Web Services – Einsatzmöglichkeiten für das Information Retrieval im WWW

Von Fabio Tosques & Philipp Mayr

Abstract: Der folgende Beitrag nähert sich dem Fokusthema Web Services aus dem Blickwinkel der Informationsrecherche im World Wide Web. Zu diesem Zweck gehen wir der Funktionsweise heutiger Web Services sowohl theoretisch als auch praktisch auf den Grund. Der Beitrag erläutert dabei die drei wesentlichen Web Services Techniken XML-RPC, SOAP/WSDL und REST. Anhand zahlreicher Beispiele einschließlich konkreter Realisierungen werden Einsatzmöglichkeiten von Web Services für die Suche präsentiert.

1 Einleitung

Das World Wide Web (WWW) steckt voller nützlicher und interessanter Informationen: „The data is out there, the data is free, and the data is extremely interesting". [4] Viele dieser Daten sind jedoch für Menschen aufbereitet (in der Markup-Language HTML kodiert), für Programme sind sie aber nur darstellbar, nicht interpretierbar.

Für fast alle modernen Programmiersprachen finden sich daher Softwaremodule, mit denen es möglich ist, HTML–Seiten nach bestimmten Informationen zu durchforsten und auszuwerten. Diese Methode der Auswertung wird auch als *screen scraping* bezeichnet. *Lincoln Stein* bezeichnet diese Methode (vgl. [12]) als „mediaeval torture" und begründet seine Aussage folgendermaßen:

> „Screen scraping is despised for various reasons. First and foremost, it is brittle. Database managers are always tinkering with the user interface, adding a graphic here, moving a button there, to improve the user experience. Each small change in a popular web page breaks dozens of screen–scraping scripts, causing anguished cries and hair–tearing among the bioinformaticists who depend on those scripts for their research of the wet labs they support. Second, it is unrealiable. There is no published documentation of what a data source's web pages are supposed to contain […] Finally, there is massive duplication of effort." [12]

Auch wenn *Stein* sich in diesem Artikel insbesondere auf den Bereich der Bioinformatik bezieht sind seine Aussagen bezüglich des *screen scrapings* auch für andere Bereiche der Datengewinnung im WWW gültig, besonders auf jene des Information Retrieval.

Eine Suche im Perl–Modul–Archiv *CPAN* (http://www.cpan.org) ergibt eine Fülle von Modulen, die das *screen scraping* unterstützen und erleichtern sollen:

- Yahoo! : 65 Module
- Google: 55 Module
- Amazon: 46 Module

Aber auch mit diesen Modulen treten die von *Stein* adressierten Probleme auf. Um also Nutzern die Tortur des *screen scrapings* zu ersparen und damit auch die eigenen Server zu entlasten, kam die Idee auf, wohldefinierte Schittstellen zu entwickeln, mit denen Programme untereinander kommunizieren und Daten austauschen können. Hilfreich zur Seite stand bei neueren Entwicklungen die *Extensible Markup Language* (XML), eine Spezifikation, mit der Datensätze nach ihrem Inhalt gekennzeichnet werden können (ein Vorname als Vorname, eine ISBN–Nummer als ISBN–Nummer, eine URL als URL, usw.). Möchte nun ein Unternehmen oder eine Institution, dass Entwickler kontrolliert auf ihre Daten zugreifen können, so müssen Schnittstellen veröffentlicht werden, die so genannten *Application Programming Interfaces* oder kurz A-PIs.

2 Web Services – Entwicklungsstufen

Die Idee, dass Computer Daten untereinander austauschen ist alles andere als neu. Web Services, wie wir sie heute kennen, sind damit nur die konsequente Weiterentwicklung verschiedener Ideen und Entwicklungsstufen, die im Folgenden kurz zusammengefaßt werden sollen.

Eines der ersten dieser Kommunikationsprotokolle war das von Sun entwickelte *Remote–Procedure–Call*–Protokoll (Sun–RPC), ein Standard, der es Clients erlaubt, auf einem Server bestimmte Prozesse aufzurufen. Um Interoperabilität zwischen verschiedenen Plattformen und Betriebssystemen zu gewährleisten, entwickelte Sun ein standardisiertes Binärformat.[1]

In einem nächsten Schritt entwickelte Microsoft das *Component Object Model* (COM). COM basiert auf Sprachunabhängigkeit, ist interoperabel und achtet streng auf Weiterverwendung der einzelnen Komponenten. Während RPC nur Prozeduren aufrufen kann, existiert COM in einer Welt von Objekten und Methoden. Mit *Distributed Component Object Model* (DCOM) wurde COM zwar weiterentwickelt, es konnte sich

[1]Web Services nutzen heute dafür XML, ein textbasiertes Format, welches sowohl die Fehleranalyse, als auch das Debugging erleichtert und damit den Einstieg erleichtert.

aber nie wirklich durchsetzen.

Wirklich Konkurrenz bekam DCOM und sein Nachfolger COM+ von CORBA (Common Object Request Broker Architecture), das ähnlich funktioniert wie COM.

Die bisher genannten Lösungen besitzen relativ hohe Einstiegshürden und sind somit nur für Spezialisten zugänglich, was ihrer Verbreitung im Wege stand. Es konnten aber Wissen und Erfahrungen gesammelt werden, die der Implementierung heutiger Web Services nützen sollten.

