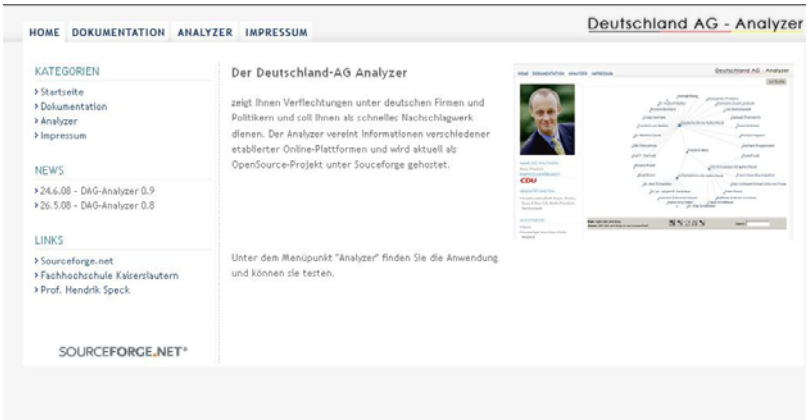
Fachhochschule Kaiserslautern

DIGITALE MEDIEN

— Medienkonzeption und Produktion —

Entwicklerdokumentation zur Studienarbeit
SS 2008



Entwicklerdokumentation

DAG-Analyzer

von

Nicolas Leyking, 856273
Kai Chamski, 855736

3. Juli 2008

Fachhochschule Kaiserslautern
Standort Zweibrücken
Fachbereich Informatik und Mikrosystemtechnik
Studiengang „Digitale Medien“
Betreuer: Prof. Hendrik Speck

Inhalt

1 Einleitung

1.1 Vorwort

Dieses Dokument beschreibt den Umgang und die Verwendung der Anwendung DAG-Analyzer und soll vor allem Programmierern als Grundlage zur Weiterentwicklung dienen. Es enthält sowohl allgemeine Informationen über die Anwendung, als auch detaillierte Informationen über die Softwarearchitektur, die einzelnen Packages und die einzelnen Klassen.

Es ist zudem sehr zu empfehlen zuerst das Benutzerhandbuch zu lesen sowie die Anwendung auszutesten damit die in dieser Dokumentation behandelten Themen besser verstanden werden.

1.2 Einleitung

Durch die bevorstehenden Wahlen stehen die Parteien sowie deren Vorsitzenden im Vordergrund. Die politisch aktiven Vorstandsmitglieder versuchen eine Politik zugunsten ihrer Firma zu betreiben und deren Interessen auf Bundesebene durchzusetzen. Für den wählenden Bürger ist es von großer Bedeutung mehr Informationen über diese und deren Nebeneinkünfte bzw. Arbeitsverhältnisse zu erhalten. Durch unsere Open Source Webapplikation möchten wir der Öffentlichkeit die Möglichkeit geben sich über die einzelnen Bundestagsabgeordneten und über ihre Nebeneinkünfte zu informieren.

1.3 DAG-Analyzer

Das Projekt DAG-Analyzer entstand im Rahmen der Veranstaltung Medienkonzeption und Produktion der Fachhochschule Kaiserslautern, Standort Zweibrücken unter der Leitung von Herrn Prof. Hendrik Speck. Ziel dieses Projektes ist es, die Zusammenhänge zwischen den einzelnen deutschen Unternehmen untereinander zu visualisieren und aufzuzeigen welcher Politiker in welcher Firma mitwirkt. Basierend auf einer Teilmenge der Firmendatenbank der Firma Hoppenstedt AG und den Offenlegungen der Nebeneinkünfte der Abgeordneten, ist es nun möglich das Netzwerk der ‚Deutschland AG' zu visualisieren.

Der Name DAG-Analyzer setzt sich aus den Abkürzungen "D" = Deutschland, "AG" = AG und "Analyzer" = Analyse zusammen. Der DAG-Analyzer ist eine webbasierte Anwendung die auf eine MySQL Datenbank zugreift und mit der Programmiersprache Java erstellt wurde. Zur Visualisierung wurde auf den Visualisierungsdienst von Many-Eyes zugegriffen (www.many-eyes.com).

Des Weiteren unterliegt der DAG-Analyzer auch einer Open Source Lizenz und wird auf dem Entwicklerportal Sourceforge.net (www.sourceforge.net) gehostet.

1.4 Lizenz und Haftungsausschluss

DAG-Analyzer wurde unter der GNU Library or General Public License (GPL) entwickelt. Die Anwendung behält sich vor weder vollkommen korrekt, noch auf dem aktuellsten Stand zu sein. Bei der Entwicklung wurde als Datengrundlage auf die Daten der Firma Hoppenstedt aus dem Jahre 2007, sowie die auf Spiegel Online veröffentlichten Politikergehälter zurückgegriffen.

1.5 Weiterentwicklung

Das Entwicklerteam von DAG-Analyzer bietet jedem Interessenten die Möglichkeit den Quellcode nach seinen eigenen Vorstellungen anzupassen und weiterzuentwickeln. Durch die sauber aufgebaute Struktur der Software im Gesamten und eine ausführliche Kommentierung des Quellcodes wird der Einstieg in die Weiterentwicklung erleichtert. Bei Detailfragen sind die beiden Entwickler Kai Chamski und Nicolas Leyking jederzeit per Mail erreichbar.

2 Anforderungen, Installation und Verwendung

2.1 Systemvorrausetzungen für die Entwicklung

Zur Weiterentwicklung des DAG-Analyzer benötigen Sie folgende Komponenten:

- Java Development Kit ab Version 5.0 der Java Standard Edition J2SE
- PHP Plug-in für Eclipse
- Lokaler Apache Webserver
- Lokale HSQLDB Datenbank (nur für den Crawler notwendig)
- Lokale MySQL Datenbank
- Java Entwicklungsumgebung (nur für den Crawler notwendig)
- PHP Entwicklungsumgebung (hier verwendet PHP plug-in für Eclipse)

Bezugsquellen:

- Java Entwicklungsumgebung: www.eclipse.org
- PHP Plug-in: www.TODO.org
- Java JDK: www.sun.com
- Lokale HSQLDB: ist schon direkt in dem Projekt integriert
- Lokaler Apache Webserver / MySQL DB: www.apachefriends.org/de/xampp.html

2.2 Installation für Entwickler

Zum einen gibt es die Möglichkeit, auf

http://sourceforge.net/projects/dag-analyzer/

das komplette Projekt über das CVS Repository auszuchecken oder auf der Website des Projektes,

http://dag-analyzer.sourceforge.net/doku.php

das komplette ZIP Archiv herunterzuladen.

