Teleseminar „Web Services“

Universität Mannheim / Universität Karlsruhe

»Web Services in der Praxis«

Fabrizio Branca (mail@fabrizio-branca.de)
Georg Hackenberg (georg@eecoo.de)
20.07.2004

1 Einleitung

Inhalte der vorhergehenden Themen waren eher theoretische Modelle und Verfahren im Umgang mit Web Services. In unserem Thema „Web Services in der Praxis“ wollen wir zeigen, wie man Web Services letztendlich wirklich implementiert und nutzt, welche Programme und Werkzeuge man dazu verwenden kann und wie man die anstehenden Aufgaben vereinfachen kann.
Leider kann dieses Thema nicht alle Programmiersprachen abdecken. Daher wollen wir uns hier auf die Erstellungen von Clients und Server mit der Skriptsprache PHP und der Programmiersprache Java beschränken. Wir wollen die Entwicklungsumgebung „Eclipse“ vorstellen und demonstrieren, welche Möglichkeiten es gibt die Vorgänge zu automatisieren, um ein angenehmeres Arbeiten mit Web Services zu ermöglichen.

2 Entwicklungsumgebung Eclipse

Eclipse ist ein Framework, das meist als freie Entwicklungsumgebung (IDE) genutzt wird. Früher (Version <= 2.1) war Eclipse als erweiterbare IDE gedacht. Seit Version 3.0 ist Eclipse selbst nur der Kern, der die einzelnen Plugins lädt, die dann die eigentliche Funktionalität (z.B. die einer IDE) zur Verfügung stellt.
Sowohl Eclipse als auch die Plugins sind vollständig in Java implementiert und damit nicht nur sprach- sondern auch plattformunabhängig.
Die Bezeichnung Eclipse ist der englische Begriff für eine Sonnenfinsternis (solar eclipse) bzw. totale Sonnenfinsternis (total eclipse). Gerüchten nach, soll es sich dabei um eine Anspielung auf die Firma Sun Microsystems handeln, mit deren Geschäftspolitik die Entwickler von Eclipse nicht einverstanden waren.
Eclipse stellte den Nachfolger für IBM Visual Age dar. Die Entwicklung soll mehr als 40 Millionen Dollar gekostet haben. Der Quellcode für Eclipse wurde dann von IBM freigegeben. Die Verwaltung und Entwicklung von Eclipse wurde vom Eclipse-Projekt übernommen.
Folgende Funktionen ermöglichen und erleichtern das Programmieren und verwalten von Projekten in verschiedenen Programmiersprachen:

- Unterstützung von Plugins (z.B.: PHP-Eclipse [PHPEclipse]; SFTP [KlompSFTP]; SQL [EclipseSQL]; uvm.,...)
- syntax highlighting editor
- code completion
- source-level debugger
- class navigator
- file/project manager
- task-orientierte Entwicklung durch Perspektiven
- CVS-Unterstützung

3 Java

3.1 Vorstellung der Umgebung

3.1.1 Apache Tomcat

Um einen Web Service in Java zu realisieren, muss man sich erst eine geeignete Umgebung einrichten. Zur Ausführung eines Web Services benötigt man eine SOAP-Implementation. In diesem Fall verwenden wir Apache Axis. Axis ist ein normales Java-Servlet und um dies auszuführen braucht man noch einen Java Application Server. Wir verwenden hier – ebenfalls von Apache – den JSP / Servlet-Container Tomcat [Tomcat]. Die Installation von Tomcat ist recht einfach. Um den Server benutzen zu können muss er gestartet sein. Dazu gibt es mehrere Möglichkeiten.

- Bei der Installation von Apache werden Skripte mitinstalliert, die das Starten und das Beenden von Tomcat ermöglichen. Tomcat läuft dann in einem Konsolen-Fenster als Programm.
- Man kann Tomcat aber unter Windows NT / 2000 /XP auch als Service einrichten. Dann wird der Application Server mit dem Start von Windows automatisch gestartet und läuft unsichtbar im Hintergrund.

- Mit dem Sysdeo Plugin für Eclipse lässt sich Tomcat angenehm aus Eclipse heraus startet und beenden.

