

Das Semantic Web - Web Service Architekturen

Markus Sinner

sinner@psitronic.de

<http://www.psitronic.de/ti/semanticweb/wsa.pdf>

31.07.2004

Seminar "Semantic Web"

Sommersemester 2004

TU Darmstadt

Fachbereich Informatik

Fachgebiet Integrierte Publikations- und Informationssysteme

Prof. Dr. Erich Neuhold

Betreuer:

Dipl.-Inform. Ingo Frommholz

Dipl.-Inform. Holger Brocks

Inhaltsverzeichnis

1	Einführung	3
1.1	Über dieses Dokument	3
1.2	Motivation	3
1.3	Vorkenntnisse	3
2	Web Service Architekturen	4
2.1	Modelle	4
2.2	Service orientierte Architektur	5
2.2.1	Rollenteilung	5
2.3	Prozess eines Web-Services	6
3	Vorstellung von Software-Lösungen	8
3.1	IBM WebSphere	8
3.1.1	Kurzbeschreibung	8
3.1.2	Installation	8
3.1.3	Dokumentation	8
3.1.4	Software-Entwicklung	8
3.2	Microsoft Web Service Enhancements (WSE)	9
3.2.1	Kurzbeschreibung	9
3.2.2	Installation	9
3.2.3	Dokumentation	9
3.2.4	Software-Entwicklung	9
3.3	SunONE	9
3.3.1	Kurzbeschreibung	9
3.3.2	Installation	9
3.3.3	Dokumentation	9
3.3.4	Software-Entwicklung	10
3.4	Apache Tomcat/AXIS	10
3.4.1	Kurzbeschreibung	10
3.4.2	Installation	10
3.4.3	Software-Entwicklung	10
3.5	Weitere Produkte	10
3.5.1	Novell exteNd	10
3.5.2	BEA Systems WebLogic Server	11
3.6	Vergleich der Lösungen	12
3.7	Performanzfragen	12
4	Web-Service Software-Entwicklung	13
4.1	Voraussetzungen	13
4.2	Entwicklung mit Java und Axis	13
4.2.1	Web-Service Bereitstellung	13
4.2.2	Web-Service Nutzung	14

5	Application Service Provider	17
5.1	Definition	17
5.2	Beispiel: eBay	17
5.3	Beispiel: google	18
5.4	Beispiel: amazon.com	18
6	Fazit und Ausblick	19
6.1	existierende Software	19
6.2	Anbieter von Web-Services	19
6.3	Wie sieht die Zukunft aus?	19
A	Begriffe	23
A.1	Web Service	23
A.2	Semantic Web Service	23
A.3	Java Web Service	23
B	Abkürzungen	24
C	Screenshots	26
C.1	Eclipse Entwicklungsumgebung	26
C.2	Sun ONE Application Framework	27

Kapitel 1

Einführung

1.1 Über dieses Dokument

Dieses Dokument ergänzt den von mir gehaltenen Seminar-Vortrag vom 22.06.2004 im Rahmen des Seminars “Das Semantic Web” [1] im Sommersemester 2004 an der Technischen Universität Darmstadt [2].

Sämtliche Inhalte wurden mithilfe von Open-Source-Anwendungen unter [10, Debian GNU/Linux 3.1-Sarge] erstellt oder verarbeitet. Dieses Dokument und die HTML-Version wurden durch \LaTeX -Programme erstellt.

Eine aktuelle Version finden sie unter <http://www.psitronic.de/ti/semanticweb/wsa/> (HTML) oder <http://www.psitronic.de/ti/semanticweb/wsa.pdf> (PDF-Version).

1.2 Motivation

Diese Ausarbeitung soll Vorschläge und Techniken vorstellen, mit denen man erfolgreich Web Service betreiben kann. Hauptaugenmerk soll auf die zur Zeit existierenden Architekturen und Implementierungsversuche von verschiedenen Herstellern gelegt werden. Der Schwerpunkt liegt auf der Vorstellung existierender Client-Server sowie Middleware-Plattformen auf service-orientierter Basis.

In den ersten Kapiteln werden Grundlagen gelegt (Kapitel 2), anschliessen stelle ich einige Umgebungen zum Bereitstellen von Web-Services vor (Kapitel 3). Am Ende wird beispielhaft ein Szenario zur Erstellung eines Web-Services und anschliessender Nutzung durch einen Client durchgespielt (Kapitel 4), sowie einige ausgewählte Application Service Provider vorgestellt (Kapitel 5).

Abschließend wird in Kapitel 6 ein Fazit gezogen und ein Ausblick gegeben.

1.3 Vorkenntnisse

Zum Verständnis dieser Ausarbeitung sind Kenntnis der Fachbegriffe des Semantic Web förderlich. Eine Übersicht oft benutzter Fachausdrücke und Abkürzungen finden sie in Anhang A. Im Literaturverzeichnis finden sich viele interessante Quellen zur Thematik.

Kapitel 2

Web Service Architekturen

2.1 Modelle

Die W3C Working Group stellt in ihrer [3, Definition von Web Service Architekturen] vier Modelle vor. Bis auf das service-orientierte werden die Modelle hier nur kurz angeschnitten.

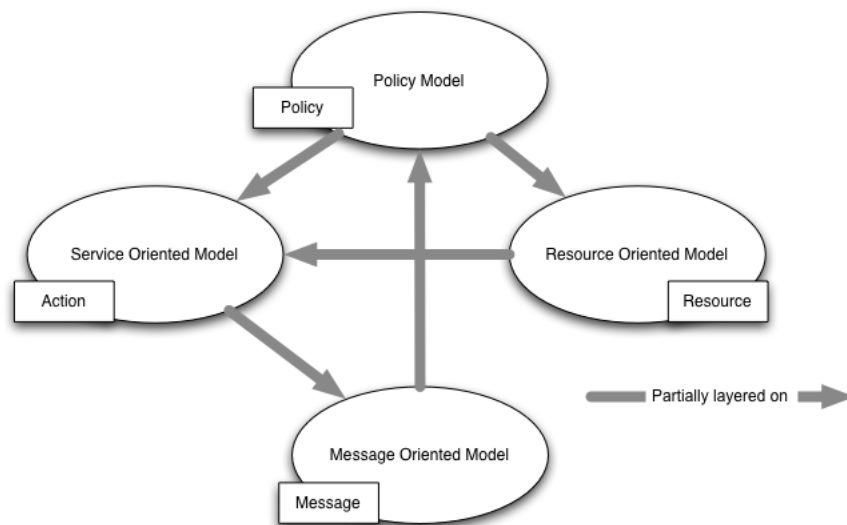


Abbildung 2.1: Das W3C Meta-Modell verschiedener Architekturen
[3, W3C Working Group Note 11 Feb 2004]

- Das Message Oriented Model (MOM)
Hauptaugenmerk in diesem Modell liegt auf dem Verarbeiten von Nachrichten. Eine Nachricht hat einen Sender und einen Empfänger, besitzt eine definierte Struktur und wird auf eine bestimmte Weise übermittelt.
- Das Resource Oriented Model (ROM)
In diesem Modell werden Web Services als eine Resource verstanden. Eine Resource hat einen Uniform Resource Identifier (URI), mit dem sie eindeutig bestimmt ist. Eine Person oder Organisation besitzt eine Ressource.
- Das Policy Model (POM)
Policy, übersetzt "Politik" oder auch "Verfahrensweise", regelt Beziehungen auf der Basis von "Erlaubnissen" (permissions). Sicherheit (security) und Qualitätsmerkmale (quality of service) werden in diesem Modell ebenfalls gut abgebildet. Daher kann in diesem Modell Sicherheit am effizientesten modelliert werden.

- Das Service Oriented Model (SOM)

Dieses Modell ist wohl dasjenige, das die größte Parallele zur realen Welt darstellt. Services werden entwickelt, um der Welt Funktionalitäten bereitzustellen. Der Schlüssel zum Erfolg dieses Ansatzes ist eine gute Beschreibung und Verbreitung der verfügbaren Services. Im Folgenden wird dieser Ansatz weiterverfolgt.