Wichtig bei einem Checkout über CVS:
Aufgrund der schlechten Performance des Sourceforge CVS Servers wurden die Firmendaten der Firma Hoppenstedt bzw. die Spiegel Online Daten jeweils in ein ZIP Archiv gepackt.
Es macht einfach keinen Sinn über den langsamen Sourceforge CVS Server mehr als 3300 Dateien auszuchecken.

Wird also über CVS ein Checkout durchgeführt müssen die beiden ZIP Files nach dem Checkout entpackt werden. Erst dann ist der Crawlvorgang möglich.

2.3 Systemvoraussetzungen für die Nutzung

Für die reine Ausführung der Anwendung müssen folgende Softwareanforderungen erfüllt sein:

- Java Runtime Environment ab Version 5.0
- Internet Browser (JavaScript aktiviert)

2.4 Installation für die Nutzung

Um das in der Anwendung zur Visualisierung notwendige Applet starten zu können, müssen Sie Applets im Allgemeinen für Ihren Web Browser zulassen. Als Laufzeitumgebung müssen Sie die Java Runtime Environment JRE1.6 auf Ihrem Computer installieren (JRE 1.6 herunterladen, folgen Sie einfach den Installationsanweisungen der JRE 1.6).

2.5 Bedienung

Für Informationen zur Bedienung der Anwendung, schauen Sie bitte in die Benutzerdokumentation.

2.6 Programmiersprachen / Skriptsprachen

Die gesamte Anwendung wurde mit PHP 4.3.8 entwickelt da der Sourceforge Webservice lediglich über diese PHP Version verfügt. Auf Datenbank Seite wurde mit der über Sourceforge verfügbaren MySQL Datenbank in der Version 4.1.20 gearbeitet oder besser gesagt die Anwendung wurde für diese beiden Komponenten (also Sourceforge PHP Version und Sourceforge MySQL Version) entwickelt.

Für den Crawler wurde die Programmiersprache Java verwendet.

2.7 Hardwareanforderungen

Für das Ausführen der Anwendung werden folgende Hardwareanforderungen vorausgesetzt:

- Mindestens Pentium 4 mit 1,2 GHz
- Mindestens 512 MB RAM, empfohlen 1024 MB oder höher
- 50 MB freier Festplattenspeicher
- Eine Bildschirmauflösung von mindestens 1024 x 768 Pixel

Für das Ausführen des Crawlers bzw. die Entwicklung sind deutlich höhere Systemressourcen empfohlen:

- Mindestens Pentium 4 mit 2 GHz
- Mindestens 1024 MB RAM oder höher
- Bis zu 200 MB freier Festplattenspeicher

3 Datenbank

Als Datenbank wurde die von Sourceforge zur Verfügung gestellte MySQL DB in der Version 4.1.20 genutzt.

3.1 Datenbankstruktur

Die Datenbankstruktur ist folgendermaßen aufgebaut: Es gibt die zentrale Tabelle *Firma*, in der die grundlegenden Daten zu den Firmen, wie *Name*, *Ort*, *Kontakt*, *URL* usw. gespeichert werden. Da spezifische Daten wie *Aufsichtsräte*, *Vorstände* etc. mehrfach zu jeder Firma vorliegen, wurden für diese Daten 7 weitere Tabellen angelegt, in welchen diese Daten abgelegt und über die jeweilige *FirmenID* referenziert werden.

Folgende Tabellen wurden mit den gecrawlten Daten der Hoppenstedt Datenbank gefüllt:

- Aufsichtsrat
- Firma
- Vorstand
- Beschaeftigte
- Umsatz
- Konzernumsatz
- Beteiligungen
- Banken

Die Politikerdaten von Spiegel Online wurden in weiteren Tabellen abgelegt, auch hier existiert eine zentrale Tabelle *Politiker*, und zwei weitere Tabellen *Unternehmensfunktion* und *Ausschuesse*.

Somit ergeben sich die drei Tabellen:

- Politiker
- Unternehmensfunktion
- Ausschuesse

Eine weitere Tabelle namens *Firmtranslate* ist dafür zuständig die Firmendaten der Spiegel Online Seite eindeutig einer Firma der Hoppenstedt Daten zuzuordnen. Dies war leider notwendig, da die Firmennamen nicht immer übereinstimmen, obwohl die gleiche Firma gemeint ist

> Bsp.: Alte LeipzigerHallesche Lebensversicherung A.G. <> ALTE LEIPZIGER Holding Aktiengesellschaft

Dies ist auch die einzige Tabelle, welche händisch gepflegt werden muss. Zu guter letzt wurde noch eine View namens *fidzuordnung* angelegt um die Performance beim Crawlen minimal zu verbessern.

Damit ergeben sich die beiden letzten Tabellen / View

- Firmtranslate
- Fidzuordnung

Für die Erstellung und dem Füllen der Tabellen haben wir mehrere SQL Scripte erstellt:

- createTables.sql
- insertFirma.sql
- insertBanken.sql
- insertAufsichtsrat1.sql
- insertAufsichtsrat2.sql
- insertAufsichtsrat3.sql
- insertAufsichtsrat4.sql
- insertAufsichtsrat5.sql
- insertAufsichtsrat6.sql
- insertAufsichtsrat7.sql
- insertAufsichtsrat8.sql
- insertAufsichtsrat9.sql
- insertVorstand1.sql
- insertVorstand2.sql
- insertVorstand3.sql
- insertVorstand4.sql
- insertBeteiligungen1.sql
- insertBeteiligungen2.sql
- insertBeteiligungen3.sql
- insertUmsatz1.sql
- insertUmsatz2.sql
- insertUmsatz3.sql
- insertUmsatz4.sql
- insertUmsatz5.sql
- insertKonzernumsatz1.sql
- insertKonzernumsatz2.sql
- insertBeschaeftigte1.sql
- insertBeschaeftigte2.sql
- insertBeschaeftigte3.sql
- insertBeschaeftigte4.sql
- insertBeschaeftigte5.sql
- insertBeschaeftigte6.sql
- insertBeschaeftigte7.sql
- insertBeschaeftigte8.sql
- insertPolitiker.sql
- insertFirmtranslate.sql
- insertAusschuesse.sql
- insertUnternehmensfunktion.sql

Diese sollten auch in dieser Reihenfolge ausgeführt werden, damit evtl. auftretende Probleme vermieden werden.

