Crawling von Datenschutzerklärung-Historien

Alexander Prull, Jörn-Henning Daug, Simon Kaleschke

05. Februar 2017

1 Projektbeschreibung

Das Praktikumsthema "Crawling von Datenschutzerklärung-Historien" beschäftigt sich mit den Datenschutzerklärungen ausgewählter Websites. Die Websites sollen regelmäßig auf neue Versionen geprüft werden. Neue Versionen sammelt der Crawler in einer Datenbank. Die verschiedenen Datenschutzerklärungen eines Unternehmens werden dann auf einer Website gegenübergestellt und Veränderungen sichtbar gemacht.

2 Lösungsansatz

Die Software ist in drei Teile aufgeteilt: Frontend, Backend und Extraktion. Das Frontend übernimmt dabei die grafische Oberfläche des Projekts. Die ausgewählten Versionen der Datenschutzerklärungen sind hier gegenübergestellt. Ein Diff-Tool macht die Veränderungen erkennbar. Die Extraktion sammelt durch einen Crawler die Datenschutzerklärungen der Unternehmen ein und legt sie in die ausgewiesene Datenbank. Das Backend übernimmt die Kommunikation zwischen Datenbank und Frontend und startet in regelmäßigen Abständen die Extraktion.

3 Softwarearchitektur

Die einzelnen Komponenten sind jeweils in einem Ordner abgelegt. Im Ordner *script* liegen die Dateien für das Crawling und die Datenbank, *tmc_frontend* enthält das Frontend und *tmc_ backend* dementsprechend das Backend. Im folgenden werden die einzelnen Komponenten des Projekts genauer beschrieben.

3.1 Frontend

Die Weboberfläche zur Darstellung und zum Vergleich der Datenschutzerklärungen bildet das Frontend des Projektes. Für die Darstellung wurde ein Onepager erstellt, der aus drei Elementen besteht. Eine ausfahrbare Sidebar auf der linken Seite dient zur Auswahl der gewünschten Website. Im oberen Bereich der Seite zeigt ein Zeitstrahl anschließend alle verfügbaren Datenschutzerklärungen anhand ihrer Eintragungsdaten an. Darunter befinden sich zwei Textfelder.
Durch Klicken auf den gewünschten Eintrag im Zeitstrahl wird der Text als Referenz in das linke Feld geladen. Wählt man nun ein zweites Datum aus, startet das Diff-Tool und zeigt im rechten Feld die Unterschiede der beiden Datenschutzerklärungen an. Dabei steht rot für „nicht mehr vorhanden“, grün für „neu hinzugekommen“ und grau für „unverändert“. Das Frontend baut sich aus *bower*-Komponenten auf und wird durch *bootstrap* optisch optimiert. Die Funktionalität im Frontend wird durch *AngularJS* unterstützt. Anfragen über zwei REST-Schnittstellen an das Backend importieren die nötigen Informationen.

3.2 Backend

Das Backend ist ein Maven-Projekt und besteht aus vier Paketen: *server_ core*, *resources*, *representation* und *tools*.
Zentraler Startpunkt ist der *server_ core*. In diesem Paket befindet sich die Hauptklasse *Start.java*. Sie startet zwei Threads für den Server und den Crawler. Der Server ist nach dem Start unter der dort hinterlegten Server-IP und dem Port 8080 erreichbar und wartet auf Anfragen des Frontends. Der Crawler startet die Extraktion (*crawl.rb*) regelmäßig. In der Voreinstellung einmal am Tag.
Wenn das Frontend eine Anfrage stellt, wird das Paket *tools* aktiv. In diesem befindet sich die Datenbankinitialisierung, sowie die Klassen für die SQL-Anfragen zum Ausliefern eines gewünschten Textes einer Datenschutzerklärung(*TextLoader.java*) und aller Datumswerte der abgespeicherten Versionen (*DateLoader.java*) eines Unternehmens.
Das Paket *resources* wandelt die Ergebnisse aus dem *tools*-Paket in JSONs um und ist zuständig für den Aufbau einer REST-Schnittstelle auf einer URL und die Auslieferung der JSONs an diese Website.
Die Klassen aus dem letzten Paket *representation* bauen die Objekte auf, die an die URLs übermittelt werden. Das Objekt *Texts* besteht aus dem Text der Datenschutzerklärung und dem Datum der Veröffentlichung. Das Objekt *Date* enthält eine ID, einen Datumswert für eine bessere Verarbeitung im Code (*SYSTEM_ DATE*) und einen Datumswert für eine bessere Lesbarkeit (*DIS-*

