

# Einführung in WebServices

Sven Tissot

**Pdv Technische Automation + Systeme GmbH  
Hamburg**

## Schlüsselworte

Web Services, XML, XSL, SOAP, WSDL, UDDI

## Zusammenfassung

Was sind Web Services? Neuer Wein in alten Schläuchen? Eine neue Sau die durch's IT-Dorf getrieben wird? Ein neuer visionärer Hyphe, der bald abgelöst wird? Oder die Möglichkeit kostengünstig heterogene Systeme auch über Unternehmensgrenzen hinweg miteinander zu verbinden?

Der folgende Artikel soll helfen, eine fundierte Antwort auf diese Fragen zu geben. Er beschreibt die grundlegenden Protokolle, Eigenschaften und Techniken die für den Einsatz von Web Services zur Anwendung kommen.

Nach einer kurzen Übersicht über XML werden die Basis-Komponenten

- SOAP
- WSDL
- UDDI

vorgelegt. Abschließend wird auf zukünftige Entwicklungen und die Unterstützung durch Oracle Werkzeuge eingegangen.

## Warum Web Services ?

Web Services stellen kein grundsätzlich neues Paradigma dar, sondern führen nur Entwicklungen zur Integration verschiedener Softwarekomponenten fort (Corba, DCOM, J2EE und RPC).

Grundsätzlich neu sind jedoch

- die Standardisierung
- die breite Unterstützung durch Hersteller
- die einfache Möglichkeit heterogene Systeme miteinander zu verbinden

Die *early adaptor* Phase ist schon fast vorbei, Google und Amazon implementieren bereits jetzt Schnittstellen via Web Services; IBM , Sun (One), Microsoft (.NET), Oracle und viele andere unterstützen mit ihren Produkten die Bereitstellung von Web Services.

Welche Vorteile bietet die Einführung von Web Services?

- Lose Koppelung unterschiedlicher Systeme
- Integration von Legacy Systemen
- Einfache Implementierung
- Standardisierung

## Kurzer Überblick über XML

XML ist eine Metasprache für das Definieren von Dokumenttypen und basiert auf der Standard Generalized Markup Language (SGML). Es ermöglicht z.B. den einfachen Austausch von Dokumenten über das Web, die validierbare Beschreibung von Konfigurationsdateien und dient als Basis für viele weitere Anwendungen: XSL(T), XML Query, XLink, XHTML, XPointer, VoiceXML und natürlich Web Services.

```
<Vortrag>
  <Autor>Sven Tissot</Autor>
  <Titel>Einführung in die Theorie und Praxis von WebServices</Titel>
  <Info Typ="Mini Lesson"/>
  <Datum>13.11.2002</Datum>
  <Keyword>SOAP</Keyword>
  <Keyword>UDDI</Keyword>
  <Keyword>WSDL</Keyword>
  <Keyword>XML</Keyword>
</Vortrag>
```

*Abb. 1 Beispiel einer XML Datei*

## DTD

Die Document Type Definition beschreibt für ein XML Vokabular welche Tags erlaubt sind, welche Verschachtelung gültig sind, Default-Werte und Wertbereiche sowie die Häufigkeit der Tags. Die DTD selbst liegt nicht als XML Dokument vor !