Wichtiger Hinweis:
Da beim Ausführen der Skripte über den Sourceforge PHPmyAdmin massive Timeout Probleme auftraten, welche auf eine Überlastung des Sourceforge Servers zurückzuführen ist, mussten alle Skripte in extrem kleine Skripthäppchen aufgeteilt werden.
Diese kleinen Skripthäppchen gilt es nun einzeln der Reihe nach auszuführen.
Wird dies durchgeführt ist die Sourceforge MySQL DB komplett mit den nötigen Daten gefüllt.

3.2 Datenbank Design

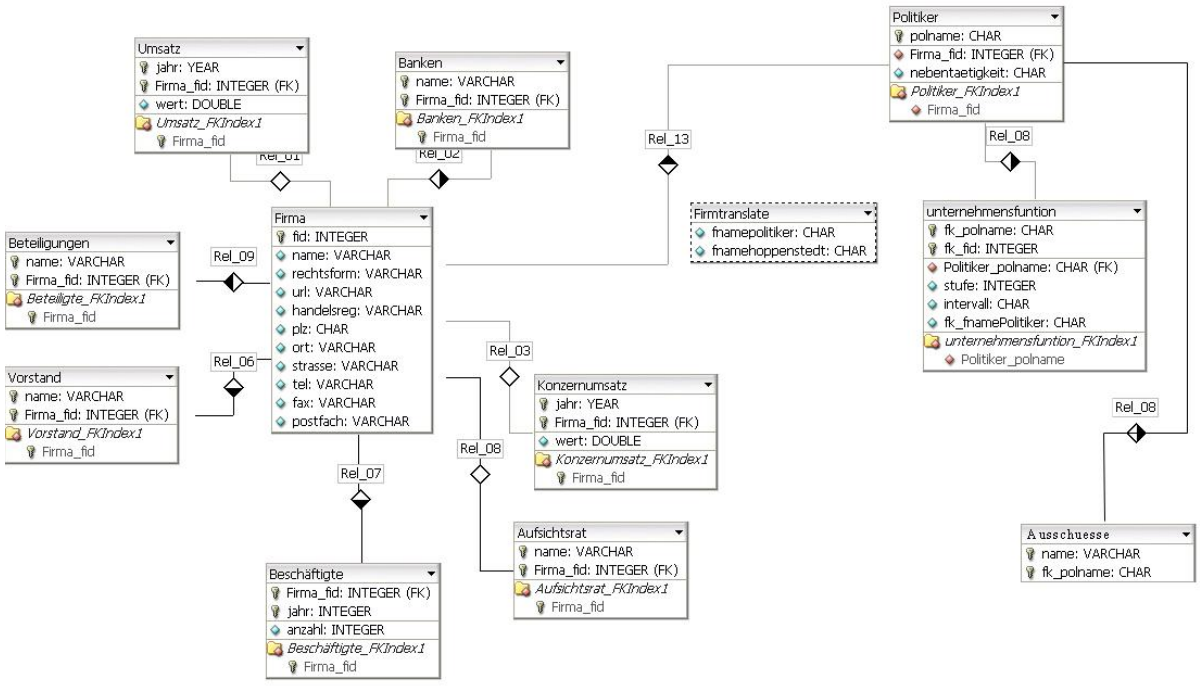


Bild 3-1:ER-Modell

3.3 Datenbank Creates

3.3.1 Tabelle Firma

```
CREATE TABLE `firma` (
`fid` INTEGER NOT NULL ,
`name` VARCHAR( 80 ) NOT NULL ,
`rechtsform` VARCHAR( 20 ),
`url` VARCHAR( 100 ),
`handelsreg` VARCHAR( 80 ),
`plz` VARCHAR( 5 ),
`ort` VARCHAR( 60 ),
`strasse` VARCHAR( 80 ),
`tel` VARCHAR( 25 ),
`fax` VARCHAR( 25 ),
`postfach` INTEGER,
PRIMARY KEY ( `fid` )
) ENGINE = MYISAM ;
```

3.3.2 Tabelle Banken

```
CREATE TABLE `banken` (
`banname` VARCHAR( 60 ) NOT NULL ,
`fk_fid` INTEGER NOT NULL ,
PRIMARY KEY ( `banname`, `fk_fid` ),
FOREIGN KEY ( `fk_fid` ) REFERENCES `firma` ( `fid` ) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE = MYISAM ;
```

3.3.3 Tabelle Aufsichtsrat

```
CREATE TABLE `aufsichtsrat` (
`aufname` VARCHAR( 80 ) NOT NULL ,
`fk_fid` INTEGER NOT NULL ,
PRIMARY KEY ( `aufname`, `fk_fid` ),
FOREIGN KEY ( `fk_fid` ) REFERENCES `firma` ( `fid` ) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE = MYISAM ;
```

3.3.4 Tabelle Vorstand

```
CREATE TABLE `vorstand` (
`vorname` VARCHAR( 80 ) NOT NULL ,
`fk_fid` INTEGER NOT NULL ,
PRIMARY KEY ( `vorname`, `fk_fid` ),
FOREIGN KEY ( `fk_fid` ) REFERENCES `firma` ( `fid` ) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE = MYISAM ;
```

3.3.5 Tabelle Beteiligungen

```
CREATE TABLE `beteiligungen` (
`betname` VARCHAR( 200 ) NOT NULL ,
`sitz` VARCHAR( 80 ) NOT NULL ,
`anteil` FLOAT,
`fk_fid` INTEGER NOT NULL ,
PRIMARY KEY ( `betname`, `fk_fid` , `sitz`),
FOREIGN KEY ( `fk_fid` ) REFERENCES `firma` ( `fid` ) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE = MYISAM ;
```

3.3.6 Tabelle Umsatz

```
CREATE TABLE `umsatz` (
`jahr` INTEGER NOT NULL ,
`wert` DOUBLE NOT NULL ,
`fk_fid` INTEGER NOT NULL ,
PRIMARY KEY ( `jahr`, `fk_fid`),
FOREIGN KEY ( `fk_fid` ) REFERENCES `firma` ( `fid` ) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE = MYISAM ;
```