PLAY_DATE).

Das Klassendiagramm zum Backend ist als Abbildung 2 im Anhang hinterlegt.

3.3 Extraktion

Die Ruby-Struktur zur Extraktion der Datenschutzhistorien funktioniert folgendermaßen:
Vom Backend aus wird *crawl.rb* aufgerufen. Dieses Skript beinhaltet alle Einstellungen zur Extraktion von jeder Webseite. Es sendet diese Einstellungen an *PrivacyExtractor.rb*, das die Extraktion dann vollzieht. Hierfür bezieht es Links zu älteren Versionen entweder vom Archiv der Webseite selbst, oder mit *ArchiveExtractor.rb* von dem Internetarchiv *archive.org*. Der Inhalt jeder Seite wird extrahiert, auf die notwendigen Elemente zugeschnitten und formatiert. Zuletzt werden mit *PrivacyDiffer.rb* die extrahierten Texte auf Unterschiede untersucht und gegebenenfalls in einer Datenbank abgespeichert. Die Datumswerte und Texte aus der fertigen Datenbank können vom Frontend abgefragt werden.
Weitere Informationen sind auf der Abbildung 2 auf Seite 8 zu finden.

4 Bedienungsanleitung

Die folgende Bedienungsanleitung enthält einen kurzen Einblick über die verwendeten Software-Komponenten, die Einrichtung und Start des Servers, sowie eine Erklärung des Crawling-Prozess.

4.1 Vorbereitung

Damit die Software reibungslos läuft, werden folgende vorinstallierte Pakete erwartet:
Für das Frontend:

- NodeJS(Npm/Bower)
- Bootstrap
- vis
- diff (Npm) bzw. jsdiff (Bower)
- jQuery

Für das Backend:

- Java

- SQLite3
- Maven

Für die Extraktion:

- Ruby 2.3.1 oder höher.
- Die Ruby-Gems *date, open-uri, open_uri_redirections, nokogiri, rubygems, sqlite3, json, differ.*

Tragen sie außerdem in der Datei *config.ppp* den von Ihnen gewünschten Pfad für die Ergebnisdatenbank ein. (Standard ist *∼/html/script/policies.db*)

4.2 Server einrichten

Damit das Backend die JSONs für das Frontend bereitstellen kann, muss in der Klasse *ServerStarter.java* im Paket *server_core* die Server-IP eingetragen werden. Um die Uhrzeit zu ändern, an dem der Crawling-Prozess gestartet wird, müssen im selben Paket in der Klasse *CrawlerStarter.java* die entsprechenden Variablen abgeändert werden.
Zum Erstellen einer *jar* muss im Ordner *tmc_backend* der Befehl *mvn clean package* erfolgen. Eine fertige *jar* liegt bereits im Unterordner *target.*

4.3 Serverstart und allgemeiner Programmablauf

Gestartet wird der Backend-Server mit der *Start.java*-Klasse durch die *jar* im Ordner *target.* Diese Klasse ist als Hauptklasse in der *pom.xml* des Maven-Projekts hinterlegt. Danach starten sich zwei Threads für den Server und den Crawler. Der Server steht danach für Anfragen aus dem Frontend bereit. Ohne den Start der *jar* kann das Frontend nicht auf die Datenbank zugreifen.
Der Crawling-Prozess wird einmal täglich durchgeführt. Zum Beenden bzw. Deaktivieren des Crawlers muss *Enter* gedrückt werden. Durch wiederholtes Drücken der *Enter*-Taste wird auch der Server beendet.