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Autor (#PCDATA)>
<!ELEMENT Datum (#PCDATA)>
<!ELEMENT Info EMPTY>
<!ATTLIST Info Typ CDATA #REQUIRED>
<!ELEMENT Keyword (#PCDATA)>
<!ELEMENT Titel (#PCDATA)>
<!ELEMENT Vortrag (Autor, Titel, Info?, Datum, Keyword+)>
```

*Abb. 2 DTD Definition*

## Schema

Als Vereinfachung und Verbesserung der DTD wurden XML-Schemata eingeführt, welche als Superset nun Typendefinitionen ermöglichen und ebenfalls als XML-Dokumente vorliegen.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="Vortrag">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Autor" type="xs:string"/>
        <xs:element name="Titel" type="xs:string"/>
        <xs:element name="Info">
          <xs:complexType><xs:attribute name="Typ" type="xs:string" use="required"/></xs:complexType>
        </xs:element>
        <xs:element name="Datum" type="xs:date"/>
        <xs:element name="Keyword" maxOccurs="unbounded">
          <xs:simpleType><xs:restriction base="xs:string">
            <xs:enumeration value="SOAP"/>
            <xs:enumeration value="UDDI"/>
            <xs:enumeration value="WSDL"/>
            <xs:enumeration value="XML"/>
          </xs:restriction></xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Abb. 3 XML Schema

Die Verarbeitung von XML Dokumenten erfolgt entweder durch Traversieren des vollständigen Baumes – d.h. Einlesen und Verarbeiten des gesamten Dokuments (**DOM**) oder Ereignis-getrieben: Das Dokument wird sequentiell geparsed und nur für die Verarbeitung relevante Knoten triggern entsprechende Aktionen (**SAX**).

## XSL

*Hier eventuell noch das XSL Beispiel (falls noch Platz ist ☺)*

## SOAP

Eine mögliche technische Definition für Web Services ist:

***Web Services sind Softwarekomponenten - beschrieben mittels WSDL - auf welche über ein Standard-Protokoll wie z.B. SOAP über HTTP zugegriffen werden kann.***

Einfacher gesagt: Web Services implementieren ein "Interface auf Applikations-Funktionalität welches mittels Standard-Protokollen bedienbar ist (Programming Web Services with SOAP)". Alternative Protokolle für die Übermittlung wären : SMTP, FTP, JABBER ... als Transportmedium sind im Extremfall sogar Disketten denkbar ☺.

Derzeit ist die Basis für WebServices SOAP und WSDL, UDDI stellt eine optionale Ergänzung dar.

SOAP ist der ultimative "Blind Date" - die beteiligten Partner wissen nichts voneinander bezüglich ihrer Implementierung oder der Plattform auf der sie laufen. Es findet ein lose gekoppelter Austausch von Informationen statt.

Was ist nun SOAP? SOAP - Simple Object Access Protocol – wurde initial spezifiziert von IBM, Microsoft, UserLand, DevelopMentor und Lotus. Es definiert eine XML basierte Envelop-Struktur zum Versenden und Empfangen von Nachrichten über ein bestimmtes Protokoll.



*Abb. 4 SOAP Message Struktur*  
(<http://xml.fujitsu.com/en/tech/soap/>)

Der optionale Header enthält Metadaten wie z.B. Informationen zur Authentifizierung, Transaktionsmanagement, Logging oder anderes. Der Body enthält die Nachricht oder die Daten. Grundsätzlich werden alle Datentypen der XML Schema Spezifikation oder daraus abgeleitete Typen unterstützt: Integer, Float, Boolean, String, Arrays, Vektoren, Strukturen.

```
<SOAP-ENC:int> 42<SOAP-ENC:int>  
<myvalue xsi:type="xsd:int">42</myvalue>
```

*Abb. 5 Beispiel alternative Datentyp Darstellung*

Als Weiterentwicklung von XML-RPC (SOAP RPC-Style: Parameter und Return Werte) bzw. EDI (SOAP Document-Style: Rechnung, Bestellung ...) wird SOAP heute auf über 20 unterschiedlichen Plattformen und mehr als 60 Sprachen unterstützt.

## Das Request/Response-Modell

Vom Aufbau her sind SOAP Messages "Ein-Weg" Nachrichten, die überwiegene Verwendung findet aber als Request/Response Paar statt. Hierfür sind Konventionen wichtig, um eine Zuordnung der Parameter zu den Antworten zu ermöglichen. Üblicherweise wird **Response** an die Requestmethode in der Response Nachricht angehängt:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <namespace1:GetVortraege xmlns:namespace1="urn:pdv.doag.webservices.GetVortraege">
      <c-gensym3 xsi:type="xsd:string">Tissot</c-gensym3>
    </namespace1:GetVortraege>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Abb. 5 SOAP Request

```
<xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:GetVortraegeResponse xmlns:ns1="urn:pdv.doag.webservices.GetVortraege"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <return xmlns:ns2="http://xml.apache.org/xml-soap" xsi:type="ns2:Vector">
        <item xsi:type="xsd:string">Einfuehrung in Web Services</item>
        <item xsi:type="xsd:string">Entwicklung dynamischer Web Seiten mit PL/SQL</item>
      </return>
    </ns1:GetVortraegeResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Abb. 6 Beispiel SOAP Response

## WSDL

WSDL (Web Services Description Language) ist eine XML Spezifikation zur Beschreibung der Schnittstellen von Web Services. Im Wesentlichen werden die Endpunkte der Kommunikationspartner beschrieben, welche Dokumente oder Anchriften austauschen. Diese Daten werden abstrakt beschrieben und dann an konkrete Protokolle und Formate gebunden. WSDL kann für beliebige Nachrichtenformate oder Netzwerk-Protokolle erweitert werden, die wesentlichen Implementierungen umfassen jedoch SOAP über HTTP.

```

<?xml version = '1.0'?>
<!--Generated by the Oracle9i JDeveloper Web Services WSDL Generator-->
<definitions
  name="GetVortraege"
  targetNamespace="http://localhost:8080/pdv/doag/webservices/GetVortraege"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://localhost:8080/pdv/doag/webservices/GetVortraege"
  xmlns:ns1="http://localhost:8080/pdv/doag/webservices/GetVortraege/schema">
  <types>
    <schema targetNamespace="http://localhost:8080/pdv/doag/webservices/GetVortraege/schema"
      xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
      <complexType name="vector">
        <sequence>
          <element name="item" minOccurs="0" maxOccurs="unbounded" type="anyType"/>
        </sequence>
      </complexType>
    </schema>
  </types>
  <message name="GetVortraege0Request">
    <part name="Autor" type="xsd:string"/>
  </message>
  <message name="GetVortraege0Response">
    <part