3 Web Services in Theorie und Praxis

Als Kerntechnologie heutiger Web Service Entwicklungen ist die Extensible Markup Language (XML) zu nennen. Ein wohlgeformtes und gültiges XML–Dokument macht aber noch lange keinen Web Service. Für die Entwicklung eines Web Services wird zusätzlich eine Technologie benötigt, mit der die XML–kodierten Daten zwischen Client und Server ausgetauscht werden können. Eine weitere Forderung verlangt, dass dieser Austausch in standardisierter Form geschieht. Schließlich sorgt ein XML–Dokument, das von einem Rechner gesendet wird, noch lange nicht dafür, dass ein Rechner, der das Dokument empfängt, dieses korrekt interpretieren kann.

Im folgenden sollen drei wesentliche Techniken dargestellt werden, mit denen Web Services, sprich der standardisierte Austausch von XML–kodierten Daten, heute realisiert werden. Dabei führt ein kurzer theoretischer Teil die konzeptionellen Grundlagen ein, dem dann konkrete Realisierungen in Form von Beispielanwendungen folgen. Auch wenn der theoretische Teil etwas tiefer in technische Details vordringt, müssen sich zukünftige Entwickler der vorgestellten Web Services nicht unbedingt mit diesen Details aufhalten, da moderne Integrationen der Services in Programmierumgebungen wie .NET, J2SE bzw. J2EE oder Module für Perl, Python, PHP usw. diese Details verstecken, damit die Entwickler sich ganz auf die eigenen Anwendung konzentrieren können.

3.1 XML–RPC

3.1.1 Die Theorie

XML–RPC war die erste konkrete Realisierung eines Web Services, der RPC–Aufrufe über das HTTP–Protokoll mit XML–Daten realisiert. Zwar bietet XML–RPC lange nicht die Möglichkeiten, die beispielweise CORBA bietet oder die Weiterentwicklung SOAP, aber es ist dafür einfacher zu implementieren.

XML–RPC wurde in erster Linie von *Dave Winer* (Userland Software Inc.) entwickelt, eine erste Implementation war 1998 im Content Management System *Frontier* enthalten. Mit der Offenlegung der Spezifikation sollten andere ermutigt werden, XML–RPC in eigenen Anwendungen zu verwenden. Fast alle gängigen Programmiersprachen enthalten heute die Möglichkeit, XML–RPC–Anwendungen zu realisieren.

Die Funktionsweise von XML–RPC ist relativ einfach: der Datenaustausch zwischen Client und Server erfolgt mit XML–kodierten Daten. Abbildung 1 stellt schematisch den Zyklus einer Anfrage mit XML–RPC dar.

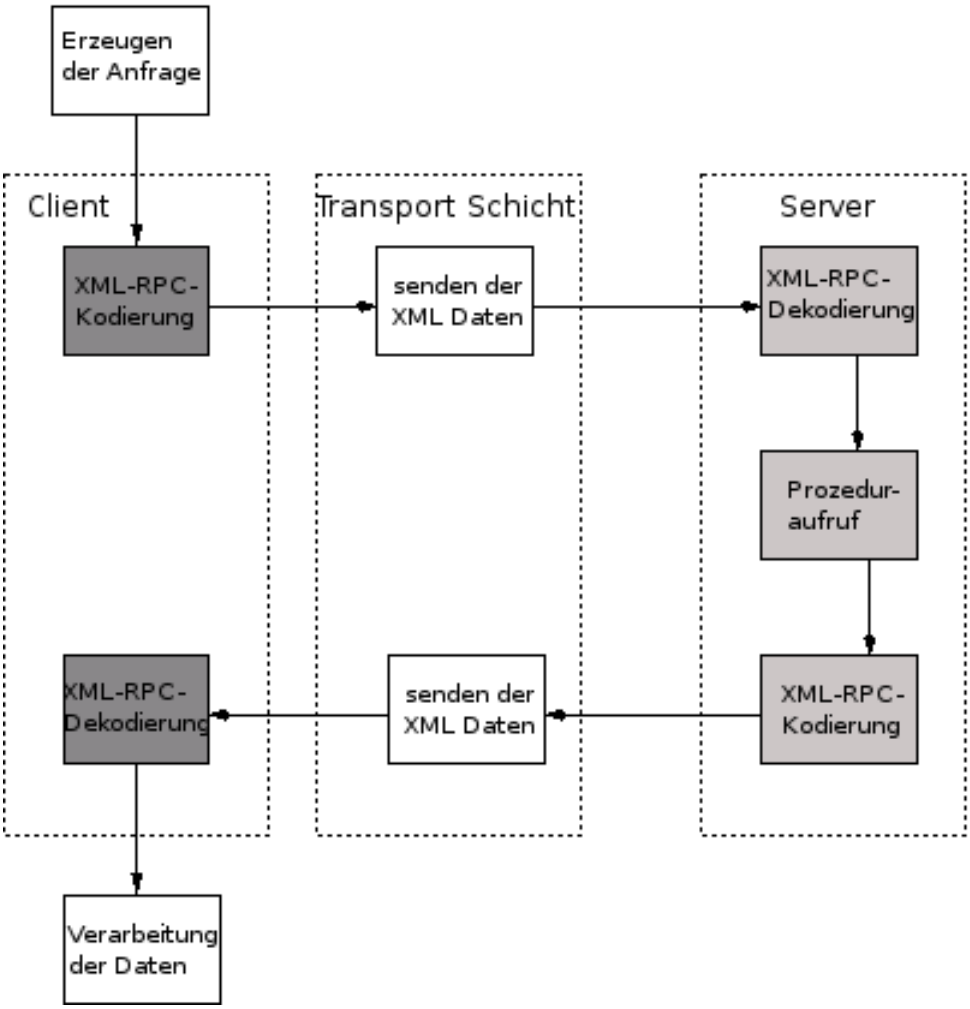

Abbildung 1: Zyklus einer XML-RPC-Anfrage

Die Einfachheit der Spezifiktion wird dadurch garantiert, dass nur sieben einfache Datentypen und zwei erweiterte Datentypen genutzt werden können.

