Vergleich von Wrappersystemen

Hauptseminararbeit im Fach Informatik
WS 2001 / 2002

vorgelegt von

Markus Weißmann

geb. am 11.01.1979 in Erlangen

Angefertigt am 14.03.2002

Institut für Informatik
Lehrstuhl für Datenbanksysteme
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: ***Dipl.-Inf. Wolfgang Hümmer***

VERGLEICH VON WRAPPERSYSTEMEN

Markus Weißmann, Wolfgang Hümmer

Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstr. 3, 91058 Erlangen
EMail: markus.w.weissmann@informatik.stud.uni-erlangen.de

Abstrakt

Wrapper sind heute ein unabdingbares Mittel um Informationen aus Quellen zu beziehen, die eine inkompatible Schnittstelle benutzen oder schlicht ein Übermass an Inhalt bieten. Da die Bezugsquellen zahlreich sind und sich die Schnittstellen v. a. im www schnell ändern, ist es essentiell Wrapper zügig und in grosser Menge herstellen zu können. Diese Seminararbeit bietet einen Überblick über drei Wrapperframeworks zur Erzeugung von Wrappern und ihre dabei angewande Vorgehensweise.

Das Hauptaugenmerk bei der Auswahl der Frameworks lag zum Einen darin, dass ihre Vorgehensweisen stark differenziert sind und zum Anderen, dass die Schnittstelle der erzeugten Wrapper vornehmlich HTML, XML oder reinen Text als Quelle benutzt, womit die Nutzung von Inhalten aus dem Internet möglich ist.

Die untersuchten Wrappergeneratoren sind die "World Wide Wrapper Factory" (W4F), das "XWRAP Elite" System und "BuildHLRT", wobei von den beiden Erstgenannten Implementierungen zur Verfügung standen.

Inhaltsverzeichnis

1 Einleitung

Wrapper sind in der Zeit des Internet eine elegante Methode, um der Informationsflut, die das World Wide Web mit seinen mittlerweile mehreren Milliarden Seiten, deren Anzahl täglich ansteigt, mit sich bringt, Herr zu werden. Vor allem das Durchsuchen dynamischer Seiten, die sich oft ändern, wie z. B. Newsdienste, nach benötigten Informationen erfordert ohne den Einsatz von Wrappern viel Zeit. Da das Internet weiter wachsen wird, und der Anteil an für eine Firma, Institution oder Privatperson interessanten Informationen weiterhin so niedrig bleiben oder eher noch absinken wird, werden immer mehr Informationsfilter benötigt. Der Anteil an unnützen Daten hingegen nimmt bedingt u. a. durch aufwendigeres Layout und immer mehr Werbung im WWW stark zu. Um dieser steigenden Diskrepanz gerecht zu werden, ist es nötig, die Erzeugung von Wrappern zu automatisieren. Nur durch die aktive Unterstützung durch den Computer wird die Generierung von Filtern und Adaptoren schritthalten können, ohne den Zeit- und Kostenaufwand zu sprengen. Zwar ist das Haupteinsatzgebiet von Wrappern derzeit unbestritten das Internet, dennoch soll nicht unerwähnt bleiben, dass ein breites Spektrum an Informationslieferanten als Quellen in Betracht kommen. Das Ziel ist es, nicht nur unterschiedliche Angebote des WWW nutzbar zu machen, sondern generell jede Quelle, sei es Datenbank oder Textdokument, anzuzapfen und diese für die vorhandene Infrastruktur zur Verfügung zu stellen.

Ziel ist es, Vorgehensweisen zur Erzeugung und Pflege von Wrappern aufzuzeigen, welche versprechen effizienter zu sein, als manuelle Vorgehensweisen.

2 Wrapper-Framework

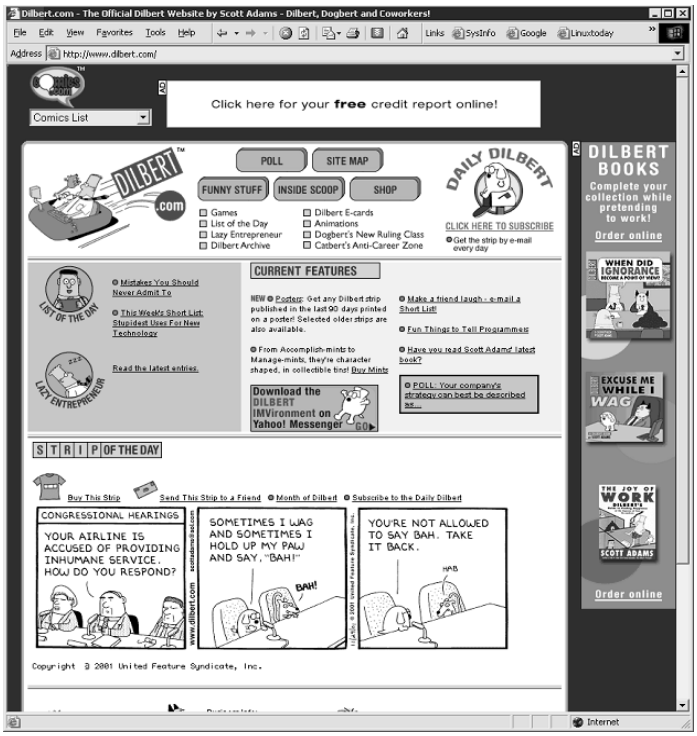
Um den Begriff des Wrapper-Frameworks zu klären, bedarf es zuerst einmal der Klärung der beiden Begriffe Wrapper und Framework. Ersterer ist im wesentlichen ein Adapter, welcher es erlaubt, im Bezug auf die Schnittstelle inkompatible Systeme zusammen zu bringen. Er greift zunächst auf die Daten aus der im angebotenen Schnittstelle zu. Der zweite Schritt besteht darin, aus den gelieferten Daten die gewünschten zu extrahieren. Abschließend werden die Informationen in das angeforderte Protokoll gepackt und ausgegeben. Eine tiefergehende Ausführung dessen ist hier [Berg02] zu finden. Für den zweiten Begriff, Framework bietet [Comp99] eine ausreichende Erklärung: "Ein Begriff aus der objektorientierten Programmierung. Eine erneut einsetzbare Design-Grundstruktur, die aus abstrakten und konkreten Klassen besteht und das Erstellen von Anwendungen unterstützt." Da die Objektorientierung schon seit längerem in der Praxis angewand wird, bedarf der erste Satz keiner genaueren Erläuterung. Die für ein Framework wirklich interessanten Eigenschaften finden sich im zweiten Satz. Die Anforderungen bestehen in einer vorhandenen Grundstruktur, realisiert z. B. durch Bibliotheken. Diese sollen den Entwickler bei seiner Arbeit unterstützen und ihm wiederkehrende Aufgaben abnehmen. Da eine einzelne Bibliothek z. B. für HTTP-Zugriffe diesen Anforderungen durchaus schon genügen kann, wird im Folgenden die Bedingung für ein Wrapper-Framework höher angesetzt: Ein Wrapper-Framework muss explizit das