3.3.7 Tabelle Konzernumsatz

```
CREATE TABLE `konzernumsatz` (
`jahr` INTEGER NOT NULL ,
`wert` DOUBLE NOT NULL ,
`fk_fid` INTEGER NOT NULL ,
PRIMARY KEY ( `jahr`, `fk_fid`),
FOREIGN KEY ( `fk_fid` ) REFERENCES `firma` ( `fid` ) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE = MYISAM ;
```

3.3.8 Tabelle Beschaeftigte

```
CREATE TABLE `beschaeftigte` (
`jahr` INTEGER NOT NULL ,
`anzahl` INTEGER NOT NULL ,
`fk_fid` INTEGER NOT NULL ,
PRIMARY KEY ( `jahr`, `fk_fid`),
FOREIGN KEY ( `fk_fid` ) REFERENCES `firma` ( `fid` ) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE = MYISAM ;
```

3.3.9 Tabelle Politiker

```
CREATE TABLE `politiker` (
`polname` VARCHAR( 80 ) NOT NULL ,
`nebentaetigkeit` VARCHAR( 3000 ) ,
PRIMARY KEY ( `polname` )
) ENGINE = MYISAM ;
```

3.3.10 Tabelle Firmtranslate

```
CREATE TABLE `firmtranslate` (
`fnamepolitiker` VARCHAR( 150 ) NOT NULL ,
`fnamehoppenstedt` VARCHAR( 150 ) ,
PRIMARY KEY ( `fnamepolitiker` )
) ENGINE = MYISAM ;
```

3.3.11 Tabelle Unternehmensfunktion

```
CREATE TABLE `unternehmensfunktion` (
`fk_polname` VARCHAR( 80 ) NOT NULL ,
`fk_fid` INTEGER NOT NULL ,
`stufe` INTEGER,
`intervall` VARCHAR( 50 ) ,
`fk_fnamepolitiker` VARCHAR( 150 ) NOT NULL ,
PRIMARY KEY ( `fk_fid`, `fk_polname` , `fk_fnamepolitiker`),
FOREIGN KEY ( `fk_polname` ) REFERENCES `politiker` ( `polname` ) ON DELETE CASCADE ON
UPDATE CASCADE,
FOREIGN KEY ( `fk_fid` ) REFERENCES `firma` ( `fid` ) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE = MYISAM ;
```

