UNIVERSITÄT SIEGEN

Hausarbeit zum Thema

WEB SERVICES

im Rahmen des Seminars
Softwarearchitekturen

Betreuer:
Dipl.-Inform. Benedikt Meurer

an der Universität Siegen

von
Peter Seemann

Inhaltsverzeichnis

1. Vorwort

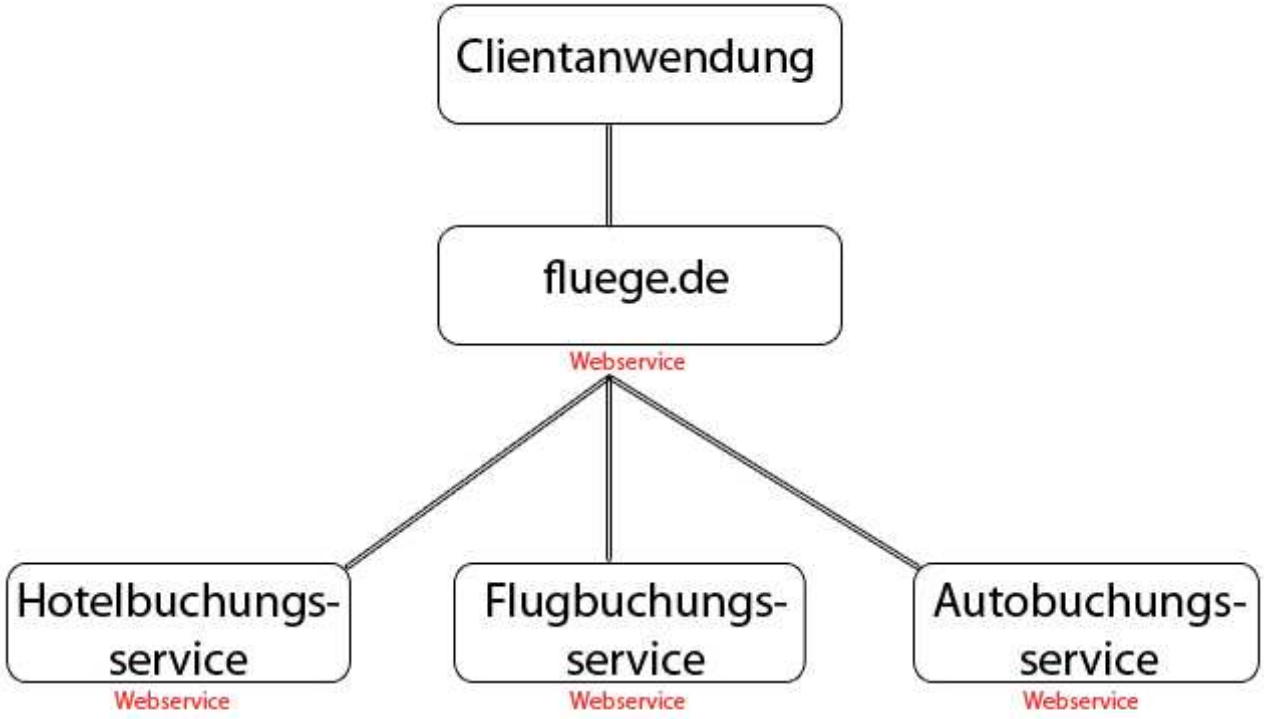
Da die Kommunikation im Netz sich nicht mehr nur auf Mensch-Maschine Kommunikation beschränkt, sondern vielfältigere Lösungen bietet, wie in etwa die Maschine-Maschine-Kommunikation und das Bereitstellen von Services, nehme ich dies als Grund, einen dieser Standards näher zu betrachten und darüber diese Hausarbeit zu schreiben. Bei diesem Standard handelt es sich um Web Services und der Architektur, die dahinter steckt. Daher gehe ich im Weiteren darauf ein, was Web Services eigentlich sind. Die dahinter steckende Architektur und sämtliche Standards, die für Web Services benötigt werden, werden anschließend erklärt. Zwei Standards mit derselben Aufgabe werden miteinander verglichen, bevor am Ende dieser Hausarbeit dann noch eine Zusammenfassung zu finden ist.