3.1.2 Die Architektur von Axis

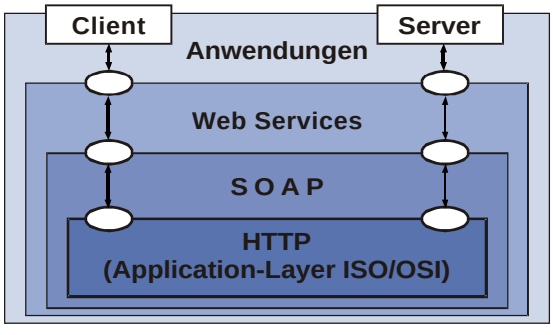
Apache Axis [ApacheAxis] (früher „Apache SOAP“) ist eine Implementation von SOAP. Das Paket enthält allerdings noch viele weitere Funktionen und Tools.
Die wichtigsten Tools zum Erstellen und Deployen eines Server / Clients in Axis:

- Java2WSDL: Generiert aus einem Java-Interface, die WSDL-Datei, die später den dazugehörigen Service beschreibt.
- WSDL2Java: Generiert nötige Klassen („Stubs“ / „Skeleton“), die es wesentlich vereinfachen einen Server oder eine Client-Applikation zu einer gegebenen WSDL-Datei zu schreiben.
- AdminClient: Tool zum deployen / undeployen des Web Services

Aus der Sicht von Tomcat ist Axis nur ein gewöhnliches Servlet. Bildlich könnte man sich vorstellen, dass Axis ein Bohrmaschinenaufsatz ist, dann wäre Tomcat die dazugehörige Bohrmaschine. Als „Servlet-Engine“ beschrieben, bietet er Axis den Nährboden zum funktionieren, ist sozusagen der Motor (=engine) von Axis und der auf Axis aufsetzenden Komponenten (Web Services).
Tomcat leitet alle /services/* und *.jws HTTP-Anfragen an den Axis-Server weiter. Diesem Servlet kommt dann die zentrale Rolle zu, folgende Aufgaben abzuarbeiten:

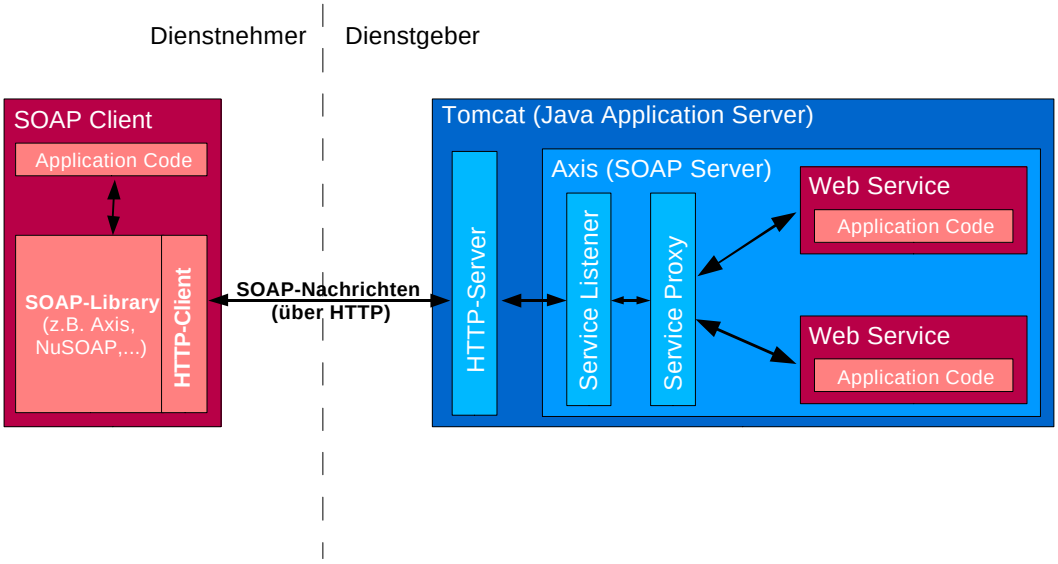
- Parsen der SOAP-Anfrage
- Lokalisierung der den Service implementierenden Klasse (dazu muss der Service „registriert“ also „deployed“ sein)
- Gegenfalls Kompilieren der Klasse (falls der JWS-Mechanismus eingesetzt wird)
- Aufruf von Handlern zur Vorverarbeitung der Anfragen
- Aufruf der eigentlichen Implementation
- Aufruf der Handler zur Nachbearbeitung der Anfrage
- Kodierung der Ergebnisse