3.3.12 Tabelle Ausschuesse

```
DROP TABLE IF EXISTS `ausschuesse`;
CREATE TABLE IF NOT EXISTS `ausschuesse` (
  `fk_polname` varchar(80) NOT NULL,
  `name` varchar(100) NOT NULL,
  PRIMARY KEY  (`fk_polname`,`name`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

4 Packages - Klassenbeschreibung

4.1 Package Crawler

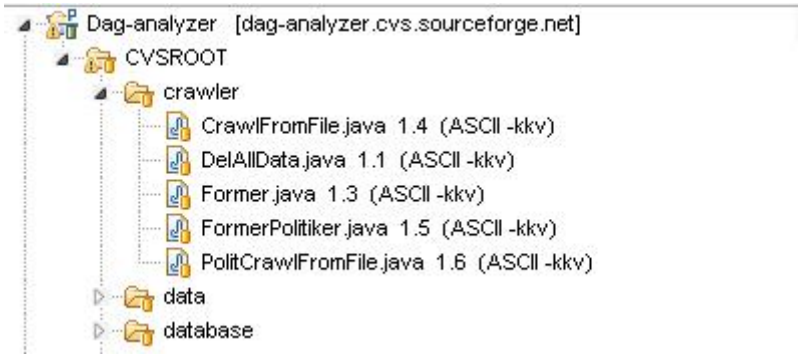


Bild 4-1:Package crawler

Das Package Crawler enthält folgende Klassen:

- CrawlFromFile.java
- DelAllData.java
- Former.java
- FormerPolitiker.java
- PolitCrawlFromFile.java

Die Klasse CrawlFromFile.java erlaubt es, alle unter *data* vorhandenen Hoppenstedt Firmenprofile, zu Crawlen und zu Parsen. Die Klasse parst den Inhalt und erzeugt DTO Objekte, die dann der DB Klasse übergeben werden. Die Dateinamen und Dateiendungen sind dabei egal. Es werden alle im Ordner befindlichen Dateien der Reihe nach abgearbeitet. Dabei enthält die Former.java alle Parser Methoden. Gleiches gilt für die Klasse PolitCrawlFromFile.java und FormerPolitiker.java allerdings werden hierzu die Profile der Politiker herangezogen.
Diese befinden sich unter *data/politiker*. Die Politikerdaten mussten auch lokal abgespeichert werden, da wir nicht sicher gehen konnten, wie lange diese online auf Spiegel Online zur Verfügung gestellt werden. Details zur Programmierung entnehmen Sie der JavaDoc, bzw. den Kommentaren im Quellcode.
Um die HSQL DB zu leeren wurde die Klasse *DelAllData.java* geschrieben. Zum leeren der HSQL DB einfach diese Klasse ausführen und die DB ist geleert.

Der Crawler für die Hoppenstedt Daten wird mit der Klasse *CrawlFromFile.java* gestartet.

Der Crawler für die Politiker Daten wird mit der Klasse *PolitCrawlFromFile.java* gestartet.

Hinweis HSQL DB:
Bei dieser Datenbank handelt es sich um eine im Hauptspeicher arbeitende Datenbank. Für den Betrieb sind lediglich die erforderliche *Librarie*, sowie die beiden unter *database* vorhandenen Files nötig. Selbstverständlich sind noch die nötigen DB- und ACCESS-Klassen notwendig aber dies ist ja Vorraussetzung bei der Kommunikation zwischen Crawler und Datenbank.
Wird ein Crawlvorgang gestartet und erfolgreich beendet wird, wird automatisch von der HSQL DB ein Insert-Skript erstellt damit diese DB bei einem Neustart auch wieder die Daten im Hauptspeicher zur Verfügung hat. Das Skript versteckt sich in der Datei *daganalyzer.script*. Wenn man es so sieht ist das Skript die eigentliche Datenbank. Es empfiehlt sich vor einem Crawlvorgang sich die Datei anzuschauen um dann nach einem erfolgreichen Crawlvorgang den Unterschied zu erkennen bzw. zu erkennen ob die DB erfolgreich vom Crawler gefüllt wurde oder nicht. War der Crawlvorgang erfolgreich sind die Daten welche gecrawled wurden in dieser *daganalyzer.script* File vorhanden. Das Script wird von der HSQL DB automatisch ausgeführt. Es ist somit keine manuelle Ausführung notwendig.

Wichtiger Hinweis:
Der Crawler ist leider noch nicht zu 100% funktionsfähig.

Folgende Einschränkungen existieren leider noch:

- Statt direkt die Sourceforge MySQL DB zu füllen wird eine lokale HSQL DB gefüllt