2.2 Service orientierte Architektur

“SOA is about thinking of your system as independent pieces that provided services instead of components that provide methods, regardless of how those services are exposed. Thinking in terms of services gives you a much better handle on what’s important in your system and what are implementation details” – Chris Sells

Diese Zusammenfassung beschreibt in wenigen Sätzen, was man unter einer Service orientierten Architektur versteht. Das System wird nunmehr als Ansammlung von Teilen angesehen, die Dienste bereitstellen. Dies löst die Sichtweise von Komponenten ab, die Methoden bereitstellen.

Web-Services sind in der Regel Client-Server Anwendungen. Die Kommunikation geschieht *service-orientiert* mittels so genannter SOAP-Nachrichten, die zumeist in HTTP-Pakete verpackt werden. Die Übertragung übernimmt in der Regel ein TCP/IP-Netzwerk, meist das Internet. Netzwerk-seitig ist die Kommunikation standardisiert und plattformunabhängig.

Der Client öffnet eine Verbindung, indem er einen so genannten *SOAP-Request* an den Web-Service sendet. Als Antwort erhält er dann einen *SOAP-Response*.

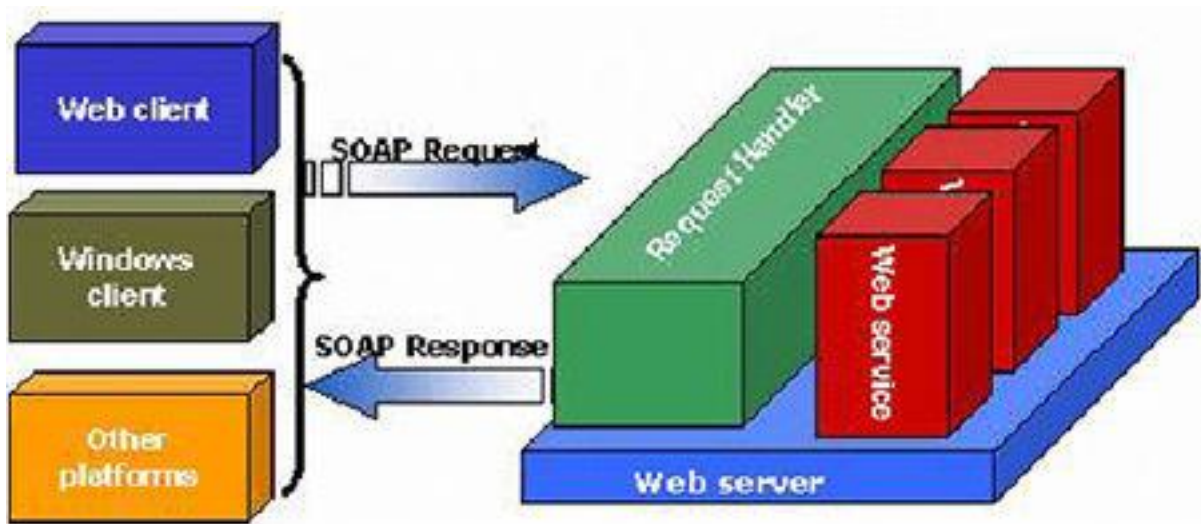


Abbildung 2.2: Web Services auf Service-orientierter Basis
[4, SURPRISE 2002 - Web services based on XML]

2.2.1 Rollenteilung

Eine service-orientierte Architektur sieht eine Rollenteilung vor. Ein Server, der *Service-Provider*, bietet dem *Service-Requestor*, z.B. einer Endanwendung oder einem weiteren Web-basierten Dienst seine Dienste an. Der Client ruft dabei die gewünschte Funktion des Servers auf und bekommt nach erfolgreicher Bearbeitung das Ergebnis zurückgesandt. Beispiele für einfache Web-Services sind Wetterberichte, die nach Angabe einer Koordinate oder eines Ortes dann einen Wetterbericht zurückliefern.

In der nachfolgenden Grafik ist schematisch dargestellt, wie die drei beteiligten Instanzen miteinander interagieren.

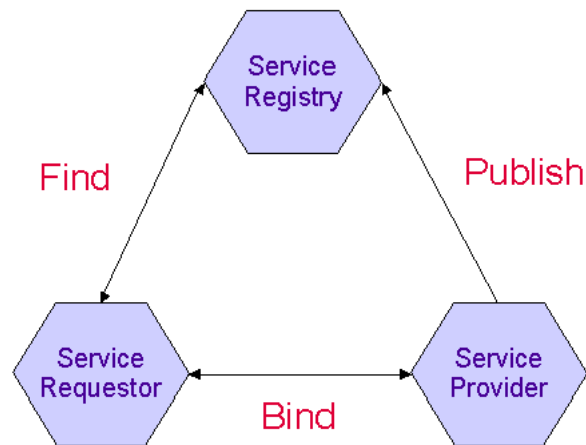


Abbildung 2.3: Rollenteilung
[6, An Introduction to Web Services]

Service Requestor

Ein *Service-Requestor* kann eine Person, eine Anwendung oder ein Agent sein, die bzw. der einen Service in Anspruch nehmen möchte. In der Regel sind bei Web-Service-Architekturen die Service-Requestor Anwendungen oder Agenten. Personen bedienen dabei die Anwendungen, welche die Services dann in Anspruch nehmen. Für einen Service-Requestor treten folgende Fragen auf:

1. wie finde ich (als Person) den benötigten oder einen ähnlichen Dienst?
2. wie findet eine Anwendung oder ein Agent einen benötigten Dienst?
3. welche Informationen muss ich an den Dienst übermitteln und mit welchen Antworten habe ich zu rechnen?
4. wie muss mit der Schnittstelle des Dienstes kommuniziert werden?

Antworten auf diese Fragen: Google, UDDI, SOAP, WSDL B :-)

Service Provider

Der *Service-Provider* ist eine Software, die einen Web Service zur Verfügung stellt. Die Kommunikation verläuft über SOAP-Nachrichten zwischen den Beteiligten.

Zur Beschreibung der Schnittstelle eines Web-Services wurde WSDL entwickelt. Dieses XML-Dokument beschreibt genau, welche Parameter der Service benötigt und welche Rückgabewerte er liefert.

Beispiel: [27, Euro 2004 Tournament Schedule Web Service]

Service Registry

Beispiel: [26, <http://www.xmethods.com>]

Unter einer *Service Registry* versteht man einen Verzeichnisdienst, der dem Service Requestor Informationen über Service Provider bereitstellt. In der Regel stellt die Service Registry Such-Mechanismen zur Verfügung, die eine automatisierte Suche nach benötigten Diensten zulassen.

Als Protokoll zur Suche nach Services wurde UDDI B zum Veröffentlichen von Verzeichnissen entwickelt.

2.3 Prozess eines Web-Services

In der Nachfolgenden Grafik wird veranschaulicht, wie ein Web-Service entsteht. Vom Design bis zur Benutzung des Services werden einige Schritte durchlaufen.

Der *Service Client* versucht zunächst, bei einer *UDDI Registry* einen geeigneten Dienst ausfindig zu machen. Danach wird ein *SOAP-Client* erstellt, der mit dem aufgefundenen Dienst kommunizieren soll. Die Programmiersprache, Plattform und das Betriebssystem, mit der dieser Client erstellt wird, spielen dabei keine Rolle.

Um sich über die Art und Weise der Kommunikation zu einigen, hat der *Service Provider* für seinen Dienst eine Spezifikation der Schnittstelle in WSDL zur Verfügung gestellt. Durch die Registrierung bei der *UDDI Registry* sind diese Informationen für den *Service Client* transparent verfügbar. An dieser Stelle sind alle Vorkehrungen zur Erschaffung einer Infrastruktur erfüllt, und der Client kann nun mit dem Service des Providers über SOAP kommunizieren.