Die XML–RPC–Spezifikation erlaubt somit eine einfache Darstellung von RPC–Aufrufen. Obwohl das Protokoll sehr einfach gehalten wurde, werden die wesentlichen Anforderungen solcher Aufrufe abgedeckt. Die Beschränkung auf ein Transportprotokoll (HTTP) und einer Betriebsart (RPC) vereinfacht die Anforderungen an eine Implementierung.

Die Vorteile bezüglich der Einfachheit bringt die Spezifikation des Protokolls schnell an seine Grenzen. *Hein und Zeller* (vgl. [8]) finden dafür Gründe in den Datentypen und in der fehlenden Metabeschreibung:

- Die Codierung der Datentypen lassen Interpretationen zu oder sind zu ungenau.
- Es lassen sich keine zyklischen Datenstrukturen abbilden, da die Datenstruktur,

die als XML–Dokument übertragen wird, ein XML–Baum ist, der keine zyklischen Beziehungen abbilden kann.

- Binäre Daten lassen sich nur über ein base64–Encoding übertragen.
- Es gibt keine Meta–Beschreibung der Aufrufe.

XML–RPC zeigt den richtigen Ansatz. Daher wurden Anstrengungen unternommen, die noch verbleibenden Nachteile in einer erweiterten Spezifikation zu adressieren. Schließlich landete die Spezifikation beim W3C–Konsortium und es wurden eine Vielzahl von möglichen Low–Level–Protokollen, Datentypen und Betriebsarten hinzugefügt. Die erweiterte Spezifikation wurde unter der Bezeichnung *Simple Object Access Protocol* (SOAP) vom W3C veröffentlicht.

3.1.2 Beispielsanwendung: Meerkat

> „Meerkat is a Web-based syndicated content reader. It is based on Rich Site Summary (RSS), an XML specification used for distributing news, product announcements, discussion threads, and other assorted content as channels. Meerkat provides a simple interface to these stories. While maintaining the original association of a story with a channel, Meerkat's focus is on chronological order – the latest stories float to the top, regardless of their source."[2]

Wir haben nun verschiedene Möglichkeiten uns mit Hilfe von *meerkat* Informationen anzuzeigen. Eine Möglichkeit besteht in der direkten Nutzung des Webinterfaces unter http://www.oreillynet.com/meerkat/index.php. Dort kann sich jeder ein eigenes Profil erstellen.

Die zweite Möglichkeit besteht darin, sich mit einem Programm, das XML–RPC fähig ist, nur die Daten zu holen und sich beispielsweise auf einer Webseite oder in einem anderen Reader anzeigen zu lassen. Wir haben uns beispielsweise ein Programm geschrieben, das uns täglich eine Zusammenfassung der zehn Top-Stories aus den Kanälen, die sich mit Web Services befassen, anzeigt. (FT: Link zum Programm einfügen!)

3.2 SOAP/WSDL

3.2.1 Die Theorie

Das *Simple Object Access Protocol* (SOAP) ist die vom W3C empfohlene und entwi-

[2] http://www.oreillynet.com/pub/a/rss/2000/03/17/about_meerkat.html

ckelte Spezifikation für die Implementierung von Web Services.[3]

SOAP ist wesentlich flexibler als XML–RPC und bietet ausreichend Raum für individuelle Erweiterungen. Es nutzt viele der intelligenten Lösungen, die XML bietet. Eine SOAP–Nachricht besteht aus:

- einem Umschlag, dem *envelope–tag*, der die Namespaces deklariert.
- einem (optionalen) Header, der Informationen darüber enthält, wie der Service die Nachricht verarbeiten soll.
- einem Textkörper, dem *body–tag*, in dem die Daten und somit die eigentliche Nachricht enthalten ist.

Listing 1 zeigt die allgemeine Form einer in einen Umschlag verpackten SOAP–Nachricht.

```
<?xml version="1.0" encoding="UTF-8" ?>
<soap-env:Envelope
      xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
  <soap-env:Header>
    <!-- Header-Informat ionen -->
  </soap-env:Header>
  <soap-env:Body>
    <!-- Daten der Nachricht -->
    <!-- Fehlermeldungen -->
  </soap-env:Body>
</soap-env:Envelope>
```

Listing 1: Allgemeine Form eines SOAP-Envelopes

Welche Methoden und Variablen der SOAP–Service nutzen kann, ist in einer *Web Service Description Language* beschrieben werden. Welche Bedeutung und Funktion WSDL in einer SOAP–Umgebung hat, entnehmen wir einem Artikel eines IBM–Entwicklers:

> „Web Services Description Language (WSDL) is an new specification to describe networked XML–based services. It provides a simple way for service providers to describe the basic format of requests to their systems regardless of the underlying protocol (such as Simple Object Access Protocol or XML) or encoding (such as Multipurpose Internet Messaging Extensions)." [10]

Der Austausch von SOAP–Nachrichten ist in Abbildung 2 vereinfacht dargestellt.

