen und zu welchen Altersgruppen diese gehören.

Wichtigste Einheit ist der Konstruktor dieser Klasse **public** GraphPanel(Graph g).

Er erzeugt im Prinzip das Grundgerüst für die GraphView. Hier werden die ganzen Renderer und TupleSets festgelegt die das Aussehen des Graphen steuert. Eine genauere Beschreibung der einzelnen Schritte findet man auf der Homepage des Toolkits Prefuse (www.prefuse.org). Auch das Forum zum Projekt „Prefuse“ auf sourceforge.net kann hier sehr hilfreich sein.

Die innere Klasse **class** DoubleClickController **extends** ControlAdapter bewerkstelligt die Aufgabe bei Click eines Knoten die Filter zurückzusetzen, sowie den Knotennamen im Eingabefeld zu setzen. Somit soll ein Crawlvorgang explizit verhindert werden und dem User die Möglichkeit geben diesen Namen explizit zu crawlen. Weiterhin erledigt diese innere Klasse die Füllung des ProfilPanel mit ihren Werten durch Aufruf der Methoden *searchByNickname*(input) und *entityReceiver*(profileEntity).

Die Methoden :

itemEntered(VisualItem item, MouseEvent e)und itemExited(VisualItem item, MouseEvent e)

… sind für die Vergrößerung eines mit der Maus überfahrenen Knoten zuständig.

Die innere Klasse **class** NodeColorAction **extends** ColorAction ist für die Colorierung der einzelnen Nodes im Graphen verantwortlich. Hier werden explizit aus jedem Knoten das Prädikat „gender“ ausgelesen und dementsprechend die Farben gesetzt.

Die innere Klasse **class** DemoControlToolTip **extends** ControlAdapter ist für den Tooltipp zuständig. Sie lädt bei einem angetippten Knoten dessen Inhalt in einem übersichtlichen Tooltip-Kontext.

5.1.4 TreeDisplay

Die Klasse TreeDisplay ist hauptverantwortlich für die komplette für die komplette Visualisierung. Hier wird ein Standardbaum erzeugt der später der Klasse GraphPanel übergeben wird. Neben den Predicate-Filter-Methoden und einigen bekannten Methoden aus der Klasse GraphPanel, finden wir hier noch ein paar nicht weiter nennenswerte GETTER und SETTER – Methoden. Wichtigster Punkt ist der Konstruktor diese Klasse. Hier verweisen wir auf die Klasse GraphPanel deren Aufbau gleich ist. Weiterhin auch hier nochmal der Verweis auf die Homepage des Toolkits Prefuse und seine Benutzerdokumentation, in der beispielhaft der Aufbau dieses Konstruktors nahe gebracht wird. (www.prefuse.org). Die wichtigste Methode ist createTreeSample(). Zuerst wird hier eine neues Objekt vom Typ Tree angelegt. Anschliessend wir die Table des TREE initialisiert und mit Werten versehen. Demnach enthält der TREE eine DB-ähnliche Struktur. Nach diesem Schritt wird der Standardknoten angelegt und mit den in der TREE-Table vorgesehen Werten individuelle auf die Spalte angepasst. Diese ROOT-Node ist später Ausgangspunkt der beiden Ansichten und der Beginn einer jeden Visualisierung dieser Art. Ein weiterer Schritt ist nachfolgend das Anlegen 3 weitere Knoten für die sozialen Zugehörigkeitsbeziehungen Freunde, Nachbarn und Gruppen. Nun ist der Standardbaum fertig und kann später mit weiteren Knoten gefüllt werden. Hierzu mehr im Paket de.lastfm.gui in der Klasse gui, Methode performSearch().

5.1.5 Gui.java

Diese Klasse erzeugt unter anderem, wie der Name schon sagt die Gui. Jedoch geschieht hier noch viel mehr, was auch direkt auf den Aufbau der Visualisierung abzielt.

Diese Klasse erzeugt die Gui, das User Interface (das Erscheinungsbild). Die Gui ist das Herzstück der Anwendung. Hier werden alle Klassen vereinigt und ins rollen gebracht. Jedoch geschieht hier noch viel mehr, was auch direkt auf den Aufbau der Visualisierung abzielt.Die Gui ist wie folgt aufgebaut:
Der Constructor beschreibt das Layout, die Größe und setzt den Titel der Anwendung. Das Layout (Border Layout, Orientierung an Himmelsrichtungen) besteht aus drei Teilen:

North (**getMenuToolPane()**):
- Menu-Bar mit den Items Help und About last.forward.
- Tool-Bar, in welcher sich das Searchfield, die Zoom-Buttons, das last.forward- Logo sowie

das Audioscrobbler-Logo mit direktem Link zur Website befindet.
Center (**getJTabbedContent()**):

- Start-Screen (dieser Tab wird nach erfolgreicher Suche entfernt, siehe Searchfield.java)

- TreeView (siehe TreeDisplay.java)

- GraphView (siehe GraphPanel.java)

East (**getJTabbedInfo()**):
- User Data (siehe Profile.java)
- Filter (siehe JFilterPanel.java)
- Statistics (siehe Package de.lastfm.analysis)