2. Was sind Web Services?

Ein Web Service ist eine Sammlung von Funktionen in einer einzelnen, in sich geschlossenen Einheit, welches im Internet veröffentlicht wird und so für andere Programme nutzbar ist. Die Web Service Technologie bietet ein Programmiermodell, um lose gekoppelte verteilte Anwendung zu erstellen, die offene Programmierstandards benutzen. Diese Architektur basiert auf Internet-Standards wie HTML, XML und SOAP und führt neue Konzepte und Technologien ein, so wie „Service Oriented Architectures“ (SOA), „Universal Description, Discovery and Integration“ (UDDI) und „Web Services Description Language“ (WSDL).

Die Idee hinter Webservices ist es, einen Übertragungsstandard zu haben, der Daten und Funktionalitäten zwischen Kommunikationspartnern austauscht, ohne dass ein Mensch in diese Kommunikation eingebunden ist. Es findet also eine reine Maschine-zu-Maschine-Kommunikation statt. Dadurch unterscheidet es sich vom bisherigen Web-Ansatz, bei dem ein Nutzer Daten von einem Webserver abruft.

Das Ziel von Webservices ist desweiteren die plattform- und sprachunabhängige Nutzung von Diensten auf entfernten Rechnern, die durch Kommunikation von XML-Dokumenten realisiert wird. Ein typisches Beispiel ist die Urlaubsbuchung im Internet. Man vergleicht die Preise auf fluege.de und bucht von dort aus ein Hotel in Ägypten, fliegt mit einer deutschen Fluggesellschaft und mietet einen Mietwagen einer englischen Leihfirma. Nun teilt der Webservice den gebuchten Dienstleistern automatisch mit, welche Leistung gebucht wurde, ohne dass der Anwender alle Leistungen separat auf deren Homepage buchen muss.



In Anlehnung an [1]

Weitere Paradebeispiele für Webservices sind Google und Amazon. Google erlaubt Webseiten, die Google-Suche auf der eigenen Seite als Webservice zu nutzen und Amazon bietet Shop-Suche und das deutsche Partnerprogramm als Webservice an. Diese Beispiele funktionieren einseitig, das heißt, dass lediglich diese Webseiten die jeweiligen

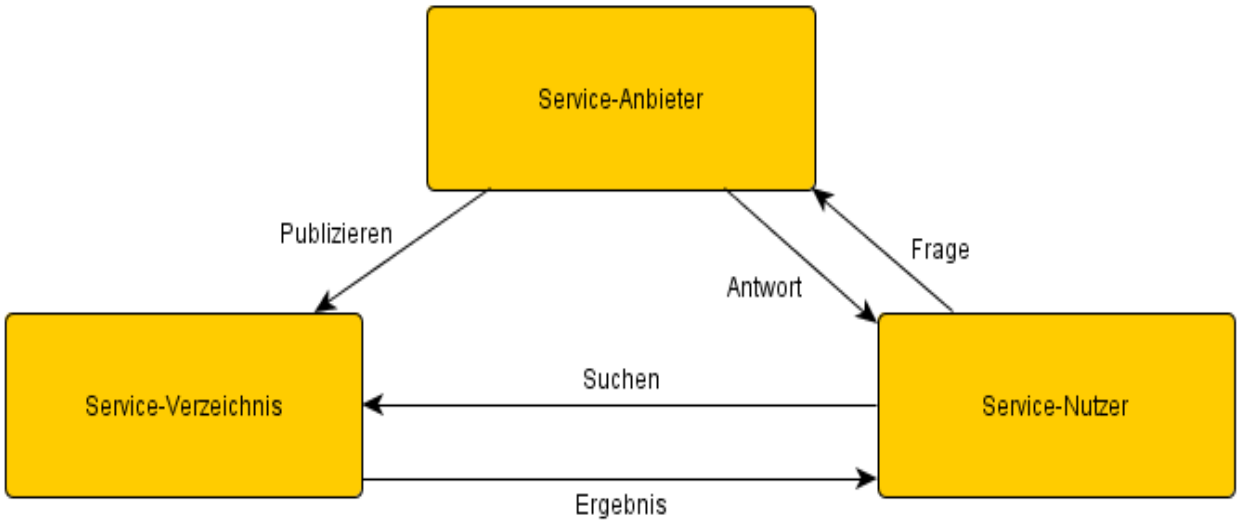
[1] Wismüller, S. 485

Webservices bei Amazon und Google aufrufen, nicht aber umgedreht. Prinzipiell ist aber eine beidseitige Kommunikation bei Webservices möglich. Dell zum Beispiel hat sein ganzes Warenwirtschaftssystem mit Webservices realisiert, um die Zulieferer an die eigenen Systeme anzubinden. [2]