[3] Beschreibungen der einzelnen Spezifikationen finden sich unter `http://www.w3.org/ws`.

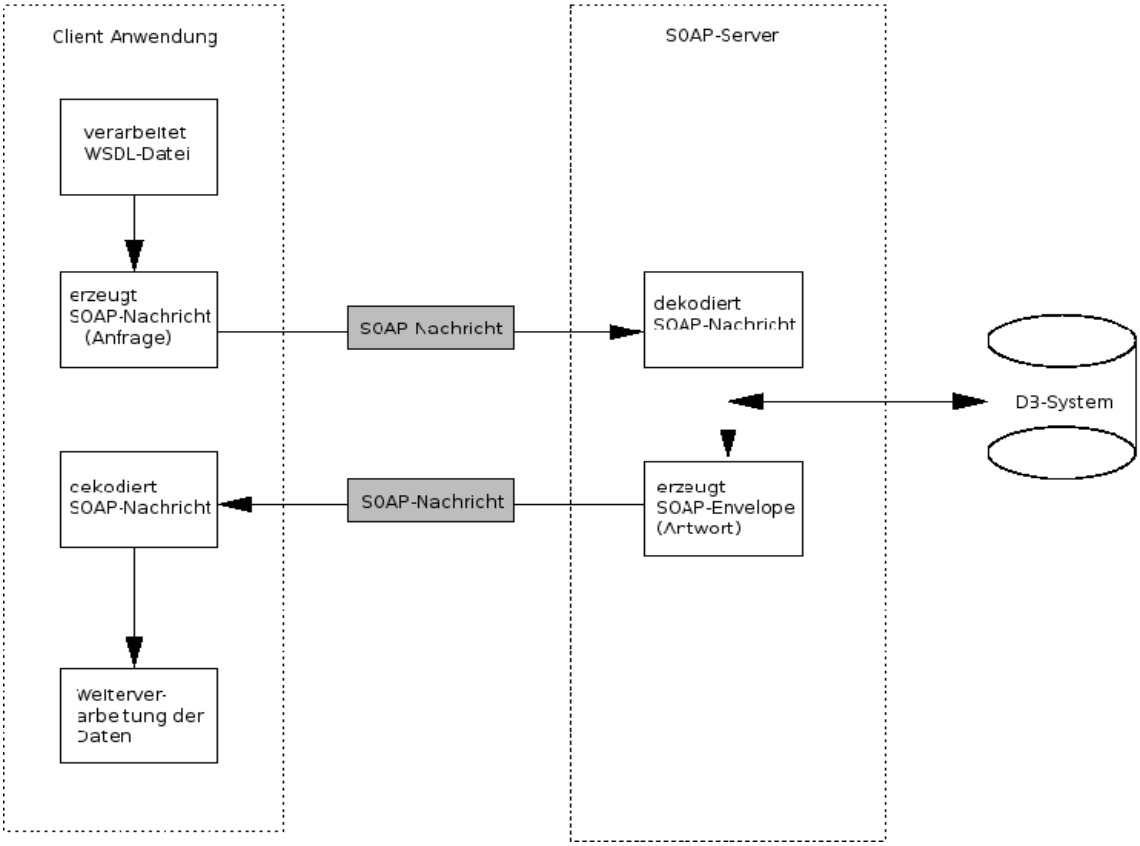

Abbildung 2: Austausch von SOAP-Nachrichten

3.2.2 Beispielanwendungen: Google Web APIs und Amazon Web Services

Google Web APIs

Google verbot von Anfang an die Methode des *screen scrapings* und drohte bei Verstößen gar mit der Sperrung von IP–Adressenräumen.[4] Im Frühjahr 2002 beugte sich aber Google der vermehrten Bitte von Forschern und Entwicklern, den Index automatisiert abfragen zu können. Damit es nun nicht zu unkontrolliertem *screen scraping* kommt, veröffentlichte Google entsprechende Schnittstellen, die *Google Web APIs*, die unter http://www.google.com/apis zum Download bereitstehen. Um diese Schnittstellen nutzen zu können sind zwei Bedingungen zu erfüllen:

[4] „Ohne im Voraus eine Erlaubnis von Google erhalten zu haben, dürfen Sie keine automatisierten Anfragen irgendeiner Art an Googles System senden. Beachten Sie bitte, dass „automatisierte Anfragen" ebenso jede Software einschließt, die Anfragen an Google schickt, um den „Rank" (Qualität) einer Website für unterschiedliche Suchanfragen zu erhalten." (vgl.: `http://www.google.de/terms.html`)

1. der Anwender muß sich registrieren, um einen Googlekey zu erhalten;
2. die verwendete Programmiersprache muß die vom Service angebotenen SOAP–Schnittstellen unterstützen.

Wir haben die Funktionsweise und wesentlichen Elemente des Google Service in einem früheren Artikel kurz beschrieben und möchten diese kurz wiederholen (siehe Mayr & Tosques, 2004)

Die Möglichkeiten des Google Service sind in der WSDL–Datei `GoogleSearch.wsdl` exakt beschrieben. Hier ist definiert, wo der Service sich befindet (die URL), was vom Entwickler implementiert werden kann (Methoden, Variablen, usw.) und wie, d.h. über welches Internetprotokoll und über welche Ports mit dem Service kommuniziert werden kann. Das SOAP–Protokoll dient dann zur Übertragung der XML–kodierten Daten (Nachrichten). Diese besteht wie erwähnt aus einem SOAP–Envelope. In einer Anfrage an den Google Service können die in Tabelle 1 aufgeführten Daten enthalten sein.