Wichtigste Methode ist hier die Methode actionPerfom(). Sie erweitert den in der Klasse TreeDisplay erzeugte Baum (TreeView) um weitere Knoten und hängt diese an die jeweiligen Standardknoten Freunde, Nachbarn und Gruppen.
Nach Eingabe eines Usernamens im Searchfield wird geprüft ob diese User bereits im gecrawlten Format in der Datenbank zur Verfügung steht. Sollte dies nicht der Fall sein, starte der Live-Crawlvorgang. Ist der User bereits mit dem Crawlflag in der DB versehen startet die Anwendung.
Zuerst wird nun über die Methode DBReader.*searchByNickname*(input) das Profil des Gesuchten im ProfilePanel angezeigt. Eine genauere Beschreibung dieser Methode finden Sie in der Beschreibung der Klasse DBReader im Paket „de.lastfm.db“. Anschließend folgt die Erweiterung des UR-Baumes um die neuen Knoten die zum Gesuchten in der DB abgespeichert sind. Weiterhin werden beiden Viualiserungskomponenten in der Gui freigeschaltet und direkt angezeigt. Auch die entsprechenden ZOOM-Buttons und Centering-Buttons erscheinen nun, mit denen man die Anwendung bedienen kann. Danach wird auch die statistische Auswertung gestartet.
Die Anwendung last.forward hat ein spezielles User Interface Design. Dieses LookandFeel, aus der Bibliothek "Quaqua.jar", lässt die Oberfläche plattformübergreifend wie MacOSX aussehen und wird in der "main-Methode" gesetzt.

5.1.6 Help.java

Diese Klasse ist ebenfalls ein Menu-Item der Menu-Bar "Info". In der GUI wird dieses JPanel mit **getHelpPanel()** aufgerufen. Die Aufteilung der Elemente wird durch ein Border-Layout umgesetzt. Genauer:
North: Der Header mit Logo. (Null Layout)
South: Hilfe zu den Controls für TreeView und GraphView. (JTabbedPane mit Null Layout)

5.1.7 Profile.java

Hier wird das ProfilePanel erzeugt, welches die persönlichen Daten eines jeden Users aufnehmen kann. Hier gibt es für die Datensätze die speziellen Felder (ID,Name,Realname,…).
Weiterhin enthält dieses Panel einen Platzhalter für das jeweilige persönliche Foto das live aus dem Internet geladen wird. Somit erreichen wir ein hohe Aktualitätsrate bei den Bildern. Ein Button "Open Url" ist ebenfalls implementiert, welcher direkt auf die persönliche Website (auf Last.fm) des angezeigten Users führt.
In der GUI wird dieses JPanel mit **getProfilePanel()** aufgerufen und ist Bestandteil von **getJTabbedInfo().** Das Layout von Picture und User Data ist mit einem Grid Layout und die Datensätze mit einem Null Layout realisiert.

5.1.8 Searchfield.java

Hier wird das Suchfeld für die Anwendung realisiert. Das Suchfeld ist ein Teil der Tool-Bar und wird in der GUI mit **getSearchfield()** aufgerufen. Hier hängt ein ActionEventHandler an, der auf den RETURN-Button der Tastatur lauscht, um die Methode **performSearch()** der Klasse ActionEventHandler.java im gleichen Paket zu starten.
Das KeyEvent **VK_ENTER** löst nach erfolgreicher Suche einige Ereignisse aus. Zum Beispiel werden die Toolbar-Buttons (Zoom)sichtbar, die Funktionen von ProfilePanel und JFilterPanel werden "enabled" und der Start-Screen (**getJPanel_intro()**) wird gelöscht. Denn wenn noch keine Daten vorhanden sind würden diese Funktionen für den Benutzer keinen sichtbaren Erfolg bringen.

5.1.9 JFilterPanel.java

In diesem JPanel (Grid Layout) erzeugen wir die Filter-Kriterien für die Visualisierung. In der GUI wird dieses JPanel mit **getJFilterPanel()** aufgerufen und ist ebenso wie das ProfilePanel Bestandteil von **getJTabbedInfo().**
Zum Einen haben wir 3 Checkboxen (Female,Male,Undefined), die zu Anfang alle selektiert sind. Diese beziehen sich auf die GaphView-Darstellung und blenden dort Knoten je nach belieben aus und ein. Die Listener für diese Checkboxen befinden sich im Paket „de.lastfm.listener“ mit ihren zugehörigen KlassenNamen „ComboxListener“.
Weiterhin erzeugen wir 3 Listboxen für Playcount, Age, und Country. Hier kann später nach den jeweiligen Filter-Kriterien in der GraphView selektiert werden. Die Listener liegen auch hier im Paket „de.lastfm.listener“ mit ihren beschreibenden Klassennamen.