3. Die Architektur von Web Services

Das architektonische Konzept hinter Web Services ist die service-orientierte-Architektur (SOA). Bei dieser Architektur stellt ein Service-Anbieter einen Service zur Verfügung. Dieser wird dann in einem Service-Verzeichnis publiziert, welches mit den gelben Seiten vergleichbar ist. In diesem Verzeichnis kann man nach Services suchen, mit der Adresse (URL) kann man dann den Service anfragen und erhält eine Antwort. Die weitere Kommunikation läuft auch über Frage-Antwort.

In folgender Grafik sei dies noch einmal verdeutlicht:



In Anlehnung an [3]

[2] Hauser, S. 12-17
[3] Hauser S.16

SOA definiert drei Rollen als die Grundlagen von Web Services:

1. Übermittlung: Die Kommunikation zwischen Anbieter, Verzeichnis und Nutzer setzt voraus, dass Nachrichten gesendet werden.
2. Beschreibung: Damit der Nutzer weiß, um was es sich bei dem Service handelt, müssen die Methoden, die von ihm bereitgestellt werden, beschrieben werden.
3. Verzeichnisdienst: Services werden in einem Verzeichnisdienst hinzugefügt, damit die Nutzer diese finden können.

Die Basistechnologien in dieser Architektur sind HTTP, SMTP, etc. für die Kommunikation, SOAP und XML-RPC für die Nachrichtenübermittlung, WSDL für die Service-Beschreibung und UDDI zum Auffinden des Services. Diese Technologien werden im weiteren Verlauf erklärt. [4]

4. XML-Technologien

Die Web Services-Architektur verwendet die Extensible Markup Language (XML), was soviel bedeutet wie „erweiterbare Auszeichnungssprache“, als standardisierten Weg, um Daten in strukturierter und maschinenlesbarer Weise zu repräsentieren. XML ist eine Ansammlung verwandter Technologien. Der wichtigste Teil für das Verständnis von Web Services ist die XML 1.0 Spezifikation, welches im Folgenden erklärt wird. [5]

Ein klarer Vorteil von XML ist, dass es plattformunabhängig und somit prädistiniert für den Austausch von Daten über das Internet ist. Durch das World Wide Web Consortium (W3C) wurde eine klare Spezifikation von XML-Dokumenten herausgebracht. Es ist ein Textdokument, dass aus Tags besteht, die Elemente umschließen. Diese Elemente können verschachtelt werden. Es ähnelt in der Syntax der HTML-Syntax, die Syntax ist

[4] Hauser, S. 15-17
[5] W3C-1

jedoch strikter und im Gegensatz zu HTML handelt es sich um Tags, die die Bedeutung der Informationen festhält, nicht die Formatierung.

Ein XML-Dokument besteht weiterhin aus einem Header und einem Body. Der Header besteht aus XML-Version, Zeichensatz und gegebenenfalls des Dokumententyps. Ein typischer Header könnte wie foglt aussehen:

<?xml version="1.0 "encoding="UTF-8" ?>

<!DOCTYPE addressbook SYSTEM ¨addressbook.dtd¨>

XML-Bodies widerum sind eine verschachtelte Folge von Elementen.

Im Folgenden ist ein Beispiel zu sehen von einer CD-Sammlung.

```
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
</CATALOG>
```

In Anlehnung an [6]

Die XML-Sprache definiert keine Elemente. Die Elemente und ihre Bedeutung werden von der Anwendung bestimmt. [7] [8]

5. Simple Object Access Protocol

Simple Object Access Protocol (SOAP) ist ein einfaches Protokoll für den Austausch von Informationen in einer verteilten Umgebung. SOAP kann Methodenaufrufe machen mit und ohne Rückgabewert.

Dies ist vergleichbar mit einem Frage/Antwort-Spiel:

Frage: Wie war das Fußballergebnis von heute?