4.4 Webseite einzeln crawlen

Normalweise wird der Aufruf zur Extraktion vom Backend aus gestartet. Wollen Sie jedoch eine einzelne Webseite crawlen, gehen sie folgendermaßen vor:

1. Versichern Sie sich, dass die Webseite und zugehörige Optionen im Quellcode vermerkt sind (siehe dazu Abschnitt 7.2).

2. Sind noch keine Informationen zu der Webseite vorhanden oder wollen Sie alle Informationen von null an neu generieren lassen, wechseln sie in den Ordner *script* und starten sie den Aufruf *ruby crawl.rb webseite fetch*. Wollen sie bspw. alle Informationen für Google neu generieren lassen, ersetzen sie im Aufruf *webseite* durch *google*.

3. Wollen Sie nun überprüfen, ob eine neue Version verfügbar ist, starten Sie den Aufruf *ruby crawl.rb webseite update*.

4.5 Datenbankstruktur

Die SQLite-Datenbank hat folgende Spalten:

Spalte	**Bedeutung**
ID	Primärschlüssel.
SYSTEM_DATE	Datum, an dem die Datenschutzhistorie in dem aktuellen Zustand war.
DISPLAY_DATE	Gleiches Datum in benutzerfreundlicher Schreibweise.
LINK	Hyperlink, wo die extrahierte Historie zu finden ist.
CONTENT	Der extrahierte Plaintext.

5 Ergebnisse

Das Projekt ist in der Lage eigenständig und dauerhaft 20 Websites zu crawlen und dadurch die Datenbank aktuell zu halten. Eine Liste aller gecrawlten Webseiten ist auf Abbildung 3 auf Seite 9 zu finden.
Die Weboberfläche besitzt eine Sidebar, einen Zeitstrahl und zwei Textfelder. Die Datenschutzerklärungen der 20 gecrawlten Websites sind über die jeweiligen Reiter der Sidebar aufrufbar. Auf dem Zeitstrahl entsteht dann eine Übersicht, wann die Website ihre Datenschutzerklärung aktualisiert hat. Wählt man ein Datum aus, erscheint die Datenschutzerklärung als Text und kann mit einem anderen Zeitpunkt über das Diff-Tool verglichen werden.

6 Einschränkungen

Das Projekt ist auf 20 Websites angesetzt worden. Da die Strukturen der Websites immer unterschiedlich sind, muss auch der Crawler darauf abgestimmt sein. Weitere Websites müssen also jedes Mal neu untersucht und die Crawling-Routine programmiert werden.

Sollte sich in Zukunft die Struktur einer der 20 Websites ändern, so funktioniert der Crawling-Prozess nicht mehr. Eine Wartung des Crawlers muss einkalkuliert werden.

7 Erweiterungsmöglichkeiten

7.1 Daten weiterverarbeiten

Die gewonnenen Daten in der Datenbank können unabhängig von der Webseite vielseitig weiterverwendet werden. So wäre bspw. die Anbindung an weitere Anwendungsprogramme denkbar, z.B. zur Bewertung der Datenqualität oder zur Suche nach Schlüsselwörtern.

7.2 Webseite hinzufügen

Wenn Sie eine Webseite zum Tool hinzufügen möchten, so brauchen Sie i.A. den Namen der Webseite und einen Hyperlink zur Webseite, auf der die Datenschutzerklärung zu finden ist. Wie weiter vorzugehen ist, wird im folgenden beschrieben:

7.2.1 Frontend

Alle Strukturen zum Aufbau des Frontends sind durch AngularJS weitestgehend dynamisch und auf Bedarf anpassbar. Die Reiter in der Sidebar passen sich der Daten aus der Restschnittstelle an und alle weiteren Websiten sollten ohne Probleme integriert werden.