SCHLÜSSELWORT	BEDEUTUNG
Key	der Googlekey
query	die Anfrage selbst
start	Offset des ersten Treffers
maxResults	Anzahl der Ergebniss (nur 1 - 10 ist möglich)
filter	Dubletten aussortieren
restrict	die Suche auf einen bestimmten Bereich beschränken
safeSearch	Kindersicherung
lr	Suche auf eine bestimmte Sprache einschränken
ie	input encoding (nur UTF–8 ist möglich)
oe	output encoding (nur UTF–8 ist möglich)

Tabelle 1: Elemente, die in einer Anfrage an den Google Service enthalten sein können.

Die Daten, die der Google Service bei einer erfolgreichen Anfrage schickt, sind ebenfalls in der WSDL–Datei definiert und in Tabelle 2 zusammengefasst. Es handelt sich dabei im Wesentlichen um jene Elemente, die von Google auch per Webinterface geliefert werden. Diese Elemente können nun von einem XML–Parser bequem ausgewertet und in eigenen Anwendungen weiterverarbeitet werden.

SCHLÜSSELWORT	BEDEUTUNG
URL	URL der Ergebnisseite
snippet	kurze Beschreibung des Ergebnisses
title	Titel der Ergebnisseite
cachedSize	Größe der Seite in KB
relatedInformationPresent	ähnliche Seiten vorhanden?
hostName	der Hostname des Ergebnisses
directoryCategory	Eintrag in Google Directory
estimatedTotalResultsCount	Anzahl der Treffer
searchQuery	die Suchanfrage, die geschickt wurde
resultElements	eine Liste der Resultate
searchTips	Korrekturvorschläge
searchTime	Dauer der Suche im Google Index

Tabelle 2: Wesentliche Elemente, die in einer Antwort des Google Service enthalten sind.

Wir haben ein paar Beispielprogramme geschrieben, die eine Auswertung der Google Ergebnisse, die vom Service geliefert werden, vornehmen. Dabei handelt es sich beispielsweise um eine Top–Level–Domainanalyse der URLs, die in den Ergebnissen enthalten sind oder um ein Programm, das die Häufigkeit der Dokumenttypen analysiert und grafisch auswertet. Diese und weitere Programme können unter der URL http://bsd119.ib.hu-berlin.de/~ft/ ausprobiert werden.

Amazon Web Services (AWS)

Von den frei zur Verfügung stehenden Web Services sind jene von Amazon die bisher am besten entwickelten. Laut *Paul Bausch*, dem Autor des bei O'Reilly erschienen Buchs *Amazon Hacks* [2] geht keiner der bisher genannten Services so aggressiv vor wie Amazon:

> „The company has, in essence, outsourced much of its R&D, and a growing portion of its actual actual sales, to an army of thousands of software developers, who apparently enjoy nothing more than finding creative new ways to give Web surfers access to Amazon merchandise and earning a few bucks in the process." [11]

Das Ergebnis sind viele kleine Amazon–Shops, die Kunden zu Amazon führen, die sonst unter Umständen nicht dort gelandet wären. Im Ergebnis entsteht eine Art Sym-

biose, ein Geben und Nehmen: Amazon fordert von den Entwicklern, dass potentielle Kunden zu Amazon geführt werden, die Entwickler bekommen dafür ein paar Prozente für jeden Artikel, der über ihre Webseiten bei Amazon gekauft wurde.

Amazon veröffentlichte die Services am 16. Juli 2002. In der aktuellen Diskussion, welche der beiden Spezifikationen, SOAP oder REST die bessere Lösung ist, wählte Amazon einen Kompromiss: es werden beide Verfahren in gleicher Weise unterstützt.

Auf welche Daten kann nun zugegriffen werden? Im Grunde auf alle, die auch auf der Amazon Seite ersichtlich sind, nur in strukturierter Form. Dabei werden zwei Abfragemodi unterschieden: „lite" und „heavy". Die Elemente einer „lite"-Antwort sind in Tabelle 3 enthalten. Eine „heavy"-Antwort enthält wesentlich mehr Daten (vgl. [2]).

BESCHREIBUNG	PFAD (XPATH)
ASIN	Details/Asin
Produktbezeichnung	Details/ProductName
Katalogeintrag	Details/Catalog
Autor(en)	Details/Authors/Author
Erscheinungsdatum	Details/ReleaseDate
Hersteller / Verlag	Details/Manufacturer
Preise	Details/ListPrice, Details/OurPrice, Details/UsedPrice
Image URL	Details/ImageUrlSmall, Details/ImageUrlMedium, Details/ImageUrlLarge
Argumente der Anfrage	Request/Args/Arg

Tabelle 3: Elemente einer Amazon ‚lite' response

Die Antwort von Amazon (XML–Daten) sehen beispielsweise so aus (gekürzt):

```
<?xml version="1 . 0" encoding="UTF-8" ?>
  <ProductInfo>
    <Detailsurl="http://amazon.com/o/0140042598">
      <Asin>0140042598</Asin>
     <ProductName>On the Road</ProductName>
     <Catalog>Book</Catalog>
     <Authors>
       <Author>Jack Kerouac</Author>
     </Authors>
  </Details>
</ProductInfo>
```