Erstellen von Wrappern fördern und zu diesem Zweck das Durchsuchen von (HTML-)Dokumenten vereinfachen. Die Erstellung eines Wrappers bedarf nicht zwingend eines Frameworks, jedoch ist eine erhöhte Produktivität durch ein solches zu erwarten, da viele Module immer wieder benötigt werden und Dank des Frameworks nicht neu erzeugt werden müssen. Ein gutes Wrapper-Framework spart Arbeitszeit durch die Möglichkeit schneller neue Quellen zu erschließen, sich in kürzerer Zeit an veränderte Quellen anzupassen und, durch die Modularität, weniger Fehler machen zu können.

Jedes Framework wird auf eine Reihe wichtiger Kriterien hin untersucht. Wie funktioniert die Extraktion der Daten aus dem Dokument? Wie werden die Extraktionsregeln erstellt und evtl. in welcher Sprache? Wie kann auf die Dokumente zugegriffen werden? Auf welche Art ist der fertige Wrapper in ein größeres System einzubinden? Was sind die Voraussetzungen an die Laufzeitumgebung des Wrappers? Zu diesen Fragen gesellt sich zudem eine natürlich subjektive Bewertung die unter anderem die Benutzbarkeit und die auftretenden Kosten des Frameworks beleuchtet. Falls Unzulänglichkeiten aufgetreten sind werden diese an dieser Stelle ebenfalls erörtert.

2.1 Beispiel eines Wrappers: "gci"

Bevor die Bewertung von Frameworks möglich ist, ist es nötig sich mit dem Endprodukt des Generators zu beschäftigen. Dazu wird ein Beispiel-Wrapper bewertet. gci ist ein von Hand erstellter Wrapper, dessen Designziel lediglich darin besteht zu funktionieren. Zur Motivation wird der Wrapper eine Internetseite filtern, die eher der Unterhaltung dient: www.dilbert.com (Abbildung 2.1).




```
<html>
...
<td bgcolor="#ffffff">
<a href="/comics/dilbert/archive/
        images/dilbert2001121015501.gif">
<img src="/comics/dilbert/archive/
        images/dilbert2001121015501.gif“
        border=0 alt="Today's Dilbert
        Comic">
</a><br><br>
</td>
...
</html>
```

a.) Webseite in Browseransicht b.) Der HTML-Code

Abb. 2.1: Dilbert Webseite (Browser, HTML-Code)

Auf dieser Seite befindet sich ein täglich wechselnder kurzer Comic, der "daily dilbert". Ziel des Filters ist es, diesen Comic, der als Bild in die Seite eingebettet ist, zu extrahieren und diesen lokal auf dem Massenspeicher des Computers zu speichern und ihn an ausgewählte Adressaten per E-Mail zu verschicken. Um den Code kurz und verständlich zu halten, ist der Wrapper als Unix Shell-Skript implementiert, was zwar nicht besonders ressourcenschonend ist und deshalb für den Masseneinsatz disqualifiziert, aber die Übersichtlichkeit erhöht und z. B. keine tieferen TCP/IP-, HTTP- oder SMTP-Kenntnisse erfordert.

```
00  #!/bin/bash
01  # gci (c)Markus Weissmann, 2002
02  mailto="markus.w.weissmann@informatik.stud.uni-erlangen.de"
03  mailcc="wolfgang.huemmer@informatik.uni-erlangen.de"
04  mailfrom="dailydilbert@faui61.informatik.uni-erlangen.de"
05  date=`date +%B" "%d" "%Y`
06  subject="Daily Dilbert from ${date}"
07  picpath=/mnt/pictures/dilbert
08
09  path=`curl -s "http://www.dilbert.com/"
10          | tr [:blank:],'<','=','"' '\012'
11          | grep "/comics/dilbert/archive/images/dilbert"
12          | tail -n1`
13
14  file=` echo $path | tr '/' '\012' | grep ".gif" `
15  url=http://www.dilbert.com${path}
16
17  curl -s ${url} -o ${picpath}/${file}
18  echo "" | sendemail -q -a ${picpath}/${file} -u ${subject} -f ${mailfrom} -t ${mailto} -cc ${mailcc}
```

Zur Erläuterung des Quellcodes:

- "mailto" (02) ist die Variable, welche den unmittelbaren Adressaten enthält.
- "mailcc" (03) besteht aus allen Mailadressen, an die die E-Mail außerdem gehen soll.
- "mailfrom" (04) ist die Adresse des Absenders.
- "date" (05) bekommt die Ausgabe des "date"-Befehls, welche in der Form "Monatsname-Tag des Monats-Jahr" ausgegeben wird.
- "subject" (06) entspricht der Betreff-Zeile der E-Mail, die sich aus "Daily Dilbert from" und dem Datum zusammensetzt.
- "picpath" (07) ist der (absolute) Pfad zu dem Depot der Dilbert Bildersammlung.
- "path" (09) wird die Ausgabe des Programms "curl" zugewiesen, nachdem diese noch drei Filter durchlaufen hat und wird dann den relativen Pfad zu dem Bild enthalten.
- "file" (14) erhält den aus "url" entnommenen Dateinamen des Bildes (ohne Pfad).
- "url" (15) besteht aus dem Protokoll, dem Servernamen und dem Pfad des Bildes auf dem Server, also den kompletten "Uniform Resource Locator" des Bildes.