7.2.2 Extraktion

Eine neue Website ändert nichts an der Struktur des Backends. Es muss aber die Extraktion bearbeitet werden. Fügen Sie in *crawl.rb* den Namen der Webseite, den Link, den Namen der Tabelle, in der die Informationen gespeichert werden sollen, wie die HTML-Struktur zu modifizieren ist, etc. ein. Näheres wird mit Beispielen im Skript selbst erläutert.

7.3 Weitere Daten aus Datenbank abfragen

Wenn dem Frontend weitere JSONs zur Verfügung gestellt werden sollen, gehen Sie wie folgt vor: Erstellen Sie eine neue Klasse für ein Objekt im Paket *representation*. Danach formulieren Sie eine SQL-Anfrage in einer Klasse des Paketes *tools*. Außerdem müssen dort die Ergebnisse das Objekt aus *representation* befüllen und in einer *List<t>* gesammelt werden. Zum Schluss erstellen Sie im Paket *resources* eine neue Ressource, an die die *List<t>* als JSON ausgeliefert wird.

8 Anhang

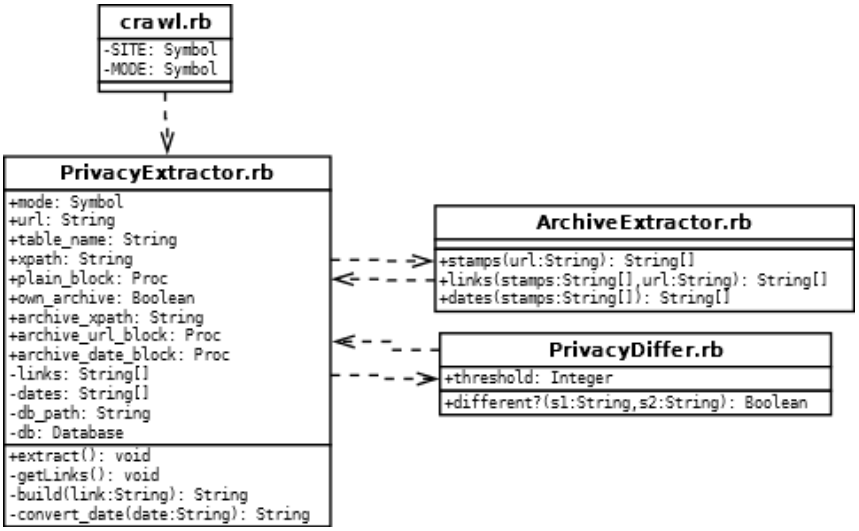