Listing 2: Auschnitt der XML-Daten einer Amazon Antwort

Beispielanwendungen, welche die Amazon Web Services nutzen:

- Amatex – Amazon meets BibTeX

Amatex wurde im Rahmen einer Diplomarbeit der Abteilung Software Entwicklung der Universität Oldenburg entwickelt. „Amatex soll LaTeX2ε–Anwendern helfen, das mühselige Suchen der Bücherdetails zu beseitigen. Die Idee von Amatex ist es, BibTeX–Einträge automatisch zu erstellen ...“ [1] Für diesen Artikel haben wir Amatex ausprobiert und die Ergebnisse für die Erstellung des Literaturverzeichnises mit BibTeX benutzt. Unter http://www.2ndminute.org:8080/amatex/pages/main.jsp kann die Schittstelle zu Amazon selbst ausprobiert werden.

- Amazon Light 4.0 (http://www.kokogiak.com/amazon4)

  Amazon Light 4.0 ist ein alternatives Webinterface, mit dem auf die Amazon Datenbanken zugegriffen werden kann. Amazon Light 4.0 bietet neben den Möglichkeiten, die Amazon selbst bietet (Kauf des Buches, der CD oder DVD usw. oder Eintragen in Wunschzettel), weitere Features:

  - Send URL to a friend with GMail
  - Set up a DropCash Campaign
  - Send URL and Title to your Blog on Blogger
  - Send URL and Title to your Del.Icio.Us Account
  - Lookup a book at your Local Library

- Eigene einfache Codebeispiele, die den Amazon Web Service nutzen.

  Wir haben unter http://bsd119.ib.hu-berlin.de/~ft/ ein paar einfache Beispiele für die Veranstaltung der DGI abgelegt, an denen der Amazon Web Service leicht zu verstehen ist.

3.3 REST

3.3.1 Die Theorie

Die Web Service Spezifikation *Representational State Transfer* (REST) wurde von *Roy Fielding* in seiner im Jahr 2000 veröffentlichten Dissertation entwickelt und vorgestellt [7]. Die Schlüsselinformation im REST–Modell ist die *resource*. Jede Information, die benannt werden kann ist eine *resource*: ein Dokument, eine Homepage, ein Suchergebnis usw. Ein *resource identifier* identifiziert in der Form einer URI ein bestimmte Ressource, die dann in ihrer Repräsentation betrachtet werden kann: die Web-

seite selbst, die Repräsentation eines Dokuments usw.

Im REST–Modell gibt es daher sechs Datenelemente: *resource, resource identifier, resource metadata, representation, representation metadata, control data*. Die Nutzung der REST–Schnittstelle ist für erste Versuche relativ einfach: Sie geben in einem Browser, der XML–Daten anzeigen kann (z.B. Internet Explorer, Mozilla Firefox) beispielsweise folgende URL ein:

http://xml-eu.amazon.com/onca/xml3?locale=de&t=dummy&dev-t=[token]&type=lite&f=xml&KeywordSearch=hacks

Diese URL stellt eine Ressource dar, in diesem Fall ein Suchergebnis. Einen Ausschnitt der XML–Daten, die Amazon zurücksendet sind in Abbildung 3 dargestellt. Ein ähnliches Ergebnis erhält man, bei einem ersten Versuch mit dem *Yahoo!* Service via REST.