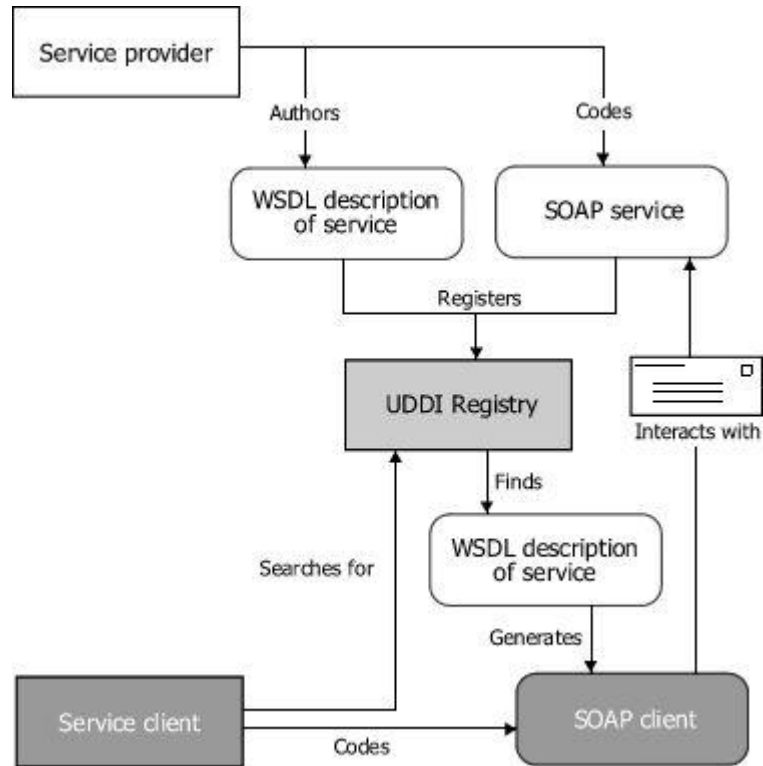


Abbildung 2.4: Web Service Prozess
[4, SURPRISE 2002 - Web services based on XML]

Kapitel 3

Vorstellung von Software-Lösungen

Im folgenden Kapitel werden Software-Lösungen zum Einrichten und Betreiben von Web-Services beschrieben. Ich habe versucht, die Beurteilung der mir vorliegenden Produkte objektiv und nach einem gewissen Schema durchzuführen, jedoch ist mir dies nicht bei allen vorgestellten Produkten gleich gut gelungen. Diese Liste erhebt keinen Anspruch auf Vollständigkeit.

3.1 IBM WebSphere

3.1.1 Kurzbeschreibung

IBM WebSphere ist eine Lösung zur Entwicklung von “Enterprise Web Services for e-Business”, Zielgruppe sind Großunternehmen. Die Anschaffungskosten für den Betrieb von WebSphere sind nicht zu unterschätzen, alleine für Lizenzen werden Preise von ca. 10.000 bis 15.000 EUR genannt. Dazu kommen selbstverständlich die Kosten für Hardware, welche ebenfalls einen Großteil der Gesamtkosten ausmachen, denn IBM sieht für den Betrieb ihres Produktes leistungsfähige Server vor.

3.1.2 Installation

Auf der Homepage von IBM [13] ist eine Demoversion des Produktes erhältlich. Die Installation gestaltete sich unter Linux und Java SDK 1.4.2 [20] problemlos. Anders verhielt es sich im Betrieb, die Funktionen und Möglichkeiten der Konfigurierbarkeit offenbaren sich dem Nutzer nicht sofort.

3.1.3 Dokumentation

Die mitgelieferte Dokumentation ist unübersichtlich, und obwohl Dokumente in mehr als zehn Sprachen angeboten werden enthalten diese lediglich englischen Text. So dauert es einige Stunden bis man in der Lage ist, die Administration des Produkts in den Griff zu bekommen.

Eine Einführung in Web-Services findet man unter <http://www-136.ibm.com/developerworks/webservices/>

3.1.4 Software-Entwicklung

Für die Entwicklung von Web-Services bzw. von Anwendungen zur Nutzung von Web-Services unter WebSphere schlägt IBM das WebSphere Studio [14] vor. Die Technologie basiert auf Eclipse [15] und stellt somit ein mächtiges Werkzeug zur Erstellung von mehrsprachigen, plattformübergreifenden Anwendungen dar. Siehe auch [Screenshot|eclipse](#) im Anhang.

3.2 Microsoft Web Service Enhancements (WSE)

3.2.1 Kurzbeschreibung

Web Service Enhancements[16] ist eine Erweiterung von Microsoft Entwickler-Werkzeugen durch Funktionen zum Gestalten bzw. Arbeiten mit Web-Services. Diese Lösung integriert sich in laufende Microsoft IIS Installationen und erweitert diese um Web-Service-Funktionalität. Zielgruppe sind daher wohl Kunden, die bereits eine IIS-basierte Infrastruktur am Laufen haben.

3.2.2 Installation

WSE setzt auf den IIS auf. Aus diesem Grund wird eine Installation von *Windows 2000*, *Windows Server 2003* oder *Windows XP* benötigt. Zur Entwicklung von Web-Services wird *Microsoft Visual Studio*® .NET und die .NET class library (C# oder Visual Basic) benötigt.

3.2.3 Dokumentation

Die Dokumentation ist vorbildlich, überschaubar und gut lesbar. Eine Probe-Installation habe ich jedoch nicht vorgenommen, weil mir die benötigten Grundvoraussetzungen zu horrend erschienen und benötigte Lizenzen garnicht zur Verfügung standen. Microsoft bietet eine umfangreiche [17, Online-Bibliothek] zum Thema Web-Services an.

3.2.4 Software-Entwicklung

Die Entwicklung entsprechender Anwendungen wird unter *Microsoft Visual Studio*® .NET betrieben. Durch die gute Online-Dokumentation sollte sich eine Einarbeitung nicht allzu schwierig gestalten - vorausgesetzt man beherrscht die zugrunde liegenden Techniken wie C# oder Visual Basic. Aufgrund des horrenden Aufwands zur Installation der benötigten Komponenten habe ich auf eine detaillierte Erörterung verzichtet.

3.3 SunONE

3.3.1 Kurzbeschreibung

SunONE [18] ist weniger ein Produkt, sondern vielmehr eine Philosophie. Hinter dieser Marke SunONE verbirgt sich eine Software-Infrastruktur zur Bereitstellung von Services im Web. Die verschiedenen Komponenten bauen allesamt auf dem hauseigenen Java auf und sind auf Zusammenspiel getrimmt. Früher nannte sich dieses Konzept "Sun ONE middleware and developer products".

Einsatzgebiet sind vor allem große ERP-Systeme. Für kritische Anwendungen hat Sun eine Hochverfügbarkeitslösung als Basis vorgesehen.

3.3.2 Installation

Unter <http://www.sun.com/software/download/index.html> findet man einiges an Produkten zur Installation auf verschiedensten Systemen und unter verschiedensten Betriebssystemen. Solaris- und Linux-Software auf SPARC- oder x86-Hardware werden von Sun bevorzugt genannt.

Serversoftware ist unter http://www.sun.com/software/products/appsrvr/home_appsrvr.html zu finden.

3.3.3 Dokumentation

Auf http://docs.sun.com/db/coll/s1_wspde_en sind einige Dokumente zum Einarbeiten in SunONE veröffentlicht. Die Fülle an Informationen ist erdrückend und schwer durchschaubar.

3.3.4 Software-Entwicklung

Zur Entwicklung von Anwendungen legt Sun sein Produkt Sun ONE Application Framework nahe. Auf der entsprechenden Seite können Testversionen heruntergeladen werden. Siehe auch [Screenshot](#) sunonescreen im Anhang.

3.4 Apache Tomcat/AXIS

3.4.1 Kurzbeschreibung

Als einziges Produkt aus der Open-Source-Szene geht Apache Tomcat/Axis [11] ins Rennen. Die Leistungsfähigkeit dieses Produktes kann den kommerziellen Lösungen als ebenbürtig angesehen werden. Die Verfügbarkeit des Quellcodes und die niedrigen Anschaffungskosten mögen für manche Anwendungsfälle ein hinreichendes Argument sein für die Entscheidung zugunsten dieser Lösung. Wie jedoch üblich bei den meisten Open-Source-Projekten ist Support und Gewährleistung eingeschränkt, da kein haftbares Unternehmen dahinter steht.