- Die Daten der Ausschuesse werden leider nicht fehlerfrei in die HSQL DB eingetragen

Durch diese Einschränkungen wurden für die Einrichtung der MySQL DB extra alle nötigen Skripte generiert welche unter dem Punkt 3.1 zu finden sind.

Grundlegender Ablauf des Crawlvorgangs:

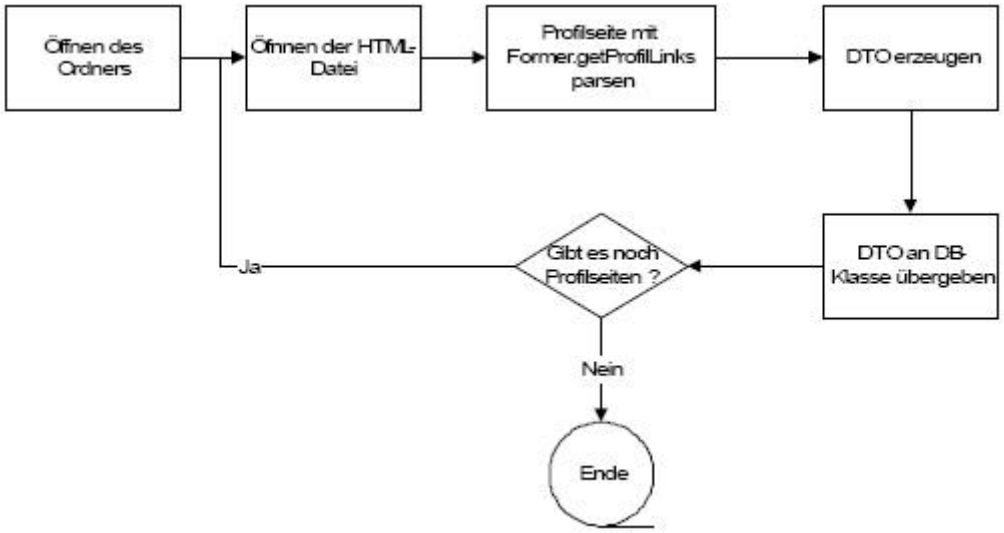


Bild 4-2:Ablauf Crawlvorgang

4.2 Package Datenbank

4.2.1 Package DB

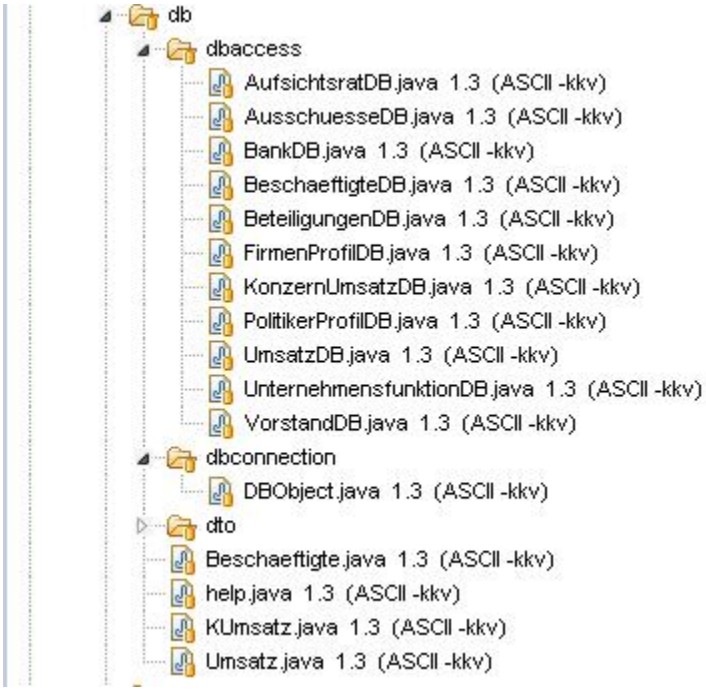


Bild 4-3:Package db

Unter *db* ist lediglich die Klasse *help.java* von Bedeutung. Diese unterstützt den Crawler bei seinem eigentlichen Crawlvorgang.

4.2.2 Package db.dbaccess

Zu jeder DTO Klasse gibt es auch eine DB Klasse. Diese DB Klasse stellt Methoden zur Verfügung, die die für die Anwendung wichtigen Datenbank Aktionen auf die dazugehörende Tabelle erledigen (z.B. finde alle, finde nach ID, finde nach Name, füge ein…) Die Abfrage Methoden liefern je nach Logik entweder ein DTO zurück oder eine Collection (java.util.LinkedList) mit mehreren DTOs zurück. Die Methode zum einfügen hat eine DTO als Parameter.

4.2.3 Package db.connection

Dieses Package enthält nur die Klasse *DBObject.java*, die über eine von ihr erzeugten Instanz eine Datenbankverbindung aufbaut, und diese über die Objektmethode *getConnection()* zurückgibt. Sie enthält 2 zentrale Methoden: *getConnection()* > liefert *SQLConnection* (java.sql.Connection) zurück *getDataSource()* > liefert eine Datasource auf die DB.

4.2.4 Package db.dto

In diesem Package befinden sich alle *DatabaseTransferObjects*. Diese Klassen repräsentieren jeweils eine Tabelle der Datenbankstruktur, d.h. jede Klasse hat die Attribute, der zugehörigen Tabelle.
Dazu enthält sie noch 3 Konstruktoren:

- Standard Konstruktor (erzeugt leeres Objekt)
- Konstruktor mit ResultSet als Parameter um aus einer selektierten Tabellenzeile ein Objekt zu erzeugen).
- Konstruktor mit Einzelparameter zum „manuellen“ erzeugen und initialisieren eines Objektes.

Ansonsten verfügen diese Klassen noch über die nach den Java Bean Konventionen üblichen *getter* und *setter* Methoden. Auch hier gilt, Details zur Programmierung entnehmen Sie der JavaDoc, bzw. den Kommentaren im Quellcode.

4.3 Package libs

Zwar nicht weiter erwähnenswert aber der Vollständigkeit halber hier aufgeführt ist das Package *libs*. Unter *libs* finden Sie alle benötigten jar Dateien, die zur Entwicklung unter dem Java Build Path eingebunden werden müssen.

4.4 Package util

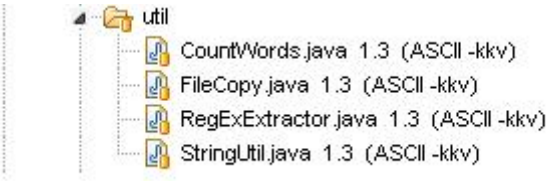


Bild 4-4:Package util

Auch im *util* Package befinden sich lediglich 4 Hilfsklassen, *CountWords.java* erlaubt es in einem sehr langen String die Häufigkeit bestimmter Wörter zu zählen. *FileCopy.java* enthält wie der Name schon sagt, eine Methode um eine Datei an einen bestimmten Platz zu kopieren. Die letzten beiden Klassen *RegExExtractor.java* und *StringUtil.java* werden benötigt um in den Regulären Ausdrücken, welche vor allem in den Formerklassen beim crawlen vorkommen, diverse Modifikationen vorzunehmen. Beispielsweise zum Ersetzen der HTML Tags.

4.5 PHP Anwendung