Der Aufruf in den Zeilen 09 bis 12 ist das Herz des Wrappers: Der Befehl "curl" [Curl01] kopiert die Datei der angegebenen URL http://www.dilbert.com/ und gibt diese auf der Standardausgabe aus. Die Standardausgabe wird dann an den "tr"-Befehl per Unix-Pipe übergeben. Dieser ersetzt die Zeichen <, =, " durch einen Zeilenumbruch ("/012"). Da so sicher gestellt werden kann, dass alle HTML-Tags, welche alle mit einem "<" beginnen, in einer neuen Zeile stehen, kann nun mit dem Programm "grep" in ihnen gesucht werden. Der HTML-Tag "img" enthält das Attribut "src=", nach welchem in Anführungszeichen eingebettet die URL des Bildes folgt. Die Anführungszeichen sind ebenfalls durch einen Zeilenumbruch ersetzt worden, so dass die URL nun separiert in einer eigenen Zeile steht. "grep" überprüft nun alle Zeilen auf das Vorkommen der Zeichenkette "/comics/dilbert/archive/images/dilbert", welche nach einer manuellen Sichtung der HTML-Datei im Pfad des gesuchten Bildes vorkommt. Alle übrigen Zeilen enthalten den Bildpfad und es wird willkürlich mit "tail" die letzte Zeile ausgewählt.

In Zeile 14 wird der gefundene Pfad mittels des "echo"-Kommandos in das "tr"-Programm per Unix-Pipe, welche lediglich die Standardausgabe des "echo" auf die Standardeingabe von "tr" umleitet, übernommen und dort an den Verzeichnisbegrenzern ("/") in Zeilen aufgeteilt. In diesen kann nun analog nach einem String ".gif", der erwarteten Dateiendung des Bildes gesucht werden, was den Dateinamen des Ziels liefert.

Der befehl "curl" in Zeile 17 kopiert nun lediglich die Datei von der gefundenen URL auf den lokalen Massenspeicher an die Stelle, welche in "picpath" angegeben wurde.

Nun bleibt nur der Aufruf des Befehls "sendemail" (18), der das Bild Abbildung 2.2 mit den anfangs gesetzten Parametern als E-Mail verschickt.

Der Wrapper gci ("get changing image") ist voll funktionsfähig und befindet sich auch im Einsatz.l

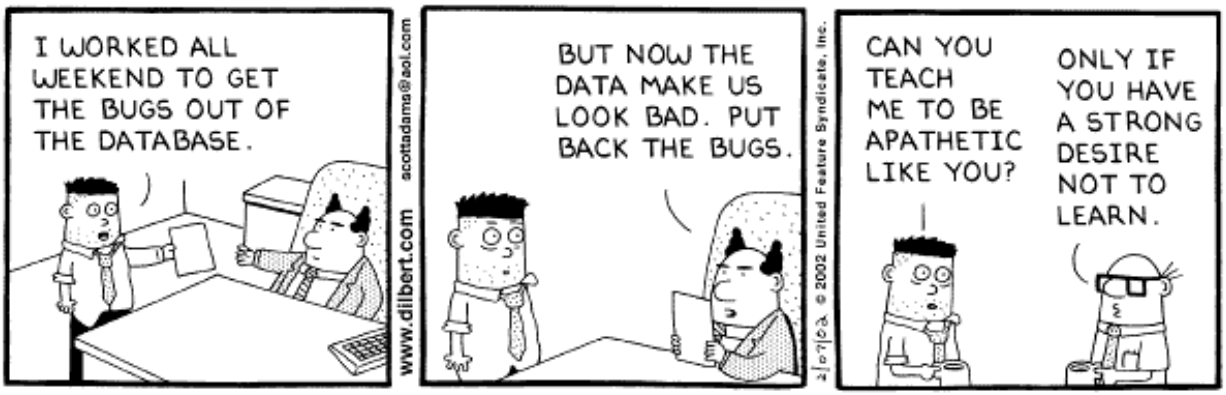


Abb. 2.2: Beispiel des daily dilbert

2.2 Klassifikation und Bewertung

Die Beurteilung von gci fällt sehr einseitig aus: Auf der einen Seite ist er mit wenigen Unix-Kenntnissen schnell zu erstellen. Auch ist er sehr stabil gegen viele Änderungen der gewrappten Seite, solange das Bild an dem ausgemachten Pfad auf dem Server liegt und sich das Dateiformat, genauer die Dateiendung, nicht ändert. Er lässt sich einfach auf andere Seiten, die wechselnde Bilder bieten, umstellen. Auf der anderen Seite überwiegen jedoch die Nachteile, welche größtenteils mit diesem Ansatz auch kaum lösbar sind: Die Anforderungen, denen gci genügt, sind nur sehr einfacher Art. Der Wrapper kann nur eine URL aus dem Dokument filtern, da die Seite bei der Bearbeitung zerstört wird. Selbst bei einer Zwischenspeicherung des Ursprungsobjekts muss der Parser für jede Informationseinheit das gesamte Dokument erneut durchlaufen, was einen noch höheren Ressourcenaufwand bedeuten würde, als gci schon aufweist: Für einen Durchlauf müssen weitere 11 Prozesse erzeugt werden, was diese Vorgehensweise für einen Masseneinsatz disqualifiziert.

Ein weiterer Nachteil dieses monolithischen Ansatzes ist der, dass der Wrapper sich nur sehr schlecht erweitern lässt: Es kann nur über einen weiteren Prozess auf die Funktionalität einer mächtigen Programmiersprache wie C, Perl oder Java zugegriffen werden. Die Möglichkeiten der Shell sind im Vergleich hierzu doch sehr bescheiden, was sich auch bei der Kommunikation mit der üblichen Umgebung eines Wrappers zeigt: Die Schnittstellen, die der Wrapper bereitstellt umfassen praktisch nur die Standardein- und Ausgabe und das Dateisystem, aber keinen Datenbank- oder umfassenden Netzwerkzugriff. Des weiteren funktioniert der Wrapper selbstverständlich nur in einer Unix-Umgebung.

Um all diese Unzulänglichkeiten des Beispielprogrammes zu beseitigen, werden nun verschiedene Wrapper-Frameworks betrachtet, welche sich der Lösung der verschiedenen Probleme annehmen.

3 Vergleich von Wrapper-Frameworks

Wrapper-Frameworks sollen die Erzeugung aber vor allem auch die Wartung von Wrappern vereinfachen und automatisieren. Durch diese Automatisierung können Zeit und Kosten gespart, Fehler vermieden und u. U. auch die manuelle Suche in den zu durchsuchenden Dateien begrenzt werden. Auch wird der Wiederverwendungswert erhöht und der Gesamtprozess stark vereinfacht, was es dem Wrapper-Bauer ermöglicht, sich auf wichtigere Aspekte zu konzentrieren, als z. B. welcher HTML-Tag die gewünschte Information umfasst.