Die Zielgruppe ist offen. Durch den vorliegenden Quellcode können große Konzerne leicht die Software an ihre Bedürfnisse anpassen. Kleine Firmen oder einzelne Personen können aber genauso Web-Services kostengünstig mit dieser Technik entwickeln.

3.4.2 Installation

Für den Betrieb von Web-Services werden zwei Komponenten benötigt. Zum einen eine installierte Version von Apache Tomcat, und zum anderen die Erweiterung Axis für Apache Tomcat.

Die Installation von Axis gestaltet sich denkbar einfach. Falls man bereits über Erfahrungen mit Apache Tomcat verfügt, muss man lediglich Axis als Web-Applikation für Tomcat installieren. Die Installation von Tomcat an sich ist jedoch auch nicht weiter schwierig, benötigt aber Zeit. Installationsanweisungen für Tomcat 4.1 sind unter <http://jakarta.apache.org/tomcat/tomcat-4.1-doc/RUNNING.txt> zu finden. Für die gängigsten Linux-Distributionen existieren vorgefertigte Pakete.

Anweisungen zur Installation von Axis als Web-Applikation für Apache Tomcat sind unter <http://ws.apache.org/axis/java/install.html> zu finden. Beim vorgehen nach den Installationsanweisungen sollte man beachten, dass der Benutzer, unter dessen Profil Tomcat und Axis laufen, Voll-Zugriff auf das Verzeichnis hat, unter dem man die Dateien zu Axis installiert. In der Regel läuft der Server nämlich unter einem speziellen virtuellen Benutzer (bei Debian der Benutzer tomcat4), und diesem wird kein Schreibrecht erteilt bei der Erstellung von Verzeichnissen als root.

3.4.3 Software-Entwicklung

Unter 4 wird beispielhaft die Programmierung, Nutzung und Anwendung von Web-Services unter Linux/Apache/Tomcat/Axis beschrieben.

3.5 Weitere Produkte

Die hier aufgeführten Produkte seien zur weiteren Recherche kurz angeschnitten. Mit Installation und Dokumentation habe ich mich nicht weiter befasst.

3.5.1 Novell exteNd

Auf der Novell Homepage gelangt man über die Such-Option schnell auf die gewünschte Seite zum Produkt *exteNd*. *exteNd* wird mit NetWare 6.5 ausgeliefert und läuft unter einer virtuelle Java Maschine.

Novell exteNd Workbench 4.1.1 und Novell Application Server 5.0 zusammen stellen eine Entwicklungsumgebung für Web Services unter Novell NetWare zur Verfügung.

<http://www.novell.com/collateral/4621359/4621359.html>

3.5.2 BEA Systems WebLogic Server

Die Produkte von BEA Systems zur Entwicklung von WebServices bieten eine gute Abstimmung untereinander und sollen eine reibungslose Entwicklung und Veröffentlichung bieten. Wie viele andere Produkte basiert das System auf einer virtuellen Java Maschine.

Auf der Homepage [19] von BEA Systems kann man eine freie Entwicklerversion der Plattform herunterladen. Aus Zeitgründen habe ich dies jedoch nicht weiter evaluiert.

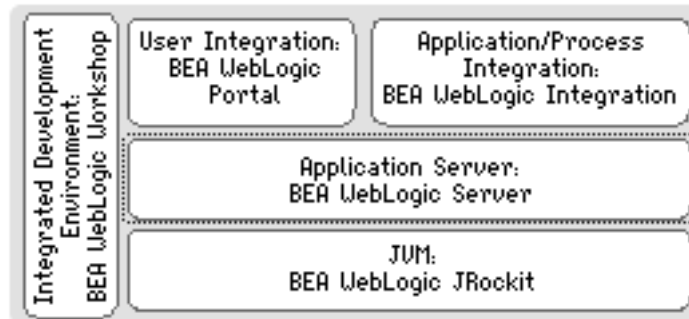


Abbildung 3.1: BEA WebLogic Plattform
[19, BEA Systems Homepage]

3.6 Vergleich der Lösungen

Einige Kriterien der gängigsten Lösungen für Web Service Architekturen sind hier tabellarisch aufgelistet.

	WebSphere	SunOne	WSE	Axis
Hersteller	IBM	Sun Microsystems	Microsoft	(Apache/Tomcat Project)
Zielgruppe	Enterprise	Enterprise	Mittelstand	offen
Programmiersprache	Java	Java	C# / Vis. Basic	Java (/ C++)
Plattformen	++	+	-	++
Dokumentation	-	-	++	+
Anschaffungskosten	sehr hoch	sehr hoch	hoch	moderat
Client-Entwicklung	einfach	?	einfach	einfach
Service-Entwicklung	aufwändig	aufwändig	aufwändig	einfach bis aufwändig

3.7 Performanzfragen

Zu dieser Thematik habe ich hier [9] einen interessanten Artikel gefunden, der sich intensiv mit der Frage nach der Performanz von SOAP-basierten Systemen befasst.

Bei meinen Studien von IBM WebSphere ist mir dessen schlechte Performanz ins aufgefallen. Mein System (AMD Athlon XP 1900+) brauchte für eine einfache mitgelieferte Web-Applikation mehrere Minuten, um überhaupt zu reagieren.

Kapitel 4

Web-Service Software-Entwicklung

In diesem Kapitel möchte ich beispielhaft vermitteln, wie man Web-Services bereitstellen kann und wie man Software entwickeln kann, die Web-Services nutzt.

Die Entwicklung entsprechender Programme wird auf Basis von *Apache/Tomcat Axis* durchgeführt. Da diese Dienste auf Java basieren sollten entsprechende Grundkenntnisse in der Programmierung von Java vorhanden sein, jedoch dürften die Konzepte auch ohne solche verstanden werden.

4.1 Voraussetzungen

Zum reibungslosen Entwickeln sind folgende Punkte zu erfüllen:

- Installiertes und lauffähiges Java SDK 1.4.x [20]. Die Version ihres installierten Java können sie über den Befehl `java -version` herausfinden. `java` muss sich dazu im Pfad befinden.
- Installierte und lauffähige Version von Apache/Tomcat inklusive der Web-Applikation Axis 3.4.2. Für genaue Installationsanweisungen konsultieren sie <http://ws.apache.org/axis/java/install.html>.
- Der Benutzer, unter dessen Profil Tomcat läuft, benötigt Schreibrechte auf das Axis-Verzeichnis. Unter meiner Installation (`/usr/share/java/webapps/axis`) habe ich einfach mit `chown -R tomcat4.users /usr/share/java/webapps/axis` Tomcat diese Rechte zugesprochen.

Ich habe die Entwicklung auf [10, Debain/GNU Linux 3.1 Sarge] durchgeführt.

4.2 Entwicklung mit Java und Axis

Die nachfolgende kurze Einleitung kann detaillierter unter <http://ws.apache.org/axis/java/user-guide.html> nachvollzogen werden.

4.2.1 Web-Service Bereitstellung

Die Entwicklung eines eigenen Web-Services unter Axis gestaltet sich erstaunlich einfach. Hat man die Voraussetzungen erfüllt, dann steht einer Publizierung nichts mehr im Wege. Für einen einfachen Service sind wenige Handgriffe erforderlich.

Code

Dieser kleine Beispiel-Code stellt einen Web-Service bereit, der für zwei Zahlen eine Berechnung durchführt (Subtraktion und Addition). Das Beispiel mag vielleicht wenig nützlich sein, doch demonstriert es die Einfachheit der Programmierung.

Man sollte vor der Veröffentlichung sicherstellen, dass der Code sich kompilieren lässt. Dies kann man durch `javac Calculator.jws` erreichen.