Die eigentliche Anwendung hat mit Java eigentlich gar nichts mehr zu tun. Das in der Anwendung befindliche Applet zu Visualisierung, wird nur über einen Applet Aufruf eingebunden und kommt von dem Visualisierungsdienst der Seite www.many-eyes.com. Von daher ist kein Java mehr nötig.
Zu dem Visualisierungsdienst gibt es noch folgendes zu erwähnen.
Bei diesem Dienst hat man die Möglichkeit Datensets hoch zu laden. Bei so einem Datenset handelt es sich um eine zweispaltige Tabelle. Diese Tabelle ist nicht mehr ein reines Text-File welches den Tabulator als Spaltentrenner benutzt. In der ersten Zeile muss noch angegeben werden wie die jeweilige Spalte heisst.

Da ein kleines Beispiel mehr erklärt wie reiner Text hier ein kleines Beispiel:

Parent	Child
CDU	Angela Merkel
SPD	Kurt Beck
SPD	Gerhard Schröder

Die Spalte Parent enthält die Partei. Die Spalte Child die dazugehörenden Politiker. Wird dieses Daten Set nun bei many-eyes hochgeladen und die entsprechende Vernetzungs-Visualisierung ausgewählt, bekommt man direkt die Visualisierung angezeigt und kann sich über Firefox den für das Applet entsprechenden Applet Aufruf heraus kopieren und für den DAG-Analyzer weiter verwenden.

Nochmals ein kleiner Überblick über die Vorgehensweise:

1. Datenset hochladen
2. Vernetzungs-Visualisierung auswählen
3. Im Seitenquelltext entsprechenden Applet Aufruf speichern

Wichtige Anmerkung:
Dieser Vorgang ist momentan noch manuell durchgeführt worden. Da der Dienst aber keine Captchas verwendet ist dies auch völlig automatisiert machbar. Es ist auch zwingend erforderlich dies zu automatisieren da es über 3000 Firmen sowie über 150 Politiker zu Visualisieren gibt.

Des Weiteren muss noch ein Algorithmus entwickelt werden, welcher die Vernetzung erstellt. Schließlich muss das ja in dem jeweiligen Datenset eingetragen werden.

Zudem ist es noch notwendig in der Datenbank ein Feld für den jeweiligen Visualisierungsaufruf einzubinden. Am besten es wird eine entsprechende Tabelle rein für diese Zuordnung erstellt mit dem Fremdschlüssel der Firma bzw. dem Fremdschlüssel des Politikers.

Leider waren die oben genannten Punkte während unseres Projektes aufgrund eines Zeitmangels nicht mehr realisierbar und es liegt in unserem Interessen dass diese Punkte noch von einem weiteren Entwicklerteam abgearbeitet werden.

Folgende Visualisierungen existieren bereits und können auch in der Anwendung live ausgetestet werden:

- Firma SAP AG
- Politiker Friedrich Merz
- Politiker Heinz Riesenhuber
- Firma IDS Scheer AG
- Firma Aareon AG
- Firma Deutsche Börse
- Aufsichtsratsmitglied Wilhelm Haarmann (nur über Firmen-Visu erreichbar bsp. *SAP AG*)
- Aufsichtsratsmitglied Erhard Schipporeit (nur über Firmen-Visu erreichbar bsp. *SAP AG*)
- Aufsichtsratsmitglied August-Wilhelm Scheer (nur über Firmen-Visu erreichbar bsp. *SAP AG*)

Die Datensets für diese Firmen sind im Ordner *stuff* für die Veranschaulichung ebenfalls verfügbar.

Für die Politiker haben wir als Grundlage zur Visualisierung, die Tätigkeiten in Firmen genommen sowie die in den betreffenden Firmen sitzenden Aufsichtsrats- bzw. Vorstandskollegen.

Bei den Firmen diente als Grundlage die Vernetzung der betreffenden Aufsichtsrats bzw. Vorstandsmitglieder mit anderen Firmen sowie deren betreffenden Aufsichtsrats- bzw. Vorstandskollegen.

