Recherche zu RSS und Möglichkeiten zu Verarbeitung

# RSS allgemein

RSS war ursprünglich eine Abkürzung für „Rich Site Summary“, wird aber heute als „Really Simple Syndication“ ausgeschrieben. Es handelt sich dabei um ein Dateiformat für Webfeeds, die wiederrum Informationen über Änderungen von Webseiten beinhalten.

Abgerufen werden können RSS-Feeds wie andere Webseiten über HTTP (Hypertext Transfer Protocol). Im Gegensatz zu „normalen“ Webseiten enthalten diese allerdings keinen HTML-Code (Hypertext Markup Language), sondern XML-Code (Extensible Markup Language). Diese Sprachen sind relativ ähnlich, allerdings ist der Zweck von XML nicht wie bei HTML die Beschreibung von Webinhalten, sondern die baumartige Strukturierung von textbasierten Informationen.

# Aufbau

Für Newsfeeds gibt es drei wesentliche Standards, diese heißen RSS 2.0, RDF (Ressource Description Framework) und Atom. Diese Standards ähneln sich im Aufbau und den Darstellbaren Informationen sehr stark. Die Formate RSS 2.0 und RDF benutzen darüber hinaus für die meisten Felder gleiche Tagnamen, wodurch sie in vielen Fällen austauschbar sind. Aufgrund dieser Ähnlichkeiten wird auf eine getrennte Beschreibung der genannten Formate verzichtet. Stattdessen folgt eine Beschreibung für RSS 2.0, wobei Abweichungen für die anderen beiden Standards ergänzt werden.

Grundsätzlich gliedert sich der Aufbau in einen allgemeinen Teil, der den Newsfeed beschreibt und darin enthalten beliebig viele Abschnitte mit Informationen über die bereitgestellten Einträge.

Grundkenntnisse der Sprache XML werden im folgenden Abschnitt vorausgesetzt. Insbesondere sei an dieser Stelle darauf hingewiesen, dass alle Elemente mit entsprechenden Abschlusstags (</tagname>) abgeschlossen werden müssen.

## Kanal

Eine .rss-Datei beginnt immer mit mindestens einer Verarbeitungsanweisung für den XML-Parser, die den XML-Standard und den Zeichensatz festlegt. Außerdem können an dieser Stelle weitere Informationen, wie zum Beispiel zu verwendende Stylesheets folgen.

Das Wurzelelement einer RSS-Datei heißt <rss> und hat Parameter zur Kennzeichnung der Version. Darin enthalten ist ein Unterelement mit dem Namen <channel>. Bei Atom gibt es stattdessen nur ein einzelnes Wurzelelement mit dem Namen <feed>.

Ein gültiger Channel muss die Unterelemente <title>, <link> und <description> jeweils genau einmal enthalten. Darüber hinaus gibt es noch eine Vielzahl von optionalen Elementen, besonders erwähnenswert ist das Element <ttl>.

<title> Dieses Element enthält den Titel des Feeds.

<link> Dieses Element enthält die URL der zum Feed gehörenden Webseite.

<description> Dieses Element enthält eine kurze Beschreibung zum Kanal.

<ttl> Abkürzung für „Time to live“. Dieses Element enthält eine Empfehlung, nach   
 welcher Zeitspanne in Minuten der Feed aktualisiert werden sollte.

## Einträge

Ein Kanal kann beliebig viele Elemente des Typs <item> (bei Atom <entry>) enthalten. Diese Elemente repräsentieren die Einträge des Feeds. Jeder Eintrag kann wiederrum durch eine Reihe von Untereinträgen näher beschrieben werden. Für Einträge sind alle Elemente optional, allerdings muss mindestens ein Titel oder eine Beschreibung vorhanden sein. Auch an dieser Stelle sollen die wichtigsten Eigenschaften aufgelistet werden.

<title> Titel des Eintrags

<description> Kurze Zusammenfassung des Inhalts (heißt bei Atom <summary>)

<link> URL, unter der der vollständige Artikel zu finden ist.

<pubDate> Publikationsdatum (heißt bei Atom <updated>)

## Beispiel

  
Der (manuell gekürzte) Quelltext eines RSS-Feeds der Tagesschau.