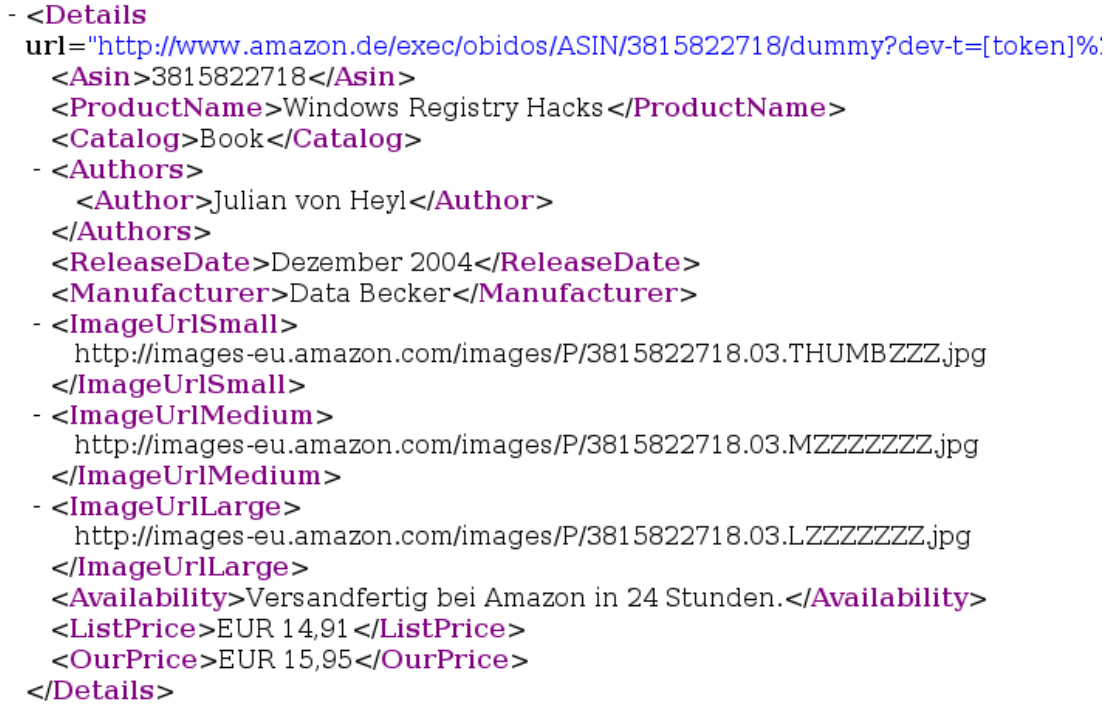

Abbildung 3: Auschnitt einer Antwort von Amazon

3.3.2 Beispielanwendung: Yahoo! Search Web Services

Yahoo! Search Web Services

Die *Yahoo! Search Web Services* (http://developer.yahoo.net) ermöglichen es mit eigenen Programmen auf folgende *Yahoo!* Datenbanken zuzugreifen:

- Image Search
- Local Search
- News Search
- Video Search
- Web Search

Die *Yahoo! Search Web Services* basieren auf REST (Representational State Transfer). *Yahoo!* begründet die Entscheidung für REST wie folgt:

> „Our goal is to make Yahoo! Search Web Services available to as many developers as possible. REST based services are easy to understand and accessible from most modern programming languages. In fact, you can get a fair amount done with only a browser and your favorite scripting language."[5]

Rate Limits sind bei *Yahoo!* IP–basiert, für jede Anwendung, die den Yahoo! Service nutzt, kann eine so genannte *application id* registriert werden, wobei dann pro IP und Anwendung folgende Beschränkung gilt: innerhalb von 24 Stunden können 5000 Anfragen gesendet werden.

Für Entwickler stehen neben der Dokumentation der APIs bei *Yahoo!* folgende Informationsquellen zur Verfügung (vgl. http://developer.yahoo.net):

- Sammlung von Programmen
- Mailing–Liste
- Weblog
- Wiki

Damit bietet der Yahoo! Service weitere Möglichkeiten als der Google Service. In Tabelle 4 sind die wichtigsten Unterschiede zwischen den beiden Web–Service–Schnittstellen aufgeführt.

[5] http://developer.yahoo.net/faq/

	Yahoo	Google
Protokoll	REST	SOAP / WSDL
Registrierung	pro Anwendung	pro Schlüssel
max. Trefferzahl	50.000 pro IP	10000 pro Schlüssel
Zugriffskontrolle	IP–basiert	Schlüssel–basiert
Suchmöglichkeiten	Image Search, Local Search, News Search, Video Search, Web Search	Web Search

Tabelle 4: Unterschiede zwischen dem Yahoo! Service und dem Google Service

Yahoo! bietet Entwicklern des *Yahoo!* Services eine Plattform an, auf der die eigenen Anwendungen veröffentlicht werden können. Unter http://developer.yahoo.net/wiki/index.cgi?ApplicationList finden sich daher eine Vielzahl von Anwendungsbeispielen in verschiedenen Programmiersprachen: .NET, ASP, VB.NET, Java, Ruby, PHP, Perl usw.

Exemplarisch soll hier kurz ein Plugin für den Firefox–Browser vorgestellt werden. Das Plugin blendet eine so genannte Sidebar ein, in der die Überschriften und URLs der ersten 30 Ergebnisse einer *Yahoo!* Websuche angezeigt werden. Eine Beispielsuche nach ,,Information Retrieval" sowie die Ergebnisse in der *Yahoo!*-FirefoxSearchSidebar sind in Abbildung 4 dargestellt.

Abbildung 4: FirefoxSearchSidebar: Beispiel einer Yahoo! Search Anwendung

Auch Amazon bietet die Möglichkeit den Service über die REST–Schnittstelle zu nutzen. Da damit aber grundsätzlich die gleichen Daten angefordert werden können, gehen wir hier nicht weiter auf diese Anfragen via REST ein.

Zum Abschluss versuchen wir die wichtigsten Möglichkeiten der Web Services knapp zusammenzufassen.

4 Zusammenfassung und Ausblick

Wo liegen nun die Vorteile und Möglichkeiten von Web Services für Informationsspezialisten? Mit den oben genannten Beispielen sollte ein erster Eindruck geboten werden, wie frei verfügbare Web Services für die Suche im Web genutzt werden können. Eine Nutzung der angebotenen Services mit eigenen, individuellen Programmen bietet folgende Vorteile:

- Zum einen die Kontrolle des Layouts der Ergebnisse.
- Erweiterte Suchmöglichkeiten können mit Hilfe von Programmen implemen-

tiert werden, beispielweise können Suchwörter permutiert werden oder Suchanfragen können automatisch importiert und exportiert werden.[6] Weitere Anregungen für eigene Anwendungen finden sich in Buch „Google Hacks" (vgl. [5]), inzwischen in der zweiten Auflage erschienen, sowie im Sommer erscheinenden Buch „Yahoo! Hacks" (vgl. [3]).

- Die Weiterverarbeitung der Daten mit eigenen Anwendungen kann zuverlässig gestaltet werden.
- Verschiedene Services können kombiniert werden.

Die oben genannten Vorteile inspirierten beispielsweise *Erik Benson* zur Implementierung von AllConsuming, einer Webseite, die einige der Möglichkeiten heutiger Web Services nutzt.