Es werden drei konkrete Wrapper-Frameworks betrachtet, von denen zwei als Implementierung für Tests zur Verfügung standen.

3.1 W4F – WWW Wrapper Factory

Die World Wide Web Wrapper Factory (W4F) ist ein Produkt der Firma Tropea Incorporated. Ursprünglich als Forschungsobjekt begonnen, wurde es später als Shareware vertrieben. Eine auf 30 Tage Laufzeit begrenzte Version wurde freundlicherweise von Arnaud Sahuguet <sahuguet@gradient.cis.upenn.edu> auf Anfrage zur Verfügung gestellt. Der derzeitige Status der Software ist nicht bekannt. Die Homepage von Tropea-Inc. ist derzeit nicht unter der im Handbuch angegebenen URL [Trop01] erreichbar.

Die beiden Autoren, Fabien Azavant und Arnaud Sahuguet haben mit W4F ein Wrapper-Framework geschaffen, welches nicht nur Wrapper in Java erzeugt, sondern auch selbst in der Multiplattformsprache gehalten ist. W4F erwähnt in seinem Handbuch sowohl die Win32-Laufzeitumgebung als auch eine Unix-Umgebung, in der es benutzt werden kann (eine JVM vorausgesetzt). Ebenfalls unterstützt wird die Plattformunabhängigkeit durch die XML-formatierte Ausgabe der erzeugten Wrapper.

Ein einleitender Satz aus dem [AzSa00] erläutert die Absichten und Ziele von W4F: "W4F is a toolkit for building software components that integrate Web-published data into applications." Hiermit zielt W4F auf die in [AzSa00] erwähnten "weblications", Anwendungen die auf Daten, die aus dem WWW bezogen wurden, arbeiten.

3.1.1 Zugriff

Die von W4F generierten Wrapper unterstützen als Zugriffsoperationen neben dem direkten Dateizugriff drei http Methoden:

- HTTP HEAD, um die HTTP-Informationen über ein Dokument, welche z. B. die Dateilänge, den Servertyp oder die lokale Zeit des Server beinhalten können, von einem Server zu beziehen (Der HTTP-Header ist nicht mit dem HTML-Tag "head" zu verwechseln!),
- HTTP GET, um die Datei selbst zu erhalten und
- HTTP PUT, um selbst Daten an einen Webserver übermitteln zu können, beispielsweise um Formulare auf einer Internetseite automatisch ausfüllen zu können, deren Ergebnis danach mit HTTP GET geholt werden kann

Nicht zur Verfügung stehen SSL-verschlüsselter Zugriff per HTTPS.

3.1.2 Extraktion

Als Extraktionsoperatoren verwendet W4F die HTML Extraction Language (HEL). Sie ist eine an XSL erinnernde Extraktionssprache, mit welcher bestimmte Tags in HTML/XML Dokumenten adressiert werden.

```
html.body.table[0].tr[0]->td.getAttr(bgcolor)
```

Dieses Beispiel wandert in einem HTML-formatierten Text den SGML-Baum hinab. Zuerst nimmt es den Wurzelknoten <html>, dann verzweigt es zu <body>. Danach wandert es in den 0. (1.) <table>-Ast, dann in den 0. (1.) <tr>-Ast und liefert von dieser Stelle, aus allen nachfolgenden <td>-Tags, den Wert des Attributs "bgcolor" als Ergebnis. Weitere zulässige Operatoren sind:

Operator	Bedeutung
.	direkter Nachfolger im Baum
->	alle Subknoten im Baum
-/>	alle folgenden Knoten im Dokument
.numberOf(Tag)	Anzahl der Tags im Subknoten
.getAttr(attribute)	Attributwert "attribute" des Knoten/Blattes
WHERE bool AND bool	logische Verknüpfung

Tab. 3.1: HEL-Operatoren

HEL besitzt nicht die Mächtigkeit von XSL und ist zudem proprietär, was jedoch nachzusehen ist, da HEL älter ist, als die Standardisierung von XSL.

Zur Überprüfung der HEL besitzt W4F eine grafische Benutzerschnittstelle, den Wizard (Abbildung 3.1) – ebenfalls in Java gehalten – mit welchem es möglich ist, die in HEL erstellten Regeln unmittelbar zu überprüfen.

3.1.3 Interne Darstellung

Wrapper die dem W4F Framework entsprungen sind, verwenden intern eine Datendarstellung, welche die Autoren als Nested String List (NSL) bezeichnen. Sie nimmt sich aus wie die in Scheme und anderen Lisp-Dialekten verwendeten Listen, die dort als universelle Datenstruktur benutzt werden. In der NSL gibt es ebenfalls Elemente, welche jeweils bis zu einen linken und einen rechten Nachfolger besitzen können.

Der von W4F erzeugte Java-Code sieht beispielsweise folgendermaßen aus:

```
/* 7 W4F & regexp-Bibliotheken importieren */
...
RetrievalAgent          ra = new RetrievalAgent("http://db.cis.upenn.edu/W4F");
HtmlDocumentParser      hdp = new HtmlDocumentParser(org.openxml.dom.DocumentImpl.class) ;
org.w3c.dom.Node        root = hdp.parse(ra) ;
ExtractionPathParser    epp = new ExtractionPathParser() ;
ExtractionPath          ep = epp.parseExtractionPath("html->a[i:*] ( .txt # @+href ) where
                             a[i]@href != null") ;
NestedStringList        result = ep.getNSLFrom(root) ;
...
```

Zuerst werden die W4F und die Bibliothek [Regx01] eingebunden, ohne welche der Wrapper nicht auskommt. Es werden eine URL und ein Extraktionspfad in HEL übergeben. Die Rückgabe des Parsers ist eine NSL, welche nun von dem Wrapper - analog wie in Lisp - durchlaufen werden will. Diesen Part muss nun wieder der menschliche Wrapper-Generator übernehmen.

Abb. 3.1: W4F Wizard

Durch die Übergabe der aus der Quelle extrahierten Daten in einem Java-Programm eröffnet den Wrappern natürlich alle externen Anbindungen von Java, allen voran den Datenbankzugriff per JDBC [Jdbc02] und die XML-Ausgabe [Jxml02].