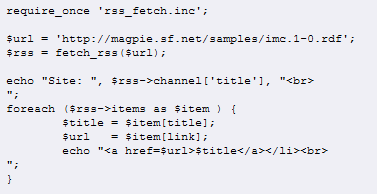
# Tools zur Verarbeitung

Die Projektaufgabe beinhaltet das Einlesen von RSS-Feeds, das Extrahieren, Filtern und Zusammenführen von darin enthaltenen Informationen. Für einige dieser Teilaufgaben lassen sich bereits vorhandene Tools einsetzen. Eine Auswahl dieser Tools soll an dieser Stelle kurz vorgestellt werden.

## MagpieRSS

Dokumentation und Download: <https://sourceforge.net/projects/magpierss/files/>

MagpieRSS ist eine PHP-Library, die Funktionen zum Anfordern, Zwischenspeichern und Auswerten von RSS-Dateien zur Verfügung stellt. Die Ausgabe der extrahierten Informationen liegt dabei in der Hand des Web-Entwicklers, somit hat dieser freie Hand bei der Auswahl und Formatierung der angezeigten Informationen. Da es sich hierbei um eine PHP-Library handelt, eignet sich MagpieRSS vor allem zur Ausgabe einer RSS-Datei auf einer Weboberfläche.

  
Ein einfaches PHP-Beispielskript zur Auswertung und Anzeige von RSS-Feeds.   
Dieses Script gibt eine Reihe von Links aus, die den Titel jedes Feedeintrags als Text enthalten und auf die zugehörigen URLs verweisen.

## ROME

Dokumentation und Download: <https://rometools.github.io/rome/index.html>

ROME ist ein Java-Framework für RSS und Atom Feeds. Mithilfe dieses Frameworks lassen sich RSS-Dateien anfordern, zwischenspeichern, auswerten und neu erstellen. Es stellt einer Java-Anwendung unter anderem die formatunabhängigen Klassen SyndFeed und SyndFeedEntry zur Verfügung, die sich zur internen Verarbeitung der Daten eigen. Die Möglichkeit, RSS-Feeds nicht nur in PHP-Skripten, sondern auch in eigenen Serveranwendungen zu verarbeiten, bringt einige Vorteile mit sich:

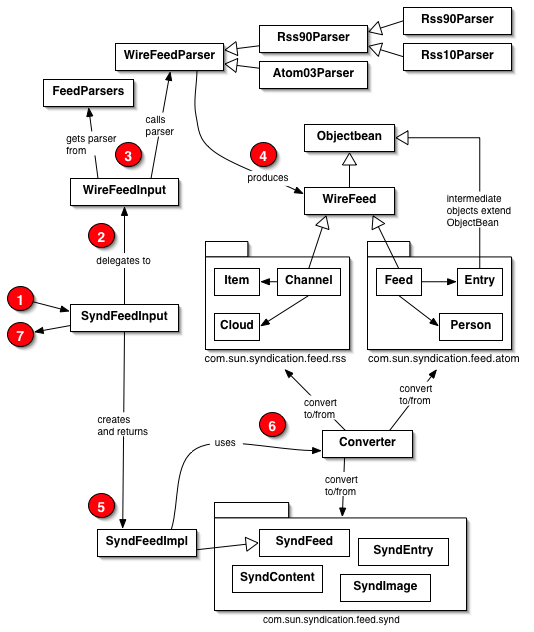
* Das PHP-Skript kann relativ einfach gehalten werden, da es sich nicht mit dem Einholen und Filtern von Informationen aus externen Quellen befassen muss. Die Komplexität des Projekts wird auf diese Weise besser verteilt.
* Die Webanwendung verhält sich performanter, wenn die darzustellenden Informationen zum Zeitpunkt des Abrufs bereits zur Verfügung stehen und nicht bei jeder Aktualisierung neu ermittelt werden müssen. Dies ist möglich, wenn diese Informationen in einer von PHP unabhängigen Anwendung generiert bzw. gefiltert werden.

  
Auch mit ROME lässt sich ein RSS-Feed mit nur sehr wenig Code anfordern und parsen.

### Wichtigste Anwendungsfälle

* Einlesen von RSS-Feeds: Der Server fordert regelmäßig den RSS-Feed eines Nachrichtenportals an, um diesen nach neuen Inhalten zu durchsuchen. Damit dieser in einer Programmlogik verarbeitet werden kann müssen die einzelnen Felder des Feeds eingelesen und auf eine vorher vereinbarte Art und Weise in Form von Objekten dargestellt werden. Dabei darf es keine Rolle mehr spielen, in welchem Format der Feed zuvor vorlag.
* Ausgabe von RSS-Feeds: Ein Nachrichtenportal (oder die im Rahmen dieses Projekts zu erstellende Webanwendung) stellt seinen Nutzern einen neuen Artikel zur Verfügung. Damit diese auch mithilfe automatisierter Tools über den Artikel informiert werden können, wird die URL zusammen mit einem Titel, einer Beschreibung und weiteren Informationen in einen RSS-Feed geschrieben. Dieser muss einem zuvor festgelegten Standard entsprechen, damit automatisierte Anwendungen diese Informationen korrekt auslesen können.