3.1.3 Zusammenhang der Komponenten



Die Kommunikation von Anwendungen über Web Services baut auf dem ISO/OSI-Basisreferenz-Modell auf. Die äußerste Schicht dieses Modells, die „Application Layer“ ist für Web Services die Grundlage und in diesem Modell die innerste Schicht. SOAP, ein XML-Format, wird meistens über HTTP versenden. Web Services benutzen SOAP zur Kommunikation und die Anwendungen verwenden Web Services, um mit anderen Anwendungen über das Netz zu kommunizieren.

Im Detail sind die Kommunikation zwischen Dienstnehmer (Client) und Dienstgeber (Server) so aus:



Der Client hat in seiner Applikation eine Library eingebunden, die sich um die Kommunikation über SOAP mit dem Web Service kümmert. Diese Library hat meistens auch direkt einen http-Client integriert, der – eine Schicht tiefer – auch die Kommunikation über http steuert. Für Java kann man in Axis enthaltene Klassen dazu verwenden. (In PHP z.B. NuSOAP; Perl: SOAP::Lite,…)
Die beim Server ankommenden Nachrichten werden von Tomcat als Web Service anfragen identifiziert und an Axis weitergeleitet. Axis stellt den Service Listener bereit, um die Nachrichten entgegenzunehmen. Diese werden dann vom Service Proxy in Aktionen umgesetzt und an den entsprechenden Web Service, der die Aktion implementiert geschickt. Dazu muss der Web Service dem Service Proxy „bekannt“ sein. Zu diesem Zweck muss man nach der Erstellung des Web Services, diesen bei Axis über den Web Service Deployment Descriptor (WSDD) registrieren („deploy“).