Die Technik hinter AllConsuming wertet permanent aktualisierte Weblogs nach Hinweisen auf Bücher aus. Dabei werden die Weblogs nach Links zu Amazon, Barnes & Noble, Book Sense und anderen Buchshops durchsucht. Aus diesen Referenzen wird dann eine Art Ranking der „populärsten" Bücher erstellt. Besucher der Seite können sich diese Rankings für verschiedene Zeiträume ansehen: die „populärsten" Bücher von heute, von gestern, dem letzten Monat oder dem letzten Jahr.

Interessant an AllConsuming ist, dass für die Auswertung der Weblogs und für die Darstellung der Informationen auf den Webseiten verschiedene Web–Service–Schnittstellen verwendet werden: *Weblogs.com*, *Amazon.com*, *Technorati.com*, *Alexa.com*, *Google.com* und neuerdings *Yahoo! Search Web Services*. In einem weiteren Schritt bietet der Entwickler von AllConsuming eine eigene Web Service Schittstelle zu seinen Datenbanken an, mit dem die Ergebnisse auf eigenen Seiten eingebunden werden können. Ausführlich beschrieben ist die Idee und Realisierung von AllConsuming in einem Artikel von *Erik Benson* (vgl. [4]).

Eine solche Möglichkeit, verschiedene Datenbanken mit einer individuellen Suchmaske abfragen zu können, fordert auch *Lincoln Stein* für den Bereich der Bioinformatik: „This opens the door to genome 'portals' for those researchers who prefer to access multiple data sources from a single familiar environment." [12]

Wir können uns eine solche Lösung auch für andere Bereiche der Forschung vorstellen. Insbesondere jene Hoster von Datenbanken, die ihre Produkte verkaufen möchten,

[6] Beispiele finden sich u.a. unter http://bsd119.ib.hu-berlin.de/~ft/

sollten sich hier ein Beispiel an Amazon nehmen. Auch wissenschaftlichen Informationsanbietern sollte es im Grunde relativ egal sein, wie ihre Kunden zu den Produkten finden. Hier bieten Web Services sicher noch ein großes Potenzial.

Anderen, wie Google oder Yahoo!, kann die Offenlegung standardisierter Schnittstellen, unter Umständen das Überleben sichern.[7] Zu dieser Überzeugung gelangt beispielsweise *Charles H. Ferguson* in seinem Artikel „What's Next for Google" (vgl. [6]). Möchte Google nicht irgendwann das gleiche Schicksal erleiden wie Netscape, können solche Schnittstellen helfen, Nutzer an sich zu binden. Dies aber scheint Google noch nicht hundertprozentig erkannt zu haben:

> Peter Norvig, the companys [Google, Inc.] director of search quality, told Technology Review, we've had the API project for a few years now. Historically, it's not been that important: is's had one person, sometimes none. But we do think that this will be one important way to create additional search functions. Our mission is to make information available, and to that end we will create a search ecology. We know we need to provide a way for third parties to work with us. You'll see us release APIs as they are needed." [6]

Die Google Web APIs haben ihren Beta–Status folglich auch seit 2002 nicht verloren. Offensichtlich wurde die Weiterentwicklung bisher tatsächlich nicht für sehr wichtig gehalten. Der Druck aber steigt: *Yahoo!* veröffentlicht im Frühjahr 2005 funktionstüchtigere APIs, Microsoft steigt in den Suchmaschinenmarkt ein und Google legt mit Google Code[8] vor kurzem nach. Es bleibt also spannend, rund um das Thema Web Services.

[7] Auf dem Suchmaschinenmarkt ist seit dem Release von Microsoft MSN Beta Search wieder etwas mehr Bewegung gekommen.
[8] Google Code, siehe unter http://code.google.com/.

Literatur

[1] **Abels, S./Uslar, M./Beenken, P.**: Amatex – Amazon meets BIBTEX. Die TEXnische Komödie, 2004, 18–22.

[2] **Bausch, Paul**: Amzon Hacks: 100 Industrial-Strength Tips and Tools. O'Reilly & Associates, 2003.

[3] **Bausch, Paul**: Yahoo! Hacks. O'Reilly, 2005.

[4] **Benson, Erik**: All Consuming Web Services. http://www.xml.com/pub/a/ws/2003/05/27/allconsuming.html – Zugriff am 20.03.2005.

[5] **Calishain, Tara/Dornfest, Rael**: Google Hacks: 100 Industrial-Strength Tips and Tools. O'Reilly, 2003.

[6] **Ferguson, Charles H.**: What's Next for Google. Technology Review, 2005.

[7] **Fielding, Roy Thomas**: Architectural Styles and the Design of Net-work–based Software Architectures. Dissertation, University of California, Irvine, 2000. http://www1.ics.uci.edu/~fielding/pubs/dissertation/top.htm - Zugriff am 18.03.2005.

[8] **Hein, Manfred/Zeller, Henner**: Java Web Services. Addison-Wesley, München, 2003.

[9] **Mayr, Philipp/Tosques, Fabio**: Webometrische Analysen mit Hilfe der Google Web APIs. Information Wissenschaft und Praxis, 2005, 41–48.

[10] **Ogbuji, Uche**: Using WSDL in SOAP application. An Introduction to WSDL for SOAP Programmers. http://www-4.ibm.com/software/developer/library/ws-soap/index.html – Zugriff am 20.02.2005.

[11] **Roush, Wade**: Amazon: Giving Away the Store. Technology Review, 2005.

[12] **Stein, Lincoln**: Creating a bioinformatics nation. Nature, 417 May 2002, Nr. 6888, 119–120.