3.1.4 Klassifikation und Bewertung

Es ist unübersehbar, dass W4F ein manueller Wrapper-Generator ist: Die Regeln zur Extraktion müssen manuell erstellt werden, auch wenn dazu grafische Unterstützung in Form des Wizard gegeben ist. Die Wrapper beherrschen Datei- und umfangreiche http-Zugriffe. Die Extraktion erfolgt pfadbasiert, die Ausgabe direkt in einem zu schreibenden Javaprogramm.

Der Wizard von W4F ist eine große Hilfe, bei der Verifizierung der Extraktionsregeln. Da alle Komponenten in Java gehalten sind, ist W4F extrem plattformunabhängig und besitzt eine grandiose Erweiterung allein dank der mittlerweile über 100 offiziellen Java Bibliotheken. Leider ist der Wizard relativ fehlerbehaftet und es sind ein paar proprietäre Nachteile hinzunehmen: Die HEL ist eine weitere pfadbasierte Extraktionssprache, deren es seit der Einführung von XSL eigentlich nicht mehr bedarf, und die [Regx01], die zusätzlich erworben werden muss, ist seit der Aufnahme der regexp-Bibliothek des Apache Projektes [Jreg02] in die von Sun offiziell unterstütze Bibliothekensammlung ebenfalls überflüssig. Ein weiteres Problem ist der Status des W4F Projektes, der

den Anschein erweckt, dass an der Software zumindest derzeit nicht weiterentwickelt wird. Informationen zu W4F existieren in Form einiger Veröffentlichungen der Autoren, u. a. [AzSa99a], [AzSa99b], [AzSa98].

3.2 XWrap Elite

Vom Georgia Institute of Technology kommt XWrap Elite, ein "eXtensible Wrapper Generation System", wie die Entwickler es selbst nennen. Auch auf eine Implementierung dieses Wrapper-Frameworks bestand direkter Zugriff. Auf [W4fh01] steht XWrap jedem, der sich dort registriert eine kostenfreie Evaluationsversion in Form eines Webfrontends zur Verfügung. Ein Zitat von der Homepage [Xweh01] des Projekts zeigt seine Ziele auf: "XWrap Elite is an XML-enabling software tool, that can automatically generate wrapper programs for Web information sources." Der Anspruch, automatisch Wrapper generieren zu können wird im Folgenden näher beleuchtet.

3.2.1 Zugriff

Der Zugriff auf die XWrap-Implementierung erfolgt derzeit ausschließlich über das eingangs erwähnte Webfrontend. Nach der Registrierung kann man dort über einen Proxy die Quellseite, die gewrapped werden soll ansteuern. Dabei stehen sowohl HTTP- als auch HTTPS-Zugriffe bereit. Durch dieses Verfahren ist es möglich an (fast) jede Seite im Internet, an die auch von dem eigenen Internet-Zugang Zugriff besteht, zu gelangen. Eine Anleitung hierzu ist unter [Xwwa01] zu finden. Was sich als problematisch erweisen könnte, was aber aus der XWrap Dokumentation nicht zweifelsfrei hervorgeht, ist, dass Browser im Stande sind Dinge zu tun, die nicht trivial nachvollziehbar sind: Beim Ansteuern der Zielseite kann der Benutzer sich authentifizieren, diverse Cookies speichern oder, was in vielen Webshops Gang und Gäbe ist, Seiten ansteuern, die exklusiv und temporär für ihn angelegt wurden. Der Proxy muss nun, um all das nachvollziehen zu können, nicht nur die finale URL speichern, sondern den kompletten Ansteuerungsvorgang aus Benutzersicht, welcher aus den HTTP-Anfragen des Browsers nicht unbedingt eindeutig zu schließen ist.

3.2.2 Extraktion

Die Extraktion erfolgt bei XWrap Elite halbautomatisch. Der Benutzer wählt eine Extraktionsart aus der angebotenen Liste aus und bekommt daraufhin ein vorläufiges Ergebnis, welches er entweder zu Gunsten einer anderen Extraktionsart verwirft, noch von Hand nachbearbeitet oder als finale Filterregel akzeptiert. Ein Beispiel einer guten Extraktion der Seite [LiTo01] ist in Abbildung 3.2 zu sehen.

Als Operationen stehen bei XWrap einige komplexe Untersuchungen bereit: Zuerst erfolgt die Teilbaumuntersuchung die für jeden Tag des HTML-Baumes eine Gewichtung erzeugt, auf welcher fußend die Priorität bei der Filterung ermittelt wird. Es existieren drei Teilbaumevaluationsoperationen:

- largest tag count

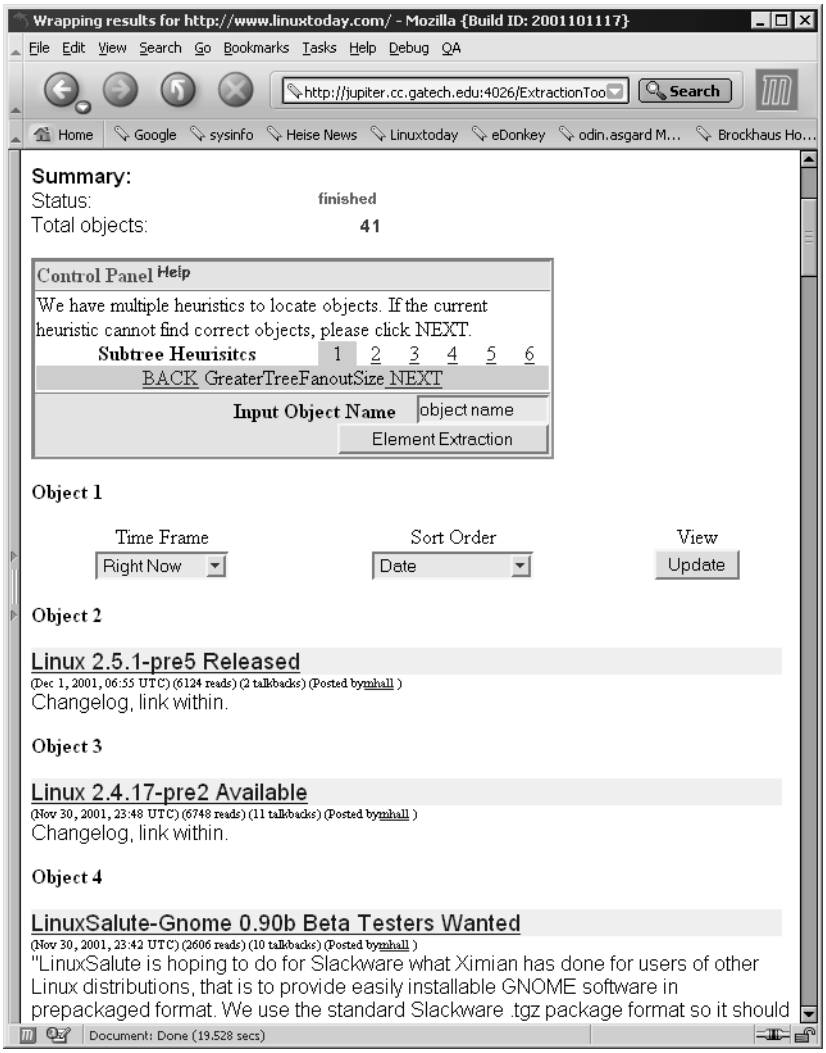


Abb. 3.2: XWrap Elite Extraktionssteuerung

- highest fanout
- largest size increase

Der "largest tag count" ermittelt eine Wertung anhand der Anzahl aller Subknoten des jeweiligen Tags. In einem HTML-Dokument bekommt der Tag <html> somit immer die höchste Wertung bei dieser Bewertung. Nur die Anzahl der direkten Nachfolger, also der unmittelbaren Kindknoten evaluiert der "highest fanout" - beispielsweise ein Tag <table>, das eine große Tabelle einleitet, wird bei diesem Verfahren eine starke Gewichtung erfahren. Als dritte Methode untersucht "largest size increase" das Verhältnis der unmittelbaren Kindknoten relativ zur Anzahl der Geschwisterknoten.

Als zweiter Schritt folgt bei XWrap die Elementextraktion. Hier wird lediglich in einer flachen Struktur, durch zwei differenzierte Verfahren, das Vorkommen der Tags bewertet:

- highest tag count
- repeating pattern heuristic

Das Erstgenannte bewertet die reine Anzahl der in dem Dokument erscheinenden Tags. Eine hohe Wertung bei diesem Bewertungsschema erfahren die oft verwendeten Tags. Bei "repeating pattern heuristic" wird die Position des Tags im Dokument berücksichtigt. Hier wird das Auftreten der Elemente in Reihe beachtet.

Der dritte Vorgang, den das Quellobjekt durchläuft, ist die Ausgabeidentifikation. Diese versucht die Ausgabe zu korrigieren, indem sie erstellte Objekte angleicht. Dazu wird zwischen stark- und schwachbindenden Tags unterschieden. Links und Text mit ¥ oder $ Zeichen sind stark bindend während z. B. ein Zeilenumbruch
 lediglich schwach bindend ist. Wie in Tabelle 3.2 zu sehen ist, wird in Objekt2 das <img> Tag, da es stärker bindet als der einfach Text in Zeile 4 und 5 von Objekt1 an das gleiche Tag in Objekt1 angeglichen und in Zeile 6 von Objekt2 plaziert, statt in Zeile 4, welche als nächste frei ist.

Nr.	Objekt 1	Objekt 2
1	Prentice Hall	Prentice Hall
2	Our Price:	Our Price:
3	$34.40	$64.75
4	You save	
5	20%	
6	<img src="...">	<img src="...">

Tab. 3.2: Ausgabeidentifikation

3.2.3 Ausgabe

XWrap Elite erzeugt wie W4F seine Wrapper als Java-Quellcode. Die Wrapper erzeugen von sich aus ein XML-Objekt, welches der Wrapper-Autor manuell weiterbearbeiten muss, um es beispielsweise per JDBC in eine Datenbank zu transferieren oder kann es mit wenigen Zeilen Code als XML-Dokument ausgeben. Der Quellcode eines von XWrap Elite erzeugten Wrappers sieht z. B. so aus:

```
...
setSubtreeHeuristic(0);
setSubtreePath(".body[1].table[0]");
setFollowNextLink(false);
ontology.addElement("price");
setObjectSeparator("tr");
xmlOutputDescription.setObjectName("news");
xmlOutputDescription.setAttributeName(3, "Titel");
xmlOutputDescription.setAttributeType(3, 1);
xmlOutputDescription.setAttributeName(2, "Link2");
xmlOutputDescription.setAttributeType(2, 3);
xmlOutputDescription.setAttributeName(1, "Time");
xmlOutputDescription.setAttributeType(1, 1);
XMLEliteExtractione26b63e5b7 eXMLOutputDescription = new
        XMLEliteExtractione26b63e5b7();
```

```
eXMLOutputDescription.run();
...
```

Anfangs werden die zu verwendenden Methoden festgelegt, danach erfolgt die Festsetzung der Ausgabeumsetzung in das XML-Objekts. Schlussendlich wird der eigentlich Wrapper gestartet, der das fertige XML-Dokument generiert. Natürlich benötigen die Wrapper die Java-Bibliotheken des XWrap-Frameworks.

3.2.4 Klassifikation und Bewertung

XWrap ist ein halbautomatisches Wrapper-Framework, je nach nötigen Eingriffen kann die Erzeugung des Wrappers sogar vollautomatisch erfolgen. Lediglich die Umsetzung des XML-Dokumentes in das Zielformat, sofern das gelieferte XML nicht das erwünschte Endprodukt ist, wird nicht unterstützt. Die Extraktion basiert auf komplexen Heuristiken und v. a. bei der manuellen Nachbearbeitung, pfadbasierten Methoden. Die Wrapper werden als Quellcode generiert, sind in Java und benötigen wie bei W4F die Framework-Bibliotheken. Die Integrierbarkeit ist dank Java sehr hoch.