3.2 Erstellen von Web Services

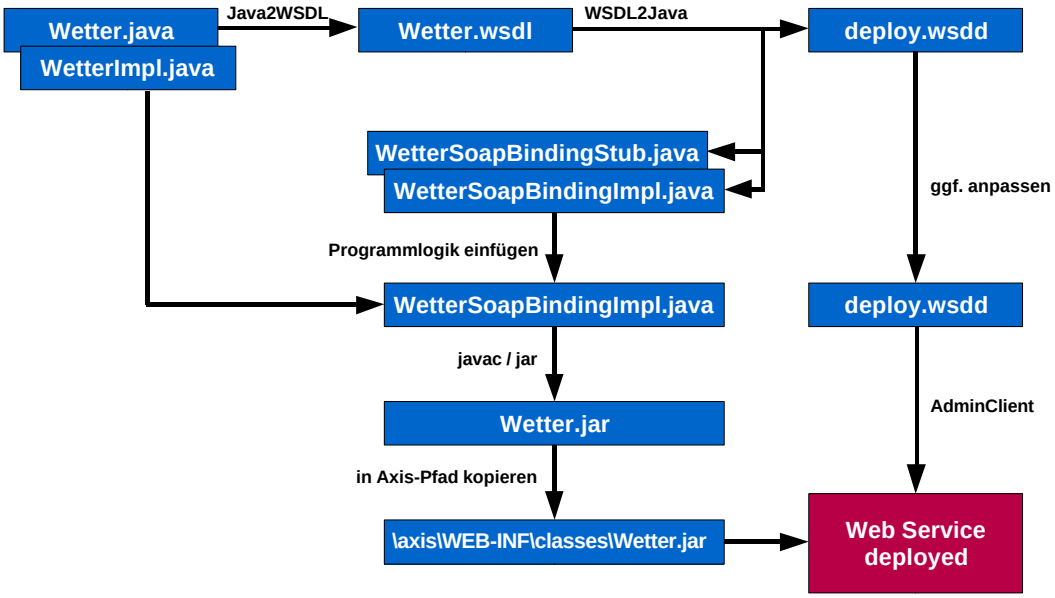
3.2.1 Server

3.2.1.1 Instant Deployment

„Instant Deployment“ (oder auch Drop-In –Deployment) ist eine Funktion von Axis, die es erlaubt Web Services Server schnell und einfach zu erstellen. Man schreibt eine Java-Klasse, mit den Methoden, die später als Web Service zur Verfügung stehen sollen. Kopiert diese Datei in den Axis-Pfad und benennt sie in eine „Java Web Service“-Datei (*.jws) um. Beim ersten Aufruf der Datei über HTTP, wird sie automatisch kompiliert und deployed. Außerdem wird eine WSDL-Datei generiert, die unter *.jws?WSDL zur Verfügung steht. Nachteil dieser Methode ist, dass man keine Möglichkeit hat, den Web Service zu konfigurieren und anzupassen.

3.2.1.2 Ausführliches Deployment

Um Mehr Einfluss auf die Konfiguration zu haben, werden Web Services in der Praxis meistens ausführlich deployed. Das ist zwar wesentlich aufwendiger, aber dafür bietet es mehr Konfigurationsmöglichkeiten (siehe SOAPMonitor).



Web Services werden in sechs Schritten erstellt:

- Erzeugung der Web-Service Schnittstelle:
  Zuerst wird eine Interface-Datei erstellt, die die Methoden mit Eingabe- und Ausgabeformaten der Parameter enthält (Wetter.java). Sinnvoll ist es, an dieser Stelle auch gleich die Funktionalität des Services zu implementieren (WetterImpl.java), so dass diese Klasse später einfacher eingebunden werden kann. Das vereinfacht das Verwalten und dient zur Übersichtlichkeit des gesamten Vorgangs.
- Mit dem Axis-Tool Java2WSDL wird aus der Interface-Klasse die WSDL-Datei generiert.
- Im nächsten Schritt lässt man von einem weiteren Axis-Tool – WSDL2Java – die nötigen Klassen zur Implementierung des Web Services erstellen. An dieser Stelle wird auch eine Deployment Descriptor erstellt (deploy.wsdd), der alle Informationen enthält, um den Service später bei Axis „anzumelden“.
  Die von WSDL2Java generierten Klassen und Dateien:
  - WetterSoapImpl.java:
    Diese Klasse wird später die eigentliche Implementierung enthalten.
  - WetterSoapSkeleton.java:
    Dieses Skeleton sitzt sozusagen zwischen Axis und der Implementation in WetterSoapImpl. Das Skelett delegiert lediglich die Anfragen.
  - deploy.wsdd
    Im Deployment-Deskriptor sind wichtige Informationen enthalten, die für die richtige Abarbeitung der Anfrage nötig sind. Dazu zählt der Name der Klasse, die den Service tatsächlich implementiert, welche ihrer Methoden aufrufbar sind, welche Art des Marshallings und des Unmarshallings verwendet wird, wie die Nachrichten kodiert werden und der targetNamespace des Services.
  - undeploy.wsdd:
    Der Undeployment-Deskriptor enthält nur den Namen des Services, damit dieser Aus der Axis-Engine wieder entfernt werden kann.
- An dieser Stelle muss nun in WetterSOAPBindingImpl.java die Programmlogik eingefügt werden. Man kann hier jetzt die Objekte der Klasse erstellen, die man am Anfang in WetterImpl.java vorbereitet hat und die entsprechenden Methoden in den Web Service Methoden aufrufen.
  Hier hat man außerdem die Gelegenheit, die WSDD-Datei anzupassen.

- Nun werden die Dateien kompiliert und ggf. in ein Package gepackt und in das richtige Verzeichnis innerhalb des Axis-Verzeichnisbaums kopiert.
- Der Web Service ist jetzt soweit vorbereitet, dass man ihn mit Hilfe des Axis-Tools AdminClient jetzt deployen kann. Der Web Service steht jetzt über den Application Server zu Verfügung.