Sie können den Code [hier herunterladen](#).

```
public class Calculator {
    public int add(int i1, int i2)
    {
        return i1 + i2;
    }

    public int subtract(int i1, int i2)
    {
        return i1 - i2;
    }
}
```

Veröffentlichung

Die Veröffentlichung des Web-Services ist denkbar einfach. Man muss lediglich den entwickelten Code als Datei mit der Endung `.jws` A.3 speichern und in das Axis-Verzeichnis (z.B. `/usr/share/java/webapps/axis/`) kopieren.

Die Axis-Installation kümmert sich nun um die Kompilation und um Bereitstellung eines WSDL. Dies geschieht beim ersten Aufruf des Services jedes mal erneut, wenn sich der Code seit dem letzten Aufruf geändert hat.

Unter `http://localhost:8180/axis/Calculator.jws` kann man den Zustand des Services überprüfen (den Port entsprechend ihrer Installation wählen). **Achtung:** Tomcat benötigt Schreibrechte auf das Verzeichnis `axis/WEB-INF/` (siehe auch 4.1).

Komplexere Services

Nun könnte man sich vorstellen einen sehr aufwändigen Algorithmus in Java zu implementieren und auf diese Weise zu publizieren. Dann können Programme über SOAP diesen Algorithmus mit den entsprechenden Parametern aufrufen. Um die Rechenzeit muss sich dann der Server kümmern, auf dem dieser Service läuft.

Die eben vorgestellte Methode zur Veröffentlichung eines Web-Services ist jedoch nicht für komplexe Infrastrukturen geeignet. Packages sind nicht möglich, und man weiß bis zur Ausführung nicht wirklich, ob der Code richtig funktioniert.

Zum Entwickeln komplexer Anwendungen und Infrastrukturen bietet sich *Axis Web Service Deployment Descriptor (WSDD)* an. Weiterführende Dokumentation zu diesem Thema entnehmen sie der Dokumentationsseite unter `http://ws.apache.org/axis/java/user-guide.html`.

4.2.2 Web-Service Nutzung

Die Nutzung des Web-Services gestaltet sich ein wenig aufwändiger: Das folgende Beispielprogramm kontaktiert unseren Web-Service *Calculator* unter `http://localhost:8180/axis/Calculator.jws`. Die Beschreibung in WSDL kann unter `http://localhost:8180/axis/Calculator.jws?wsdl` angeschaut werden. Der einfache Service *Calculator* returniert die Berechnung der beiden übergeben Parameter `"i1"` und `"i2"`, in diesem Fall einen Integer (`"addResponse"` oder `"subtractResponse"`).

Code

Sie können den Code **hier herunterladen**.

```

1 import org.apache.axis.client.Call;
2 import org.apache.axis.client.Service;
3 import javax.xml.namespace.QName;
4
5 public class TestClient {
6     public static void main(String [] args) {
7         try {
8             String endpoint =
9                 "http://localhost:8180/axis/Calculator.jws";
10
11             Service service = new Service();
12             Call call = (Call) service.createCall();
13
14             call.setTargetEndpointAddress( new java.net.URL(endpoint) );
15             call.setOperationName(new QName("http://soapinterop.org/", args[0]));
16
17             Integer ret = (Integer) call.invoke( new Object[] { Integer.valueOf(args[1]), Integer.valueOf(args[2]) });
18
19             System.out.println("Sent '"+args[0]+" '"+args[1]+" '"+args[2]+'', got "' + ret + "'");
20         } // try
21         catch (Exception e) {
22             System.err.println(e.toString());
23         } // catch
24     } // main
25 } // class

```

Zeilen	Beschreibung
1-3	Import von benötigten Klassen
8-9	Festlegung der Adresse, zu der sich der Client verbinden soll
11-12	Hier werden nun ein JAX-RPC-Objekte vorbereitet: "Service" und "Call". Diese Objekte speichern Metadaten und weiter unten dann mit zusätzlichen Parametern gefüllt und ausgeführt.
14	Die in 8-9 festgelegte Adresse wird gesetzt, das Ziel der SOAP-Nachricht
15	An dieser Stelle wird die aufzurufende Methode gesetzt
17	Hier wird nun der eigentliche Aufruf des Services initiiert. Als Parameter wird ein Array von Objekten übergeben, dessen Dimension mit der Definition des Services (WSDL B) übereinstimmen muss. In unserem einfachen Beispiel sind dies zwei Integer, von daher wird ein zweielementiger Array übergeben. Als Ergebnis erhalten wir einen Integer zurück. ¹

¹Ein anderer Service könnte auch einen anderen Typ Variablen zurückgeben. Denkbar wäre z.B. eine Liste von Namen (String) oder Zahlen (Integer). Der Typ des Rückgabewertes hängt von der Definition (WSDL)B des Services ab

Übersetzung und Ausführung

Zunächst muss man dafür sorgen, dass der Java-Compiler die benötigten Axis-Klassen findet (`.jar`). Diese befinden sich unter `/usr/share/java/webapps/axis/WEB-INF/lib/`. Hat mal also den `CLASSPATH` gesetzt, kann man mit `javac TestClient.java` den Client übersetzen.

```
> export CLASSPATH=...
> javac TestClient.java
```

Die Ausführung geschieht ähnlich. Auch hier muss man wieder dafür sorgen, dass die benötigten Klassen auffindbar sind. Nun kann man folgende Befehle (hier beispielhaft) ausführen:

```
> export CLASSPATH=...
> java TestClient add 10 20
Sent 'add 1 2', got '30'

> java TestClient subtract 10 20
Sent 'subtract 10 20', got '-10'
```

Kapitel 5

Application Service Provider

5.1 Definition

Ein Anbieter von mehreren Web-Services wird als *Application Service Provider* bezeichnet. Das Angebot an Services erstreckt sich von simplen Kleinstdiensten (wie Wetter-Prognosen) bis hin zu komplexen “Customer Relationship Management Anwendungen” im E-Business Bereich. In der Regel mieten Firmen (oder auch Personen) den gewünschten Service beim Provider.

Die Vorteile des Mietens oder Kaufens von Services liegen auf der Hand. Das Verteilen von spezialisierten Aufgaben oder Anwendungen an verschiedene Provider spart dem Auftraggeber Entwicklungsaufwand und Kosten. Durch Service-Verträge ist zudem sichergestellt, dass die Dienste verfügbar bleiben und regelmäßig gewartet bzw. angepasst werden. Ein weiterer wichtiger Punkt ist, dass Anschaffung und Wartung von Hardware vermieden wird. Auf der anderen Seite kann der *Application Service Provider* seine Infrastrukturen ständig erweitern und durch Pflege und Weiterentwicklung seinen Kunden einen guten Service liefern. Durch die dadurch entstehende Spezialisierung profitieren auch die Kunden.

Weiterführende Informationen: <http://www.asp-konsortium.de>

5.2 Beispiel: eBay

eBay stellt ein Software Development Kit (SDK) zur Verfügung, über das man eBay Auktionen durch Web-Services in eigenen Anwendungen verwalten kann.

Es existiert einige Client-Software zum Verkaufen und Verwalten von Produkten in eBay basierend auf diesem SDK.

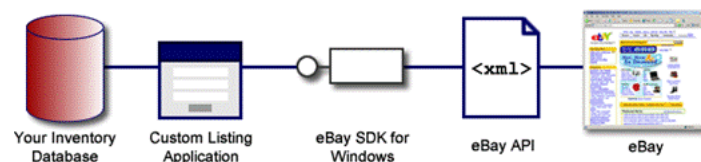


Abbildung 5.1: eBay Software Development Kit
[23, eBay SDK]

5.3 Beispiel: google

Auch die bekannte Suchmaschine Google bietet einen Web-Service an [24].