5.2 Package de.lastfm.listener

Dieses Package enthält folgende Klassen :

- Agecomboboxlistener.java
- CenteringControl.java
- ComboxListener.java
- CountryComboboxListener.java
- PlaycountComboBoxListener.java
- ZoomControl.java

5.2.1 Agecomboboxlistener.java

Dieser Listener hängt an der Auswahl der Altersgruppen im FilterPanel.
String a = "all";
String b = "0 - 20";
String c = "21 - 25";
String d = "25 - 30";
String f = "over 30";

Er vergleicht die Auswahl mit vorgegebenen Strings und filtert über die in den Klassen GraphPanel und TreeView vorhanden Knoten raus.

5.2.2 CenteringControl.java

Dieser Listener hängt am Centering-Button der Gui. Er ermöglicht die zentrierte Ansicht des Graphen oder des Baumes je nachdem in welcher Ansicht man sich befindet.

5.2.3 ComboxListener.java

Analog zu PlaycountComboBoxListener

5.2.4 CountryComboboxListener.java

Analog zu PlaycountComboBoxListener

5.2.5 PlaycountComboBoxListener.java

Dieser Listener hängt im ProfilPanel an Playcount ComboBox. Diese filtert Knoten aus den Graphen, auf die das ausgewählte Kriterium passen. Auch hier werden wieder Strings mit der Asuwhal verglichen und die jeweiligen Filter Methoden in den Klassen GraphPanel und TreeView gestartet.

5.2.6 ZoomControl.java

Dieser Listener besitzt 2 Methoden die in den jeweiligen ansichten das ZOOM IN und das ZOOM OUT erfüllt. Dieser Listener hängt an den beiden Zoom-Knöpfen der Gui.

5.3 Package de.lastfm.entity

Dieses Package enthält folgende Klassen :

- ProfilEntity.java

5.3.1 ProfileEntity.java

Klasse zum Anlegen eines Onbjektes vom Typ ProfilEntity. Diese Klasse wird gebraucht um aus der DB die Datensätze auszulesen.

5.4 Package de.lastfm.analysis

Dieses Package enthält folgende Klassen :

- JAnalysisPanel.java
- Pie3D.java

5.4.1 JAnalysisPanel.java

Hier werden die DefaultPieDataset() angelegt, die für die Konstruktion der Kuchendiagramme im StatsticPanel benötigt werden.

5.4.2 Pie3D.java

Hier werden die gefundenen Statistiken anhand von Kuchen visualisiert. Mehr dazu findet man unter www.jfreechart.org.

5.5 Package de.lastfm.db

Dieses Package enthält folgende Klassen :

- DB_Friends.java
- DB_User.java
- DB_Neighbour.java
- DB_Groups.java
- Dbconnector.java
- DBReader.java
- CountryComperator
- HSQLDBCreator

5.5.1 DB_Friends.java

5.5.2 DB_User.java

5.5.3 DB_Neighbour.java

5.5.4 DB_Groups.java

5.5.5 Dbconnector.java

5.5.6 DBReader.java

5.5.6 CountryComperator.java

Diese Klasse dient zur alphabetischen Sortierung des übergebenen Comperators. Dies wird in der Klasse ActionEventHandler in Zeile 289 genutzt zur Sortierung eines Vectors.

5.5.7 HSQLDBCreator.java

Diese Klasse dient zur Initialisierung der HSQLDB zu Beginn des Programmstartes. Hierbei wird im Paket sqlfile jenes File (lastfm.sql) zur Initialiserung der DB genutzt.
Hier werden nun die entsprechenden Tabellen und Werte angelegt. Sollte es einmal wünschenswert sein, die gesammelten Daten konsistent außerhalb der im Speicher laufenden DB zu halten, so sind auch hier bereits geeignete Methoden enthalten.
Sollte man eine konsistente Abspeicherung beispielsweise über ein DB-Datei von Hypersonics DB wünschen, so müssen die Parameter in der Klasse dbconnector auf:

```
private static final String URL = "jdbc:hsqldb:file:lastfm";
```

geändert werden. Zur Konfiguration einer MYSQL-DB dient das File lastfm_mysql.sql aus dem CVS-Repository.

5.6 Package de.lastfm.crawler

Dieses Package enthält folgende Klassen :

- Crawl.java
- Friends.java
- Neighbours.java
- Groups.java

5.6.1 Crawl.java

5.6.2 Friends.java

5.6.3 Neighbours.java

5.6.4 Groups.java

5.7 Package Images

In diesem Package sind alle in Last.forward vorkommenden Bilder gespeichert. Sie liegen alle im JPEG-Format vor und werden über entsprechende Methode in der JAR miteingelesen.

6 Klassendiagramm

7 Kontakt

Die Entwickler von Last.forward sind:

Christian Endlich
Email: endolino@users.sourceforge.net

Thomas Knoll
Email:knollson@users.sourceforge.net

Heike Scherer
Email: heike79@users.sourceforge.net

Supervisor:

Prof. Hendrik Speck
Kurs: Medienkonzeption und Produktion im Wintersemester 2006/2007
Web: http://www.egs.edu/faculty/speck.html
Fachhochschule Kaiserslautern
Standort Zweibrücken
Amerikastr. 1
66482 Zweibrücken
Web: http://www.fh-kl.de