3.2.1.3 Anpassungsmöglichkeiten der WSDD-Datei am Beispiel des SOAPMonitors

Durch das ausführliche Deployment hat man die Möglichkeit, den Informationsfluss unter einzelnen Web Services in einer Axis-Umgebung zu steuern. So kann man beispielsweise durch das Einfügen der Zeilen

```
<requestFlow>
      <handler type="soapmonitor"/>
</requestFlow>
<responseFlow>
      <handler type="soapmonitor"/>
</responseFlow>
```

in die WSDD eines Web Services den SOAPMonitor, der im Axis-Paket enthalten ist einbinden. Dies bewirkt, dass alle eingehenden SOAP-Requests und –Responses an den SOAPMonitor umgeleitet werden und von dort aus wieder zurück an den Web Service. Dadurch kann man die SOAP-Messages ansehen und protokollieren, was zum Verwalten und zur Fehlersuche sehr hilfreich sein kann.

3.2.1.4 Automatisieren der Vorgänge

Einige der Schritte zur Erstellung eines Servers kann man automatisieren, um den Vorgang zu beschleunigen. Im Folgenden werden einige Möglichkeiten vorgestellt.

3.2.1.4.1 Batch-Dateien

Die einfachste Möglichkeit ist, sich Batch-Dateien anzulegen, von denen aus alle Befehle mit den richtigen Parametern aufgerufen werden. Außerdem kann hier einfach die richtige Verzeichnisstruktur erstellt werden. Man muss den Batch-Dateien, dann über Parameter alle wichtigen Informationen übergeben.
Zusammenfassen könnte man zum Beispiel die Befehle Java2WSDL und WSDL2Java, um aus der Java-Interface-Klasse, die nötigen Java-Klassen zur Implementieren des Web Services zu erstellen. Nun kann man die Dateien anpassen und mit einer weiteren Batch-Datei die Kompilierung und da Deployment durchführen.
Nachteil dieser Methode ist, dass aufwendige erstellen der recht unflexiblen Batch-Dateien.

3.2.1.4.2 Apache Ant

Wesentlich eleganter geht die Automatisierung mit Apache Ant [Ant]. Das Vorgehen ähnelt der Erstellung und Verwendung von Make-Files. Ant-Dateien werden in XML abgelegt und enthalten alle Informationen zu Zwischenschritten. Über `ant wsdl2java` könnte man zum Beispiel die Java-Dateien erstellen lassen.
Ant lässt sich auch von Eclipse aus angenehm benutzen.

3.2.1.4.3 Eclipse-Plugins

Die schnellste Methode ist die Verwendung von Plugins mit Eclipse.

- Das Improve Axis Plugin for Eclipse [Improve] lässt eine einfache Erstellung der WSDL-Datei über das Kontext-Menü der Java-Datei zu (Java2WSDL).

Außerdem kann man mit diesem Plugin fertige Web Services über den WSDD im Kontextmenü deployen.

- Die WSDL2Java-Funktion ist im Improve-Plugin leider nicht implementiert. Daher muss man hier auf ein weiteres Plugin ausweichen: das WSDL2Java for Eclipse-Plugin von MySpotter [MySpotter]

3.2.2 Client

Ein Client lässt sich auf ähnliche Weise erstellen wie der Server. Mit dem Befehl WSDL2Java lassen sich automatisch die benötigten Klassen generieren:

- Wetter.java: (Interface)
  Dies ist das Service-Interface. Hierin finden sich die Signaturen alles im Service angeboten Methoden. Die Client Applikation verwendet dieses Interface für den tatsächlichen Aufruf des Web Services.
- WetterService.java (Interface)
  Auch das WetterService-Interface beeinhaltet die vom Service angebotenen Methoden. WetterService leitet sich von javax.xmnl.rpc.Service ab. Diese Klasse wird nur intern gebraucht und taucht im Client nicht auf.
- WetterServiceLocator.java (Factory-Klasse)
  Der ServiceLocator ist sozusagen die Factory-Klasse, die dem Client das Service-Objekt liefert.
  Es gibt zwei wichtige Methoden:
  public Wetter getWetter()
  public Wetter getWetter(URL portAddress)
  Beide liefern das gewünschte Interface zurück, sind also Factory-Methoden
- WetterSoapBindingStub.java (Klasse)
  Diese Klasse implementiert die klasse Stub-Funktionalität. Dort findet sich die Programmlogik wieder, die die SOAP-Schicht bedient. Diese Klasse wird nur intern gebraucht und taucht im Client nicht auf.