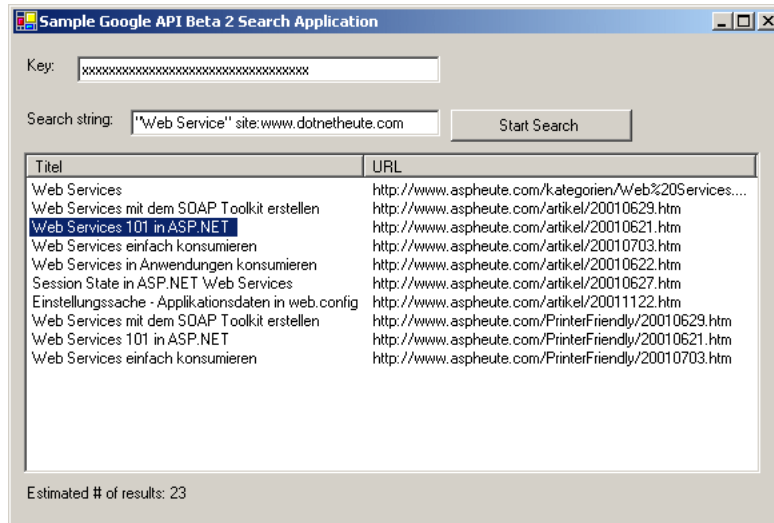


Abbildung 5.2: Programmieren mit den Google Web APIs Beta 2 [25]

5.4 Beispiel: amazon.com

Amazon bietet WebServices an, über die man die Online-Suche nach Büchern und Produkten erleichtern bzw. in eigene Anwendungen einbetten kann. Als Beispiel hier ein Java-Client, der diese Funktionen unterstützt.

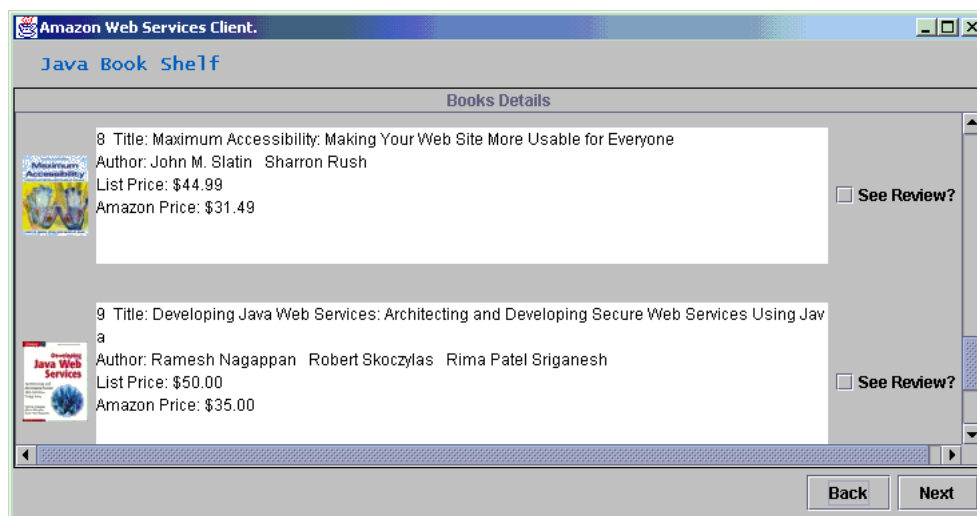


Abbildung 5.3: Amazon Web Services Client [22]

Kapitel 6

Fazit und Ausblick

Zu Beginn meiner Arbeit zu diesem Seminar-Thema war für mich völlig unklar, welche Systeme und Architekturen bereits am Markt etabliert sind.

6.1 existierende Software

Ziemlich alle SAOP-basierten Systeme zum Bereitstellen von Web-Services stützen sich auf Java, eher selten werden C/C++/C# unterstützt. Bei der Client-Entwicklung sieht es da besser aus, für alle möglichen Programmier- und Skriptsprachen gibt es entsprechende Erweiterungen, um über SOAP zu kommunizieren.

Bei den kommerziellen Anbietern ist der Einsatz meist an weitere Produkte gekoppelt, wie z.B. IIS bei Microsofts WSE. Die Entwicklung von Klassen bzw. Modulen wird meist auch nur durch die Entwicklungsumgebungen der Hersteller sinnvoll möglich. Ebenso muß man sich auf gute Vorkenntnisse und lange Einarbeitungszeiten einstellen.

6.2 Anbieter von Web-Services

Erstaunlicherweise gibt es doch schon eine Vielzahl von SOAP-basierten Web-Services, einige konkrete wurden in Kapitel 5 vorgestellt. Ebenso konnte man unter [26, <http://www.xmethods.com>] einen groben Überblick über bestehende freie Web-Services gewinnen.

6.3 Wie sieht die Zukunft aus?

Mir ist immer wieder aufgefallen, wie wenig Bedeutung den Service-Registries zugewiesen wird. Es scheint eine Menge an Service-Providern zu existieren, jedoch ist das Auffinden solcher schwierig. In Zukunft müsste hier meiner Meinung nach Arbeit investiert werden, um die Verbreitung von Information voranzutreiben.

Die Semantik wird größtenteils nicht wirklich genutzt bzw. überhaupt bereitgestellt. Hier sind große Potentiale vorhanden, die es auszuschöpfen gilt. Jedoch glaube ich kaum, dass sich die Bedeutung von Semantic im Bereich Web-Services etablieren wird, solange die großen Hersteller nicht gezielt danach streben und forschen. Für mittelständige Unternehmen dürfte nicht klar sein, dass sich durch Implementierung von semantischen Systemen ein Zukunftsmarkt öffnet, dessen Gewinnpotenziale im Moment überhaupt nicht abschätzbar sind. Die Forschung im Bereich der dazu notwendigen Techniken ist bereits den Kinderschuhen entwachsen, es fehlt jedoch die konkrete Anwendung.

Die Bedeutung von autonomen Agenten im Internet, die über Semantic Web Services untereinander kommunizieren, wird wachsen. Ob es jedoch einen technologischen Durchbruch vom Web zum Semantic Web geben wird wage ich zu bezweifeln. Für Einzel-Anwender, kleine Firmen und Vereine macht es keinen Sinn, die Thematik anzugehen, denn der entstehende Overhead ist viel zu groß. Von daher wird die Technik erst dann massentauglich, sobald ein gewisser Zwang aufkommt, Semantik zu unterstützen - ähnlich wie es sich mit dem WWW zur Zeit verhält.

Literaturverzeichnis

- [1] *“Das Semantic Web - Intelligente Suche auf strukturierten Datenkollektionen”*
Seminar an der Technischen Universität Darmstadt (TUD) im Sommersemester 2004
Fachgebiet Integrierte Publikations- und Informationssysteme
<http://www.ipsi.fraunhofer.de/%7Efrommhol/seminar/SS04/index.html>

- [2] *“Technische Universität Darmstadt (TUD)”*
Karolinenplatz 5
64289 Darmstadt
Telefon: 06151 / 16-0 oder 16-1 (Vermittlung)
Zentralfax (Poststelle) 06151 / 16-54 89
<http://www.tu-darmstadt.de/>

- [3] *“Web Service Architecture”*
W3C Working Group Note 11 Feb 2004
<http://www.w3c.org/TR/2004/NOTE-ws-arch-20040211/>

- [4] *“SURPRISE 2002 - Web services based on XML”*
2002 Timothy Harcourt, Christopher Nesbitt, Woranon Samakoses
<http://www.doc.ic.ac.uk/~crn99/surprise/report/index7.html>

- [5] *“All About Web Services”*
Imran Hameed
http://internet.about.com/library/aa_webservices_031102.htm

- [6] *“An Introduction to Web Services”*
Tracy Gardner
<http://www.ariadne.ac.uk/issue29/gardner/>

- [7] *“<http://webservices.oreilly.com/>”*
2004, O’Reilly Media, Inc.
<http://webservices.oreilly.com/>

- [8] *“Web Services Life Cycle: Managing Enterprise Web Services”*
Sun Microsystems, Inc.
<http://www.sun.com/software/whitepapers/webservices/index.html>

- [9] *“Performancemessung und Vergleich von SOAP-basierten Web Services”*
Prof. Mario Jeckle
Fachhochschule Furtwangen
mario@jeckle.de - <http://www.jeckle.de>
<http://www.jeckle.de/files/SOAPPerf.pdf>

- [10] *“Debian GNU/Linux 3.1-Sarge”*
Deutsche Debian GNU/Linux Homepage
<http://www.de.debian.org/>