### UML und Beschreibung der inneren Logik



Für dieses Projekt relevantester UseCase:

Einlesen eines RSS-Feeds

1. Wie im obigen Codebeispiel gezeigt, wird durch den Aufruf von SyndFeedInput.build() das Einlesen eines Feeds angestoßen.

2. Der Aufruf wird an WireFeedInput delegiert

3. WireFeedInput wählt einen für das vorliegende RSS-Format geeigneten Parser aus

4. Dieser erzeugt ein Objekt, das den Feed mitsamt seinem Inhalt abbildet, allerdings ist dieses Objekt formatabhängig

5. SyndfeedInput erzeugt ein Objekt der Klasse SyndFeedImpl, die die vom RSS-Format unabhängige Schnittstelle SyndFeed implementiert.

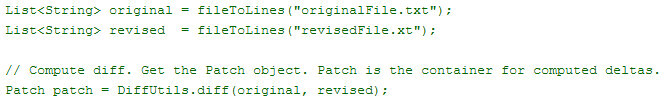
6. SyndFeedImpl ruft einen Converter auf, der die Daten des formatabhängigen WireFeed in den SyndFeed einträgt.

7. Der SyndFeed entält die Informationen des Feeds in einer formatunabhängigen Darstellung und wird zurückgegeben.

## Java-diff-utils

Dokumentation und Download: <https://code.google.com/archive/p/java-diff-utils/wikis/SampleUsage.wiki>

Java-diff-utils ist eine Java-Bibliothek zum zeilenweisen Vergleich von textbasierten Dateien. Damit lässt sich aus zwei Textdateien ein Patch-Objekt erzeugen, das Informationen über neu hinzugefügte und entfernte Textabschnitte(Deltas) enthält. Auf diese Weise lassen sich mit sehr geringem Aufwand Änderungen gegenüber der Vorgängerversion eines Feeds erkennen, sodass anschließend nur neu hinzugekommene Elemente verarbeitet werden müssen. Dies bewirkt eine deutliche Verringerung der zu verarbeitenden Datenmenge.

  
Auch die Generierung eines Patch-Objekts erfordert nur eine geringe Menge an Code.

## Feed Enlarger

URL: <http://feedenlarger.com/>

Feed Enlarger ist ein freier Service, mit dem sich Volltexte aus RSS-Feeds oder URLs generieren lassen. Durch die Nutzung dieses Service eröffnet sich die Möglichkeit, nicht nur den Titel oder die Beschreibung eines Feeds, sondern auch den Inhalt des damit verknüpften Artikels nach bestimmten Schlagwörtern zu durchsuchen. Bei Feed Enlarger handelt es sich um eine externe Webanwendung, die darauf ausgelegt ist, dass ein Nutzer eine URL einer Webseite oder eines Feeds in ein Formular einträgt. Der gleiche Effekt lässt sich allerdings erreichen, wenn man die URL, sowie einige zusätzliche Parameter mit der Methode GET übergibt. (Beispiel: <http://feedenlarger.com/makefulltextfeed.php?url=www.tagesschau.de%2Fwirtschaft%2Fglyphosat-189.html&max=5&links=remove&exc=1&submit=Create+full+text+feed>)

Da es sich um eine externe Webanwendung handelt, muss bei der Planung und Implementierung berücksichtigt werden, dass die Ergebnisse möglicherweise falsch oder unvollständig sind, und dass für die Erreichbarkeit nicht garantiert werden kann.

Eine eigene Implementierung dieses Verhaltens gestaltet sich schwierig, da der vollständige Artikel meist nur im HTML-Text der im RSS-Feed angegebenen URL zu finden ist. Allerdings gibt es keine verbindliche Norm, wie und wo der Inhalt innerhalb des Seitenquelltextes zu finden ist. Hilfreich wäre bei der Suche die Heuristik, dass es sich bei dem Artikel wohl um eine größere Zahl von langen und nah beieinanderstehenden Absätzen handelt.