Mit Hilfe dieser Klasse lässt sich nun ein Client implementieren der so aussehen könnte:

```
public class WetterClient {
	public static void main(String [] args) throws Exception {
		// Den Service oeffnen
		WetterService service = new WetterServiceLocator();

		// Nun wird der Service genutzt um einen Stub zum Service
		// Definition Interface (SDI) zu bekommen
		Wetter vorhersage = service.getWetter();

		// Der eigentliche Aufruf des Webservices
		System.out.println("Temperatur in KA: " +
					vorhersage.temp(76131));
	}
}
```

Das Objekt Proxy erfüllt nun im Hintergrund unsichtbar die Funktionalität des Web Services. Die Methoden können vom Client aus aufgerufen werden, als wären es lokale Methoden des Objektes.

4 Fazit

Das Erstellen von Web Services ist in den meisten Fällen nicht so einfach wie das Schreiben eines simplen Programms, dass auf einem Einzelplatzrechner laufen soll. Besonders in Java muss meine eine aufwendige Umgebung dafür schaffen, und auch in PHP muss zur Nutzung und Bereitstellung von Web Services ein Webserver installiert und für PHP konfiguriert sein.
In beiden Fällen lassen sich Web Services allerdings angenehm und effizient programmieren, wenn diese jeweilige Umgebung eingerichtet ist und man sich eine Palette von Werkzeugen zurechtgelegt hat.
Je nach Anforderungen eigenen sich sowohl PHP als auch Java sehr gut für die Programmieren und Nutzung von Web Services und durch die steigende Beliebtheit dieser Technik, wird in diesem Gebiet sehr viel weiterentwickelt.

5 Links

- [Tomcat] **Tomcat Application Server**
  http://jakarta.apache.org/tomcat/
- [TomcatPlugin] **Sysdeo Eclipse Tomcat Launcher plugin**
  http://www.sysdeo.com/eclipse/tomcatPlugin.html
- [ApacheAxis] **Apache Axis**
  http://ws.apache.org/axis/
- [Ant] **Apache Ant**
  http://ant.apache.org/
- [Improve] **Improve Axis Plugin for Eclipse**
  http://www.improve-technologies.com/alpha/axis/
- [MySpotter] **WSDL2Java Eclipse Plug-in**
  http://www.myspotter.com/wsdl2java.shtml
- [Eclipse] **Eclipse**
  http://www.eclipse.org/
- [KlompSFTP] **Sftp File Synchronization Plugin**
  http://klomp.org/eclipse/org.klomp.eclipse.team.sftp/
- [PHPEclipse] **PHPEclipse**
  http://www.phpeclipse.de/
- [PHP] **PHP**
  http://www.php.net
- [NuSOAP] **NuSOAP**
  http://dietrich.ganx4.com/nusoap/

6 Quellen

6.1 Allgemein

- http://www.javaworld.com/columns/jw-web-services-index.shtml
  Stay current with JavaWorld's Web Services column
- http://www.javaworld.com/channel_content/jw-webserv-index.shtml
  Browse the Java and Web Services section of JavaWorld's Topical Index
- http://www.klick-and-bau.com/

Vorlesung "Informationsintegration und Web-Portale"

- http://www.xmethods.net
  Viele nützliche Web Services.
  Außerdem interessant: Try it (von Mindreef) zum Testen von WS
- http://www.mindreef.net/soapscope/wsdldemo
  Mindreef SOAPScope (WSDL/Test)
- http://www.webservices.org/
- http://wsindex.org/

6.2 Artikel

- http://www.oreillynet.com/pub/wlg/2360
  "Web Services We'd Like To See."
- http://news.com.com/2009-1017-966099.html
  "Amazon, Google lead new path to Web services"
- http://www.oreillynet.com/cs/user/view/wlg/2342
  "Why Amazon and Google Web Services Matter"
- http://www.phpbuilder.com/columns/badar20040430.php3?page=1
  "PHP, XML, XSL, XPATH and Web Services. This could be the start of something beautiful..."
  Interessant! Mit Demo in PHP (NuSOAP)
- http://webservices.xml.com/pub/a/ws/2004/03/24/phpws.html
  "Creating and Consuming Web Services With PHP"
  Interessant: Demo in PHP