- [11] *“WebServices - Axis”*
Apache Tomcat/AXIS Online Documentation
<http://ws.apache.org/axis/>
- [12] *“Web Services Project @ Apache”*
Apache Tomcat/AXIS Online Documentation
<http://ws.apache.org/>
- [13] *“IBM WebSphere Application Server”*
http://www-306.ibm.com/software/info/ecatalog/de_DE/products/U105789N42720B65.html
<http://www-306.ibm.com/software/webservers/appserv/was/>
- [14] *“IBM WebSphere Studio”*
http://www-306.ibm.com/software/info1/websphere/index.jsp?tab=products/studio&S_TACT=103BGW01&S_CMP=campaign
- [15] *“Eclipse”*
eclipse.org
<http://www.eclipse.org/>
IBM WebSphere and Eclipse

<http://www-306.ibm.com/software/info1/websphere/index.jsp?tab=eclipse/index>
- [16] *“Microsoft Web Services Downloads”*
Downloads zum Thema “Web Service Enhancements”
<http://msdn.microsoft.com/webservices/downloads/default.aspx>
- [17] *“Microsoft Web Services Developer Center”*
Microsoft Developer Network (MSDN) Ressource
<http://msdn.microsoft.com/webservices/>
- [18] *“Serious Software Made Simple”*
Sun ONE middleware and developer products)
<http://www.sun.com/software/sunone/>
Sun ONE Application Framework

http://www.sun.com/software/products/application_framework/home_app_framework.html
Sun Java System Application Server

http://www.sun.com/software/products/appsrvr/home_appsrvr.html
Tutorial Section 1.2 - Create Login Page

<http://docs.sun.com/source/817-0446-10/get1tsk2.html>
- [19] *“BEA WebLogic Platform 8.1”*
<http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/platform>
- [20] *“Java SDK”*
Java SDK J2SE
<http://java.sun.com/j2se/>
- [21] *“Wikipedia”*
Wikipedia ist eine mehrsprachige Enzyklopädie, deren Inhalte frei nutzbar sind und es für immer bleiben werden. Die deutschsprachige Ausgabe wurde im Mai 2001 gestartet und umfasst derzeit über 130.000 Artikel. Bei Wikipedia können alle ihr Wissen einbringen.
<http://de.wikipedia.org/>
- [22] *“Developing an Amazon Web Services Client”*
Beth Stearns and Rakesh Garishakurthi
<http://java.sun.com/developer/technicalArticles/WebServices/amazonws/>

- [23] *“Integrate Your Inventory System with the eBay SDK”*
January 12, 2004 by Jeffrey P. McManus
<http://www.devx.com/dotnet/Article/19812/0/page/1>
- [24] *“Develop Your Own Applications Using Google”*
<http://www.google.com/apis/>
- [25] *“Programmieren mit den Google Web APIs Beta 2”*
Christoph Wille
<http://www.aspheute.com/artikel/20020415.htm>
- [26] *“X Methods”*
XMethods was founded in 2000 by Tony Hong and James Hong. It is located in San Jose, California.
<http://www.xmethods.com/>
- [27] *“EUFA - Euro 2004 Tournament Schedule Web Service”*
<http://www.hundhausen.com/euro2004/schedule.aspx>

Anhang A

Begriffe

An dieser Stelle werden Fachbegriffe und Abkürzungen noch einmal kurz beschrieben. Teilnehmer des Seminars [1] können dieses Kapitel getrost überspringen.

A.1 Web Service

Web-Services sind softwarebasierte Systeme, die Interoperabilität zwischen netzwerkfähigen Diensteanbietern möglich machen. Die Kommunikation läuft dabei in maschinenverarbeitender Form ab und ist auch primär zur Verarbeitung durch Maschinen gedacht. Die Kommunikation der beteiligten Systeme ist dabei völlig Plattformunabhängig.

Die Schnittstelle wird durch WSDL beschrieben, Datenaustausch wird mittels SOAP-Nachrichten geregelt. In der Regel sind SOAP-Nachrichten XML Dokumente, die über das HTTP-Protokoll ausgetauscht werden.

A.2 Semantic Web Service

Die Semantik kommt ins Spiel, wenn die beteiligten Dienste selbständig aushandeln und “verstehen”, was sie austauschen.

Semantik ist jedoch nicht Gegenstand dieses Dokuments. Im Rahmen des Seminars wurden jedoch Vorträge angeboten, die sich mit diesen Thematiken beschäftigen.

A.3 Java Web Service

Java Web Service Dateien (Endung `.jws`) ist Java-Quellcode, der bei Anfrage zur einem Web-Service kompiliert wird. Die Architektur kümmert sich dann um Bereitstellung eines WSDL und um Empfang von SOAP-Requests. JWS ist zum Betreiben von sehr einfachen Diensten und zur schnellen Entwicklung gedacht.

Anhang B

Abkürzungen

- **HTTP** *HyperText Transport Protokoll*:

TCP-basiertes Protokoll, das zum Austausch von strukturierten Dokumenten entwickelt wurde. Mittlerweile findet ein Hauptanteil des Datenverkehrs im Internet über HTTP statt.

Das Protokoll wurde 1989 von Tim Berners-Lee am CERN zusammen mit dem URL und HTML erfunden; das World Wide Web (WWW) wurde geboren.

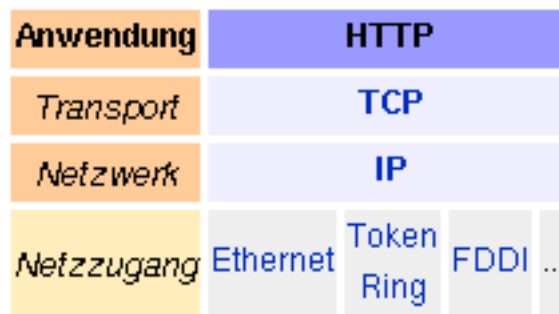


Abbildung B.1: HTTP im TCP/IP-Protokollstapel
[21, Quelle: Wikipedia - HTTP]

- **XML** *eXtensible Markup Language*:

XML bezeichnet einen Standard zur Definition von Auszeichnungssprachen, der als vereinfachte Teilmenge von SGML konzipiert wurde.

Gleichzeitig steht XML in einer losen Verwandtschaft zu HTML, welches ursprünglich (d. h. bis einschließlich zur Spezifikationsversion 4.01) selbst als Anwendung von SGML definiert wurde. Mit der "Extensible HyperText Markup Language" (XHTML) wurde der Übergang zu XML als Definitionsbasis vollzogen. Grund dafür war die einfachere Syntax und damit die Entwicklung einfacherer Parser (die Definition von SGML umfasst 500 Seiten, jene von XML bloß 26).

Die Namen der einzelnen Strukturelemente für eine bestimmte Auszeichnungssprache lassen sich frei wählen, diese Auszeichnungssprachen können dabei alle möglichen Daten beschreiben, als prominentestes Beispiel Text, aber auch Grafiken oder abstraktes Wissen. Ein Grundgedanke hinter XML ist es, Daten und ihre Repräsentation zu trennen. Also beispielsweise Wetterdaten einmal als Tabelle oder als Grafik auszugeben, aber für beide Anwendungen die gleiche Datenbasis im XML-Format zu nutzen.

Weitere Informationen unter <http://www.w3c.org/XML/>.

- **SOAP** *Simple Object Access Protocol*:

Durch das SOAP-Protokoll lassen sich Remote Procedure Calls (RPC) auf entfernten Systemen ausführen und Daten austauschen. SOAP setzt auf einige bereits etablierte Standards auf. Zur Repräsentation der Daten wurde XML gewählt, die Übermittlung der Nachrichten geschieht per TCP/IP. SOAP über HTTP ist der am weitesten verbreitetste Weg zur Übermittlung der Nachrichten. Der Benennung als "Einfaches Objekt-Zugriffs-Protokoll" ist geschichtlich bedingt und wird heute keine große Bedeutung mehr zugesprochen.