Das Web-Interface von XWrap lässt sich durchaus als gelungen bezeichnen. Es ist sehr benutzerfreundlich und intuitiv. Lediglich die Problematik der sitzungsbezogenen Daten ist unklar. Wünschenswert ist hier eine Warnung des Benutzers, dass der Wrapper nicht alles nachvollziehen kann und das Surfverhalten daran angepasst wird. Das System macht insgesamt einen sehr ausgereiften Eindruck. Sowohl die generierten Wrapper, als auch das Framework selbst sind sehr betriebsystemunabhängig, die Wrapper, da sie nur auf eine Java Virtual Machine angewiesen sind, das Framework, weil nur ein Browser benötigt wird. Nachteilig ist hier, genau wie bei W4F, dass die Wrapper zur Funktion die Bibliotheken von XWrap-Elite benötigen. Was der professionellen Benutzung von XWrap entgegensteht ist, dass es keine Möglichkeit gibt, es käuflich zu erwerben. Daher muss man sich auf das Funktionieren des Frameworks auf dem Rechner des Georgia Institute of Technology verlassen und bekommt auch keinen professionellen Support. Ein weiteres zu nennendes Manko ist das automatische Wrapping. Beim Test von XWrap gab es meist nur zwei mögliche Ergebnisse: Entweder die Extraktion verlief extrem gut und nur minimale Nachbearbeitung war nötig, oder das Ergebnis war total unbrauchbar und der Aufwand der Nachbesserung kam einer manuellen Implementierung des Wrappers gleich. Weitere Informationen zu XWrap sind in [LPHB99a] und in [LPHB99b] zu finden.

3.3 BuildHLRT

Das dritte betrachtete Framework ist eine Entwicklung des Department of Computer Science & Engineering der University of Washington und der Firma NETbot Inc. aus Seattle. Die drei Autoren sind Nicholas Kushmerick, Daniel S. Weld und Robert Doorenbos. Das Ziel dieses Projektes ist es, Wrapper durch Induktion zu konstruieren. Die Erzeugung soll vollautomatisch erfolgen und nur eine genügend große Menge an Beispieldokumenten benötigen. BuildHLRT stand leider nicht als Implementierung zur Verfügung.

3.3.1 HLRT als Extraktionskonzept

Das Konzept, auf dem die Wrapper von BuildHLRT fußen, ist das schon im Namen steckende HLRT, Head-Left-Right-Tail. Diese Methode basiert auf dem einfacheren LR-Algorithmus. Dieser durchläuft die Quelle auf der Suche nach dem Left, dem L-Argument. Sobald er es gefunden hat, sucht er weiter, diesmal nach dem Right oder R-Argument. R- und L-Argument werden nun als Objekt zusammengefasst und abgelegt. Dann fährt die Suche fort wieder nach einem L zu fahnden. Ist das Objekt komplett bearbeitet, hat LR n Objekte gefunden, die weiterverwendet werden können. HLRT fügt diesem einfachen Konzept noch einen Head, einen Kopfteil und einen Tail, einen Fußteil hinzu. Diese werden initial am Anfang bzw. am Ende des Objektes entfernt, da sie unwichtig sind oder sogar den LR-Algorithmus behindern. In Tabelle 3.3 findet sich ein einfaches Beispiel eines konkreten HLRT-Ansatzes an HTML. Am Rande sei noch der BELR erwähnt, welcher wie LR funktioniert, jedoch noch ein Beginn und Ende als Argument erwartet, mit denen er die n Objekte des LR-Verfahrens unterteilen kann. HLRT und BELR lassen sich darüber hinaus noch zum HBELRT kombinieren, der die Vorteile beider Algorithmen vereint.

```
<body>
   <p>
                <b>Spanien</b>     <i>42</i>
                <b>Irland</b>      <i>23</i>
                <b>Portugal</b>    <i>8</i>
                <b>Belgien</b>     <i>6</i>
   </p>
</body>
```

Abb. 3.3: HLRT (<body> , {,},{ ,}, </body>)

Ein Beispiel einer HLRT-Analyse findet sich in Abbildung 3.3: Die HTML-Tags "<body> " bilden den Head (H), " </body>" den Tail (T). Die linke Seite (L) bestimmt die Startpunkte "" und "", die rechte Seite (R) legt die Endpunkte auf "" bzw. "" fest. Mit diesem Head wird nach "<body> " mit der Extraktion begonnen. Folgend werden die Daten zwischen "" und "", respektive zwischen "" und "" entnommen. Sobald der Tail " </body>" erreicht ist, wird der Vorgang beendet. Durch diese Vorgaben werden aus dem HTML-Code die Tupel (Spanien, 42), (Irland, 23), (Portugal, 8) und (Belgien, 6) extrahiert.

3.3.2 PAC Analyse, Orakel, Induktion

Zur Wrapper-Erzeugung benötigt BuildHLRT lediglich eine ausreichend große Menge an Quelldokumenten. Initial muss dem Framework noch die gewünschte Genauigkeit des zu erstellenden Wrappers genannt werden, was vor Allem auch starken Einfluss auf die Anzahl der nötigen Dokumente hat.

Um diese Anzahl der benötigten Quelldokumente zu bestimmen, wird die PAC-Analyse herangezogen. PAC, oder Propably Approximately Correct errechnet anhand der gewünschten Genauigkeit und der durchschnittlichen Dokumentgröße die Anzahl der notwendigen Beispiele. Während die

Genauigkeit nur proportional in die Berechnung eingeht, wächst das Ergebnis exponentiell zur Durchschnittsgröße. Damit disqualifiziert sich dieses Verfahren für Webseiten selbst, da diese oft sehr groß sind und sich v. a. nur einige Male pro Tag ändern. Daher ist es ein Ding der Unmöglichkeit von solch einer Seite Abertausende von Beispielen zu bekommen, wenn kein jahrelang geführtes Archiv dieser Seite vorhanden, oder keine Zeit, um ein solches anzulegen ist.

BuildHLRT ist auf Orakel angewiesen, Wissensdatenbanken, die Werte erkennen und zuordnen können. Ein Orakel sollte Beispielsweise auf eine Anfrage mit "Deutschland", "Russland", "Finnland" als Antwort "Ländernamen" liefern. Solche Orakel sind für eine Vielzahl von Daten wie Postleitzahlen, Mädchennamen oder Schuhgrößen vorstellbar. Ohne geeignetes Orakel für die in den Quellen enthaltenen Werte, ist keine Erzeugung von Wrappern möglich.

Das Wrapper-Framework, ausgestattet mit Orakeln, einer gewünschten Genauigkeit und darauf basierend einer ausreichenden Anzahl Dokumente arbeitet nun die Beispiele ab und erstellt mit dem Induktion genannten Verfahren den Wrapper her. Auf den aus dem Orakel bezogenen Daten werden HLRT-Regeln erstellt und an den weiteren Dokumenten verifiziert und verfeinert. Weiterführende Informationen zu BuildHLRT und Induktion finden sich in [Kush00a], [Kush99], [Kush00b].