6.3 PHP+Java

- http://www.scottnichol.com/soap.htm

6.4 PHP

- http://flash-db.com/services/tutorials/helloworld/
  Babelfish ÜBersetzung Web Service
- http://dietrich.ganx4.com/nusoap/index.php
  Die NuSOAP-Library
- http://talks.php.net/show/soap-phpcon-ny2003/0

6.5 Java

- http://www.eli.sdsu.edu/courses/spring03/cs683/notes/AXIS/AXIS.html
  Axis Client (und mehr)
- http://www.mitlinx.de/webservices/
  Allgemeine Infos zur Einrichtung von einer Web Service Umgebung (Tomcat, Ant, Axis,...)
- http://www.mitlinx.de/webservices/art.htm
  Infos zu Apache Ant
- http://www.javaworld.com/javaworld/jw-10-2000/jw-1020-ant.html
  Automate your build process using Java and Ant

- http://info.borland.com/techpubs/bes/v5/ws_tools.html
  Tools Overview
- http://www.onjava.com/pub/a/onjava/2002/06/05/axis.html?page=1
  "Creating Web Services with Apache Axis"
  Gutes Schrittweises Tutorial mit Beispielen
- http://www.webmethods.com/meta/default/folder/0000003978
  webMethods Glue
- http://www.theserverside.com/articles/article.tss?l=Systinet-web-services-part-1
  Creating a Web Service in 30 Minutes
- http://www.systinet.com/resources/tutorials
  Viele Tutorials zu Java (als PDF)
- http://java.sun.com/webservices/docs/1.0/tutorial/index.html
  Tutorial. Unter anderem auch zu Tomcat
- http://manojc.com/
  Schrittweises Tutorial zu Java WS
- http://www.ammai.com/modules.php?op=modload&name=Sections&file=index&req=viewarticle&artid=4&page=1
  Axis Tutorial
- http://www.oio.de/axis-soap.htm
  Die SOAP Engine Apache AXIS
- http://ws.apache.org/axis/
  Apache Axis
  siehe auch Dokumentation (http://ws.apache.org/axis/java/index.html)
- http://javaboutique.internet.com/tutorials/Axis2/
  Using Apache Axis version 1 to build Web Services
- http://www.onjava.com/pub/a/onjava/2002/06/05/axis.html
  Creating Web Services with Apache Axis
- http://tools.devchannel.org/devtoolschannel/04/01/27/219249.shtml?tid=46
  "Building Java Web Services with Apache Axis, part 1"
- http://www.ncsa.uiuc.edu/people/ramonw/ws_demo/axisdemo.html
  "Getting Started using Web Services with Tomcat and Axis"
- http://www.webservicescenter.com/Articles/frame_jws_windows.htm
  "Running Java Web Services"
  Tutorial on Getting Started with Java Web Services
- http://javaboutique.internet.com/tutorials/Tomcat/index.html
  "Using Apache Tomcat 4"
- http://www.galatea.com/flashguides/tomcat-axis-win32.xml
  Integrating Axis - Tomcat 4.x on Windows
- http://www.db.fmi.uni-passau.de/publications/books/DBMSeinf/services/UniVerwaltung.htm
  Erstellen eines einfachen Web Services und Clients (Tutorial)
- http://www.xmethods.com/gettingstarted/GLUE.html
  "A Quick-Start Guide to GLUE (beta 4)"
- http://www.jsptut.com/
  JSP Tutorial

6.6 Eclipse

- http://www.eclipse.org
  Java IDE
- http://www.sysdeo.com/eclipse/tomcatPlugin.html
  Sysdeo Tomcat Plugin

6.7 Anwendungen

- http://www.amazon.com/gp/browse.html/102-2822888-8224100?node=3435361
  Amazon Web Services
- http://www.sebastian-bergmann.de/en/soap_google.php
  Google-Demo von Sebastian Bergmann
- http://www.webservicex.com/sendsmsworld.asmx
  Kostenloser Service zum Verschicken von SMS.
- http://www.php9.com/index.php/section/articles/name/Amazon%20PHP%20API
  Amazon PHP API
- http://www.javaworld.com/javaworld/jw-04-2003/jw-0425-grid.html?
  A do-it-yourself framework for grid computing