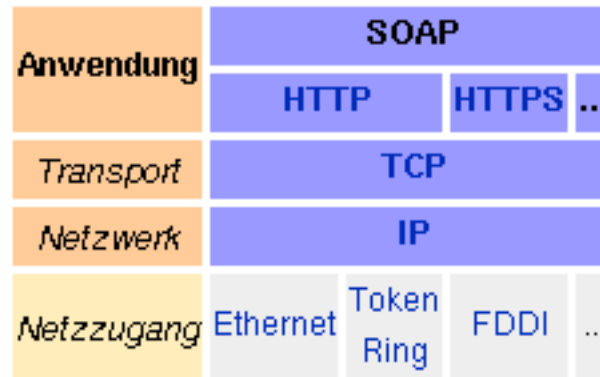


Abbildung B.2: SOAP im TCP/IP-Protokollstapel
[21, Quelle: Wikipedia - SOAP]

Weitere Informationen unter <http://www.w3.org/TR/SOAP/>

- **WSDL** *Web Service Description Language*:

Die Web Services Description Language (WSDL) definiert einen plattform-, programmiersprachen- und protokollunabhängigen XML-Standard zur Beschreibung von Netzwerkdiensten (web services) zum Austausch von Nachrichten.

WSDL ist eine Metasprache, mit deren Hilfe die Funktionalität eines Web Service beschrieben werden kann. Es werden im wesentlichen die Funktionen definiert, die von außen zugänglich sind, sowie die Parameter und Rückgabewerte dieser Operationen.

Am 15. März 2001 veröffentlichte das World Wide Web Consortium die Web Services Description Language (WSDL) Note Version 1.1. Inzwischen ist auch schon Version 2 in einem Draft-Dokument für die Sprachdefinition (core language) und die Nachrichten-Muster (message patterns) verfügbar.

[21, Quelle: Wikipedia - WSDL]

Weitere Informationen unter <http://www.w3c.org/TR/wsdl>.

- **UDDI** *Universal Description, Discovery & Integration Standard*:

UDDI bezeichnet einen Verzeichnisdienst, der die zentrale Rolle in einem Umfeld von dynamischen Web Services spielen soll. UDDI ist eine Anwendung von SOAP. Sie stellt mit Hilfe einer SOAP Schnittstelle einen Verzeichnisdienst bereit. Dieser Verzeichnisdienst enthält Unternehmen, ihre Daten und ihre Services. Dabei kann man in UDDI zwischen drei Arten der Informationen unterscheiden: Den "White Pages", einer Art Telefonbuch, den "Yellow Pages", also die elektronische Entsprechung der gelben Seiten, und den "Green Pages".

Weitere Informationen bei Wikipedia:UDDI und unter <http://www.uddi.org/specification.html>

Anhang C

Screenshots

C.1 Eclipse Entwicklungsumgebung

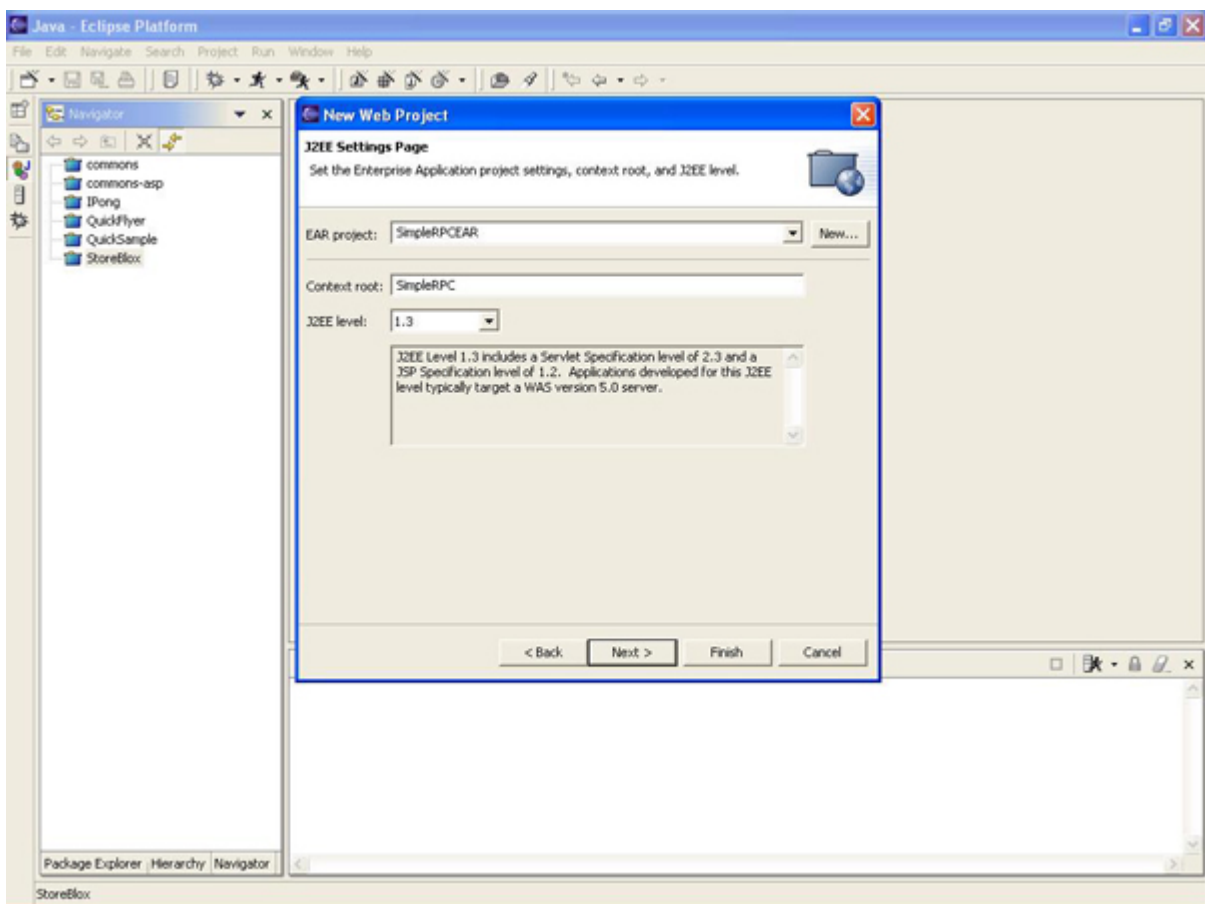


Abbildung C.1: Die Eclipse Entwicklungsumgebung
[15, Eclipse]

C.2 Sun ONE Application Framework



Abbildung C.2: Login Gestaltung mit SunONE Application Framework
[18, Tutorial Section 1.2 - Create Login Page]

Index

- .NET, 9
- Abkürzungen, 23, 24
- Apache, 10
- Application Service Provider, 17
- Architektur, 5
- Array, 15
- asp, 17
- Ausblick, 19
- Axis, 10

- BEA Systems, 11
- Begriffe, 23
- Beispiel, 13

- Client-Server Anwendung, 5

- Entwicklung, 13
- exteNd, 10

- Fachbegriffe, 23

- HTTP, 24

- IBM WebSphere, 8
- Installation, 10
- Instant-Deployment, 13

- JAVA, 8, 9
- Java, 10
- JAX-RPC, 15
- JWS, 13, 23

- Kompilieren, 13
- Kompilierung, 14

- Linux, 10

- Microsoft WSE, 9

- Novell, 10

- Open Source, 10

- Performanz, 12
- Provider, 6

- Registry, 6

- Requestor, 6
- Rollen, 5

- Schnittstelle, 5
- Semantic, 23
- Service-Aufruf, 15
- Service-Provider, 6
- Service-Registry, 6
- Service-Requestor, 6
- SOAP, 24
- SOAP-Request, 5
- SOAP-Response, 5
- Software-Lösungen, 8
- Sun, 9
- SunONE, 9

- Tomcat, 10

- UDDI, 24

- Veröffentlichung, 14
- Visual Basic, 9

- Web Service Architekturen, 4
- WebLogic Server, 11
- WebSphere, 8
- Windows, 9
- WSDL, 24
- WSE, 9

- XML, 24

- Zukunft, 19