Abbildung 1: Architektur Extraktion

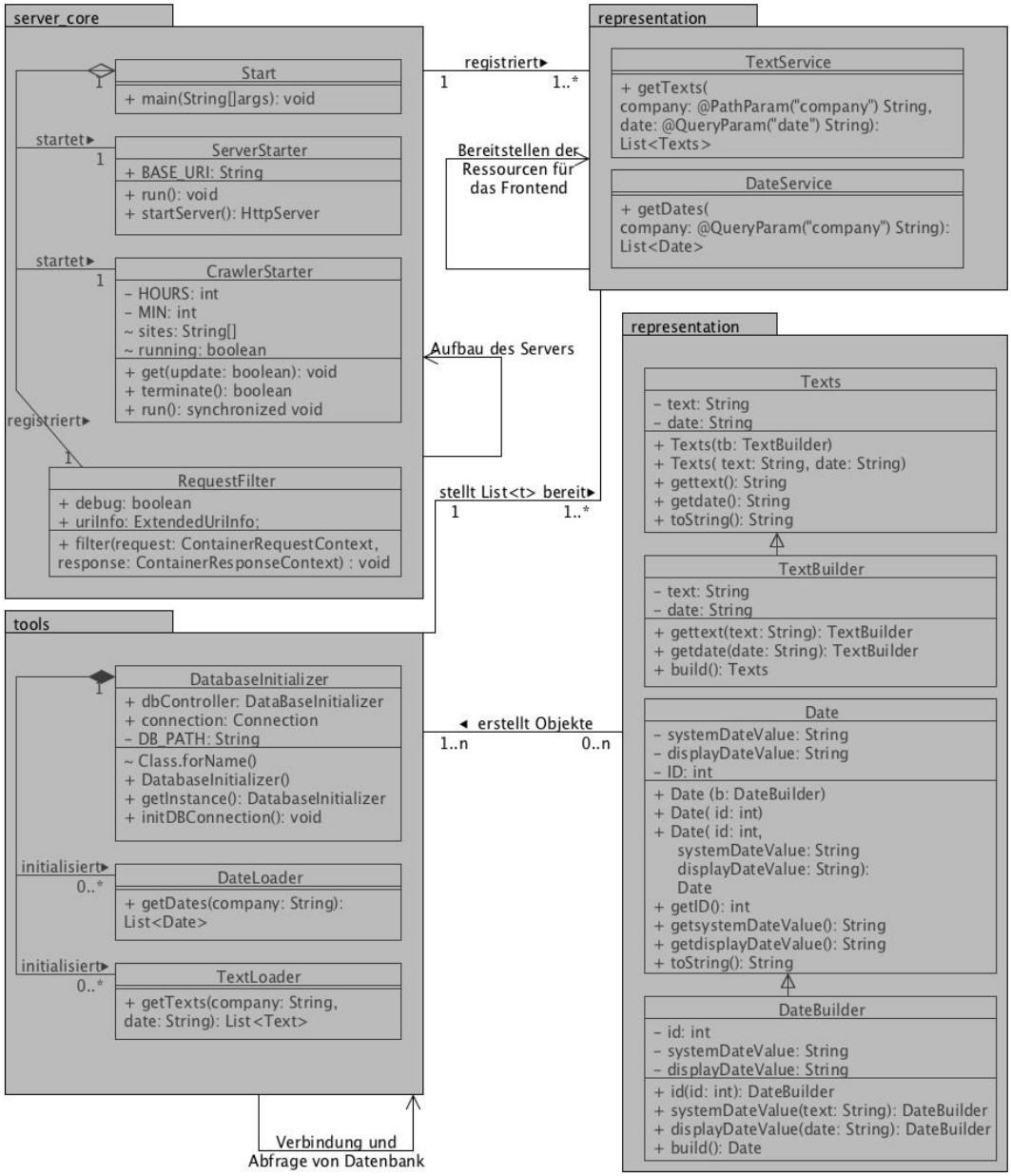

Abbildung 2: Architektur Backend

Firma	Eigenes Archiv	Versionen	Zeitspanne	Qualität
Alternate	✗	7	08/2014 - 07/2016	0.9
Amorelie	✗	11	01/2013 - 10/2016	0.7
Apple	✗	12	09/2014 - 09/2016	1.0
Burgerking	✗	2	02/2015 - 12/2016	0.7
Edeka	✗	4	11/2014 - 10/2016	0.7
Google	✓	22	06/1999 - 01/2017	0.8
Microsoft	✗	7	02/2016 - 01/2017	0.9
Payback	✗	10	09/2011 - 10/2016	0.9
Paypal	✗	2	04/2014 - 11/2016	0.9
RocketbeansTV	✗	4	10/2014 - 09/2016	1.0
Steam	✗	3	09/2012 - 11/2016	0.8
Subway	✗	3	05/2016 - 10/2016	0.9
Süddeutsche	✗	1	04/2015 - 04/2015	0.2
Trivago	✗	1	04/2016 - 04/2016	0.7
Twitter	✓	10	05/2007 - 01/2016	0.8
Uni Leipzig	✗	1	06/2013 - 06/2013	0.5
Vine	✗	5	03/2013 - 01/2017	1.0
WhatsApp	✓	2	07/2012 - 01/2017	0.9
Wikimedia	✓	4	06/2006 - 06/2014	1.0
Zalando	✗	3	09/2010 - 11/2012	0.9

Abbildung 3: Übersicht gecrawlte Webseiten