3.3.3 Klassifikation und Bewertung

Grundsätzlich kann BuildHLRT auf allen Arten von Text arbeiten, sofern passende Orakel vorhanden sind. Dieses Wrapper-Framework arbeitet vollautomatisch ohne jeden Eingriff des Benutzers. Die Extraktion erfolgt mittels des HLRT-Verfahrens, also mittels der Erkennung von Text durch reguläre Ausdrücke. Zum Ausgabeformat und ebenso zur Integrierbarkeit ist mangels einer vorliegenden Implementierung leider keine Aussage möglich.

Das Konzept von BuildHLRT verspricht viel, kann aber leider derzeit (noch) nicht viel halten. Das Problem, dass die Dokumentgröße exponentiell in die nötige Anzahl Beispiele eingeht, ist eine sehr starke Eingrenzung der Anwendbarkeit. Laut den Autoren existiert schon eine Implementierung von BuildHLRT, welche allerdings nur auf kleinen, tabellarisch formatierten ASCII-Texten arbeitet, was eine sinnvolle Nutzung im Moment unmöglich macht. Doch auch diese frühe Version ist leider nicht erhältlich, weder käuflich noch als Public Domain Software.

4 Zusammenfassung

Die vorgestellten Wrapper-Frameworks scheinen auf den Ersten Blick die Flut der sinnlosen Informationen aus dem Internet nicht wie von Zauberhand verpuffen zu lassen, während sie gleichzeitig jedem, nur die für ihn relevante Daten, zukommen lassen. Deshalb jedoch den Untergang unserer Zivilisation vorherzusagen, da wir alle im Informationsmüll ersticken werden, ist zu voreilig. Immerhin zwei der Frameworks sind als Implementierung sofort nutzbar. Jedoch ist abzusehen, dass im Zuge der Einführung von Technologien wie Sun Microsystems SunONE und Microsofts .NET

eine ganze Reihe von Wrapper-Frameworks im Stil von W4F erscheinen werden. Bis jedoch ausgereifte, kommerzielle Systeme mit den fortgeschrittenen Techniken von XWrap Elite oder Build-HLRT erhältlich sein werden ist noch ein weiterer Weg. Wer sofort Wrapper für das World Wide Web benötigt und keines der vorgestellten Frameworks benutzen möchte oder kann, dem bleibt anzuraten, seine Software modular zu halten und wenigstens die drei Module für Zugriff, Extraktion und Ausgabe separat zu halten. Auch ein Blick ins Internet kann viel Zeit und Arbeit ersparen, da viele freie Bibliotheken für oft wiederkehrende Aufgabenstellungen beim Wrapper-Bau existieren, z. B. für HTTP- und HTTPS-Zugriffe, HTML-Fehlerbereinigung oder die erwähnten Java-Bibliotheken für die Unterstützung regulärer Ausdrücke [Jreg02], Datenbank- [Jdbc02] und XML-Zugriffe [Jxml02]. Es bleibt jedoch anzumerken, dass Wrapper, welche auf SGML-strukturierten oder reinen Textdokumenten arbeiten, einen anderen Ansatz verfolgen können, als Wrapper, die auf proprietär formatierte Daten zugreifen müssen, wie beispielsweise auf Word- oder PDF-Dateien. Auch Wrapper die direkt per SQL, XQuery o. ä. auf Datenbanken zugreifen, benötigen andere Vorgehensweisen als sie die drei Frameworks bieten können.

Literatur

KuWD97 Kushmerick, N.; Weld, D. S.; Doorenbos, R. B.: "Wrapper Induction for Information Extraction" In: IJCAI, 1997

Kush00a Kushmerick, N.: "Wrapper induction: Efficiency and expressiveness" In: Artificial Intelligence 118, 2000

Kush99 Kushmerick, N.: "Regression testing for wrapper maintenance" In: AAAI/IAAI, 1999

Kush00b Kushmerick, N.: "Wrapper Verification" In: World Wide Web Journal 3, 2000

Xweh01 N. N.: XWRAP Elite Home: http://www.cc.gatech.edu/projects/disl/XWRAPElite/

Xwes01 N. N.: XWRAP Elite Service: http://jupiter.cc.gatech.edu:4026/dataextraction

Xwwa01 N N.: XWRAP Walkthrough, http://www.cc.gatech.edu/projects/disl/XWRAPElite/walkthrough.html

LPHB99a Liu, L.; Pu, C.; Han, W.; Buttler, D.: "Building an Extensible Wrapper Repository System: A Metadata Approach" In: IEEE Metadata Conference, Bethesda, Maryland, 1999

LPHB99b Liu, L.; Pu, C.; Han, W.; Buttler, D.; Tang, W.: "An XML-based Wrapper Generator for Web Information Extraction" In: SIGMOD Conference, Philadelphia, 1999

W4fh01 N. N.: W4F Home: http://cheops.cis.upenn.edu/W4F/

AzSa00 Azavant, F.; Sahuguet, A.: "World Wide Web Wrapper Factory (W4F) User Manual", 2000

AzSa99a Azavant, F.; Sahuguet, A.: "Web Ecology: Recycling HTML pages as XML documents using W4F" In: WebDB'99, 1999

AzSa99b Azavant, F.; Sahuguet, A.: "Building light-weight wrappers for legacy Web data-sources using W4F" In: International Conference on Very Large Databases (VLDB), 1999

AzSa98 Azavant, F.; Sahuguet, A.: "W4F: a WysiWyg Web Wrapper Factory" Technical Report, 1998

Curl01 N. N.: curl Homepage: http://curl.haxx.se/

Trop01 N. N.: Tropea Incorporated Homepage http://www.tropea-inc.com

Regx01 N. N.: Regular Expressions for Java, Regexp http://www.javaregex.com

Berg02 Bergler, S.: "Konzepte von Wrappern", 2002

LiTo01 N. N.: Linux Today http://linuxtoday.com

Comp99 N. N.: Computer Fachlexikon Fachwörterbuch, München, 1999

LiPH00 Liu, L.; Pu, C.; Han, W.: "XWRAP, An XML-enabled Wrapper Construction System for Web Information Sources" In: ICDE 2000, San Diego CA, 2000

Jdbc02 N. N.: http://java.sun.com/products/jdbc/

Jxml02 N. N.: http://java.sun.com/xml/

Jreg02 N. N.: http://jakarta.apache.org/regexp/