Über die Firma *SAP AG* sind zudem noch die vernetzten Aufsichtsratsmitglieder *Erhard Schipporeit*, *August-Wilhelm Scheer* sowie *Wilhelm Haarmann* als Visualisierung verfügbar. Um zu dieser Visualisierung zu gelangen, muss zunächst in der Suchmaske nach der Firma *SAP* gesucht werden. Danach muss diese Firma angewählt werden. Im nächsten Schritt muss auf der linken Seite einer der betreffenden Mitglieder angeklickt werden um zu der entsprechenden Visualisierung zu gelangen.

Auch für diese 3 Mitglieder sind im *stuff* Ordner die dazugehörigen Datensets verfügbar.

4.5.1 Ordner php

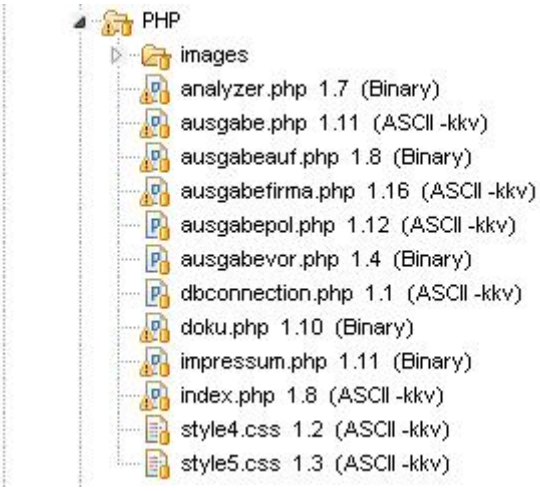


Bild 4-5:Ordner php

In diesem Ordner sind alle PHP Files vorhanden. Für die Detailfragen zu den entsprechenden PHP Files ist es am besten einen Blick in die betreffenden PHP Files zu werfen, da alle PHP Files mehr als ausreichend kommentiert wurden. Auf jedes File in dieser Dokumentation einzugehen würde die gesamte Dokumentation sehr unübersichtlich erscheinen lassen. Deshalb haben wir hier bewusst darauf verzichtet und alle wichtigen Informationen in die Sourcefiles in Form von Kommentaren eingebunden.

4.5.2 Ordner php/images

Der *images* Ordner enthält alle für die Anwendung sowie der Projekt Website benötigen Bilder.

4.5.3 Ordner php/images/politiker

In diesem Ordner sind alle Politiker Bilder die zur Anzeige auf der Seite benötigt werden.
Aus Rechtsgründen haben wir alle Firmen Logos entfernen müssen und haben an der entsprechenden Stelle nur einen Platzhalter stehen lassen.

4.5.4 Ordner data

In diesem Ordner befinden sich die Hoppenstedt Firmenprofile sowie in dem Unterordner *politiker* die einzelnen Profile der von Spiegel Online erfassten Politiker.

4.5.5 Ordner stuff

Der *stuff* Ordner enthält die am Anfang erwähnten Skripthäppchen, das create-Skript für die Tabellen sowie die Datensets der oben erwähnten Politikern bzw. Firmen.

5 Screenshots

5.1 Hinweis

Da es sich um eine Webanwendung handelt wurde die Chance wahrgenommen die eigentliche Anwendung in die *Sourceforge Project Website,* als einen Unterpunkt zu integrieren.
Die Project Website ist deshalb wie folgt aufgebaut.

5.2 Home-Bereich

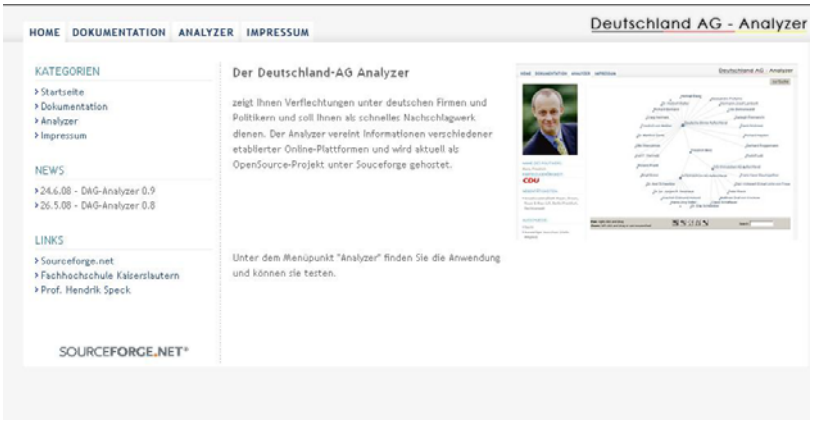


Bild 5-1:Home-Bereich

5.3 Dokumentations-Bereich

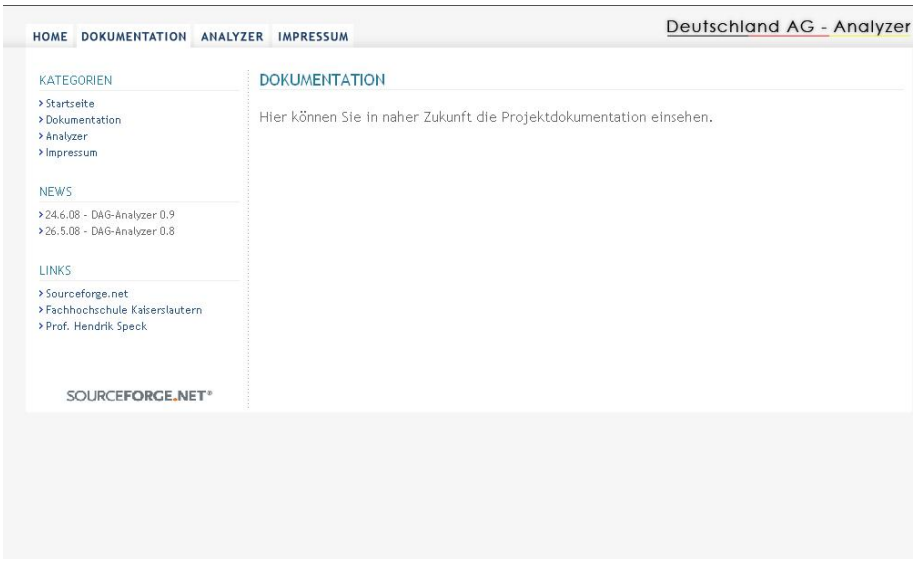


Bild 5-2:Dokumentations-Bereich

5.4 DAG-Analyzer

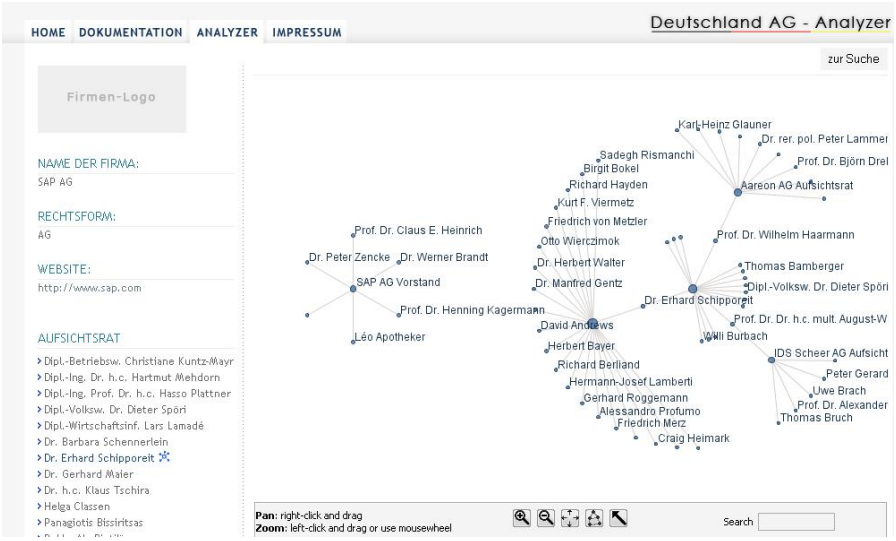


Bild 5-3:Firmen-Visualisierung

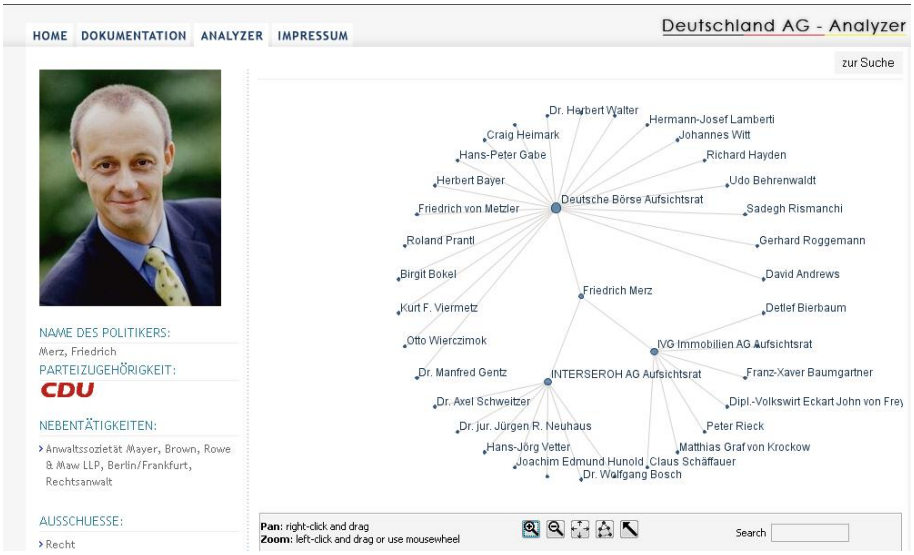


Bild 5-4:Politiker-Visualisierung

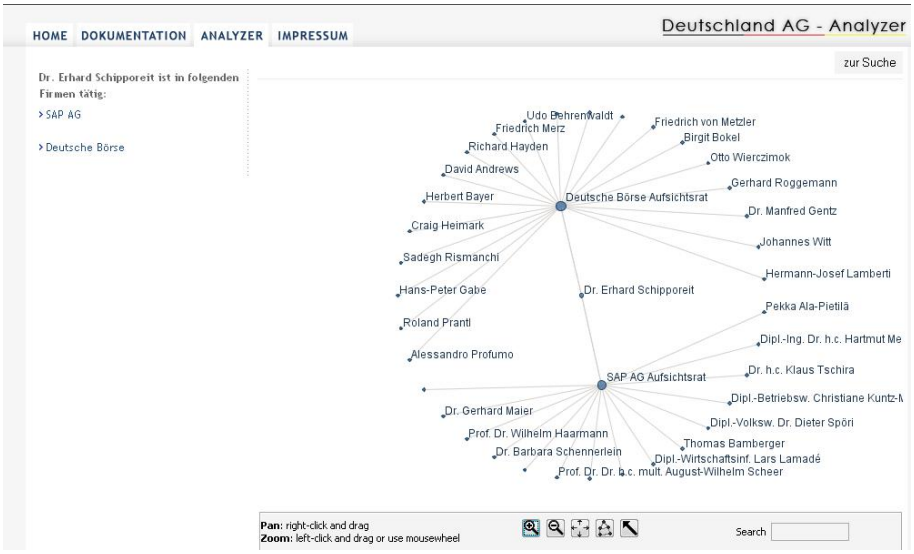


Bild 5-5:Vernetzung Aufsichtsrat

5.5 Impressum

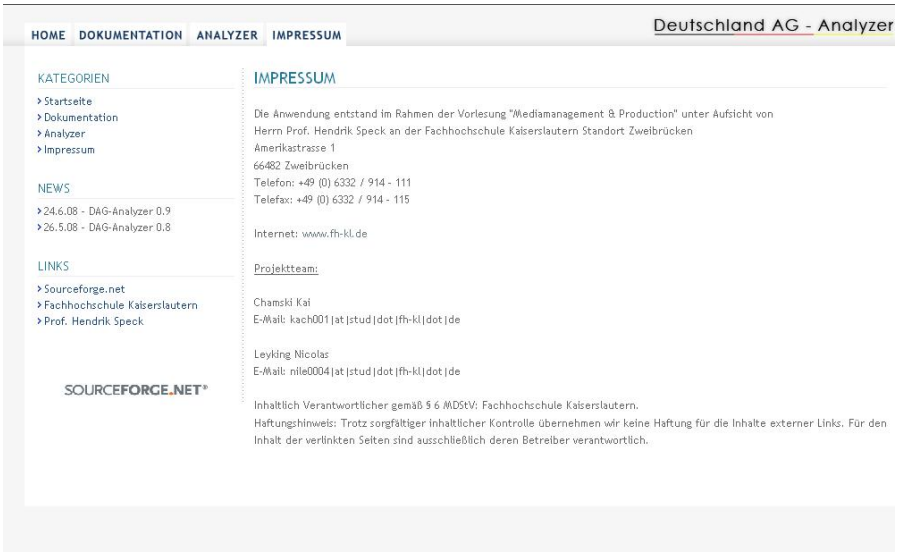


Bild 5-6:Impressum

6 Kontakt

6.1 Die Entwickler

Die Anwendung DAG-Analyzer wurde entworfen und entwickelt von:

- Kai Chamski, kach0001@stud.fh-kl.de
- Nicolas Leyking, nile0004@stud.fh-kl.de

Im Rahmen des Studienfachs *Medienkonzeption und Produktion* im Sommersemester 2008 an der Fachhochschule Kaiserslautern, Standort Zweibrücken.

6.2 Betreuender Professor

Prof. Hendrik Speck

www.hendrikspeck.com