Antwort: 2:0 für Deutschland

Wird über SOAP aber nur eine Nachricht gestellt, so benötigt man darauf keine Rückmeldung.

Nachricht: Deutschland hat 2:0 gewonnen.

SOAP ist unabhängig von Programmiermodellen, Plattform oder der Transportmethode, um SOAP-Nachrichten zu übertragen. Die Syntax basiert auf XML. Die aktuelle SOAP-Spezifikation des W3C beschreibt die Nutzung von SOAP via HTTP, aber auch andere Transportwege, wie SMTP können genutzt werden. [9]

Eine SOAP-Nachricht besteht im Grunde aus drei Teilen:

1. SOAP-Envelope: dies ist das Root-Element der SOAP-Nachricht und schließt alle anderen Elemente ein. Er ist vergleichbar mit einem Briefumschlag. Es enthält

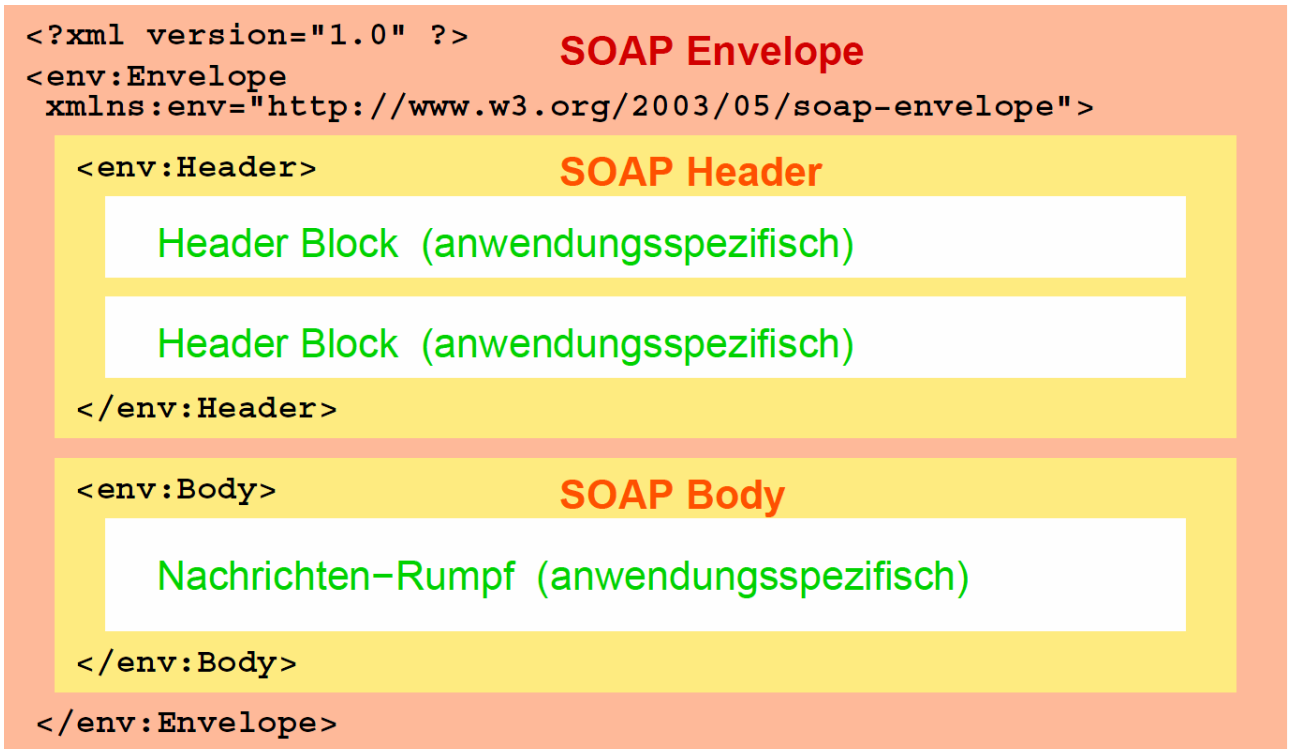
[6] W3C-2
[7] Burbiel, S.23
[8] W3C-3
[9] W3C-4

hauptsächlich Metainformationen. Anhand dessen kann der Nutzer an Informationen kommen, Objekte erstellen oder Datenstrukturen aufbauen. [10]

2. SOAP-Header: Der SOAP-HEADER ist optional und muss im Gegensatz zum Envelope und zum Body nicht in einer SOAP-Nachricht vorkommen. Der Header bestimmt zum Beispiel, wer die Informationen verarbeiten darf.
3. SOAP-Body: Im Bodyelement sind alle Objektdaten untergebracht, die übertragen werden können. Dies schließt unter Anderem Strukturen, Methoden und Objekte ein.