## Lucene

Dokumentation und Download: <https://lucene.apache.org/core/>

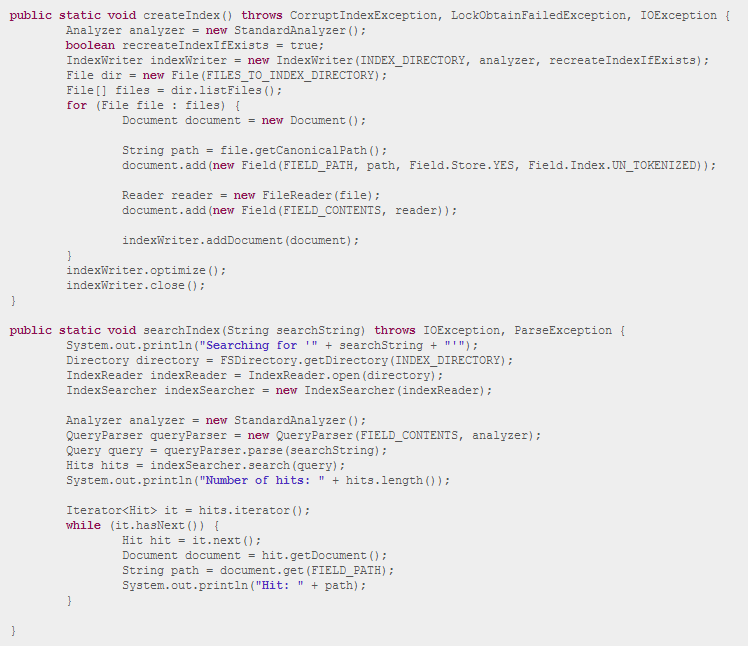
Lucene ist eine in Java-Bibliothek zur Volltextsuche. Mit dieser Bibliothek lassen sich längere Texte indizieren, die indizierten Texte lassen sich wiederrum relativ performant durchsuchen.

Trotzdem ist die Volltextsuche, also die Suche in einem Text nach bestimmten Schlagwörtern eine kritischste Aktion, die massiven Einfluss auf die Laufzeit der Anwendung hat. Dies gilt insbesondere dann, wenn viele Texte vorliegen, die nach vielen Suchbegriffen gefiltert werden müssen. Aus diesem Grund sollte bereits beim Entwurf der Anwendung darauf Wertgelegt werden, dass Indizierungen oder Volltextsuchen nicht unnötigerweise oder mehrmals ausgeführt werden.

Zu den für dieses Projekt wichtigsten Eigenschaften von Lucene gehören:

* Threadsicherheit/Parallelisierbarkeit
* Geringe RAM-Anforderungen (belegt ca. 1 MB auf dem Heap)
* Hoher Datendurchsatz (laut Angabe 150GB/h auf moderner Hardware)

Neben dem bloßen Durchsuchen von Texten bietet Lucene noch eine Reihe weiterer Funktionen (Möglichkeiten komplexer Abfragen, sortierung und Gruppierung der Ergebnisse, Hinzufügen zusätzlicher Tagfelder), die für das aktuelle Projekt allerdings nicht relevant zu sein scheinen.

  
Codebeispiel zur Verwendung von Lucene. Die gezeigten Funktionen dienen zur Generierung und dem Durchsuchen mehrerer Indexdateien.

Quellen:

* <https://de.wikipedia.org/wiki/Hypertext_Markup_Language>
* <https://de.wikipedia.org/wiki/RSS_(Web-Feed)>
* <https://de.wikipedia.org/wiki/Extensible_Markup_Language>
* <http://www.tagesschau.de/xml/rss2>
* <http://www.tagesschau.de/newsticker.rdf>
* <http://www.tagesschau.de/xml/atom/>
* <http://www.rss-verzeichnis.net>
* <http://www.freshnews.de/RSS-Einfuehrung-18061.html>
* <http://www.make-rss-feeds.com/rss-tags.htm>
* <https://validator.w3.org/feed/docs/rss2.html>
* <https://www.w3schools.com/xml/xml_rss.asp>
* <http://magpierss.sourceforge.net/>
* <https://rometools.github.io/rome/index.html>
* <https://code.google.com/archive/p/java-diff-utils/wikis/SampleUsage.wiki>
* <https://www.wprssaggregator.com/how-to-get-full-articles-from-rss-feeds/>
* <https://lucene.apache.org/core/>
* <http://www.avajava.com/tutorials/lessons/how-do-i-use-lucene-to-index-and-search-text-files.html>