Streng genommen gibt es auch noch ein viertes Element einer SOAP-Nachricht, welches aber nicht unbedingt erforderlich ist. Das Fault-Element wird genutzt, um eventuelle Fehler zu beschreiben. [11]

Diese Elemente zusammen ergeben nun eine SOAP-Nachricht, die wie im Folgenden aussehen kann:



Grafik aus [12]

[10] Burbiel, S.240
[11] Hauser, S.39-55
[12] Wismüller, S.501

6. Web Services Description Language

SOAP definiert zwar ein Protokoll, dass für Nachrichten verwendet werden kann, aber es fehlt ein Mechanismus, der beschreibt, welche Art von Nachrichten übertragen werden sollen und wo diese Nachrichten hin sollen. Die Web Services Description Language (kurz: WSDL) löst dieses Problem durch die Bereitstellung einer Methode zur Beschreibung der Kommunikation in strukturierter Art und Weise. WSDL kann als interfacedefinierende Sprache für Web Services gesehen werden. WSDL basiert auch auf XML, in welchem es Beschreibungselemente gibt, die die verschiedenen Teile der Fernaufrufe darstellen:

1. Datentypen: ein Container für Datentypdefinitionen
2. Porttypen: definiert die Schnittstellen nach aussen, mit denen der Web Service mit dem Client kommuniziert
3. Bindungen: Dort wird festgelegt, welches Protokoll für die Nachrichtenübertragung verwendet werden soll, z.B. SMTP, http, MIME, SOAP
4. Nachrichten: definiert die Daten, die übertragen werden sollen
5. Ports: an dieser Stelle wird die Adresse des Endpunktes der Verbindung spezifiziert
6. Dienste: hier werden verwandte Ports zusammengefasst.

In der Regel ist es nicht nötig, ein WSDL-File per Hand zu erstellen, da die Web Services normalerweise WSDL-Dokumente zurückliefern. Auf Grund dessen und auf Grund der Länge wird hier auf ein Beispiel verzichtet. [13] [14]

[13] Hauser, S. 75-83
[14] Burbiel, S.24

7. Universal Description, Discovery and Integration

Universal Description, Discovery and Integration (kurz: UDDI) ist ein Verzeichnisdienst, der bestimmte Informationen über serviceorientierte Dienste bereitstellt. Es kann für dynamische Webservices benutzt werden. Es benutzt die bisher erklärten Techniken XML, SOAP und WSDL. UDDI transportiert Informationen dabei über HTTP unter der Verwendung von WSDL. Es besteht im Prinzip aus zwei Teilen. Zum einen aus einer Registry, die die Metadaten von Web Services enthält und zum Anderen aus Schnittstellendefinitionen. Praktisch alle modernen Programmiersprachen unterstützen UDDI.

Nun könnte man die Frage stellen, wozu man das Ganze braucht. Dazu ein einfaches Beispiel. Sie wollen sich für ihren Handel eine Anwendung schreiben, die sowohl Artikelpflege für sie übernehmen soll, als auch das Rechnungswesen. Wenn nun Ihre Zulieferer Web Services für solche Dinge anbieten, so können sie auf ihre Dienste zurückgreifen, insofern sich die Lieferanten an Spezifikationen halten und auch UDDI bereitstellen.

Zusammenfassend kann über UDDI sagen, dass es hauptsächlich folgenden vier Zwecken dient:

1. Dem Auffinden angebotener Dienste.
2. Dem Abruf von Schemata für die gegenseitige Kommunikation.
3. automatische Neukundenaquisition
4. B2B- Verbindungen

Das heißt aber nicht, dass UDDI nicht auch nur lokal in einem einzelnen Intranet möglich ist. Die berühmtesten Firmen, die an UDDI beteiligt waren, sind unter Anderem Firmen wie „Microsoft, IBM, SUN, HP, Dell, Oracle und SAP“. [15]

[15] Burbiel, S.302

Um sich auf den Servern, die Web Services anbieten, zu registrieren, gibt es eine Art Telefonverzeichnis, in Form von drei verschiedenen Informationsregistern:

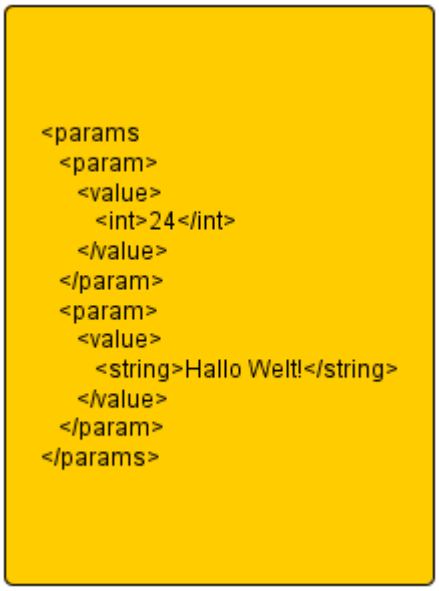
1. White Pages: hier stehen Name des Unternehmens, Adresse, Kontaktinformationen, Handelsregistereintrag, etc. des Unternehmens, das den Web Service anbietet
2. Yellow Pages: sie enthalten die Geschäftskategorie, in der der Web Service Anbieter eingeordnet werden kann
3. Green Pages: sie enthalten die technischen Informationen als WSDL. Aber auch Informationen bspw. über Geschäftsprozesse können hier zu finden sein.[16]

8. XML-RPC als Alternative zu SOAP?

XML-RPC steht für XML-Remote Procedure Call. Es ist ein Standard zur Übertragung von Nachrichten, ähnlich wie SOAP. Es basiert auch auf XML und ist deswegen ebenso plattformunabhängig. Für den Transport der Daten wird das HTTP-Protokoll genutzt. XML-RPC gab es bereits vor SOAP und ist so etwas, wie der Vorgänger, die neueste Spezifikation von XML-RPC ist allerdings von 1999. Bei XML-RPC werden beispielsweise Parameter durch mehrere Tags beschrieben, so dass die Größe und Komplexität der Nachricht erhöht wird.

Im Folgenden ist ein Beispiel für ein Integer und einen String-Parameter zu sehen, der dies noch einmal verdeutlicht.

[16] Burbiel, S.301-303



Auch auf Grund dieser Tatsachen wird XML-RPC kaum noch genutzt. So bietet Google seinen Web Service nur via SOAP an. XML-RPC ist also keine Alternative zu SOAP. [17]

9. Fazit

Der Hauptvorteil der Web Services Architektur ist, dass es Anwendungen, die mit verschiedenen Sprachen auf verschiedenen Plattformen entwickelt wurden, ermöglicht, in standardisierter Weise miteinander zu kommunizieren. Die serviceorientierte Architektur (SOA) mit ihrem Verzeichnisdienst (UDDI), macht das Auffinden und Nutzen der Web Services sehr einfach. Durch SOAP ist es möglich, Methodenaufrufe mit und ohne Rückgabewert zu tätigen, welche Dank der Web Services Description Language (WSDL) wissen, welches Ziel sie haben. Des Weiteren ist SOAP nicht nur der Standard, der neuer als der XML-RPC Standard ist, sondern ist diesem auch vorzuziehen.

[17] Hauser, S.30-39

Quellenangaben:

Bücher:

1. Hauser, Tobias/Löwer, Ulrich: Web Services. Die Standards., Bonn, 2002, Galileo Computing
2. Burbiel, Herbert: SOA & Webservices in der Praxis. Service Oriented Architecture mit XML, SOAP, .Net, Java & Co., München, 2007, Franzis Verlag

Skripte:

1. Wismüller, Roland: Skript zur Vorlesung: Client-/Server-Programmierung., Siegen, WS 2010, Universität Siegen

Onlinequellen:

1. W3C-1: http://www.w3schools.com/xml/xml_whatis.asp
2. W3C-2: http://www.w3schools.com/xml/cd_catalog.xml
3. W3C-3: http://www.w3schools.com/xml/xml_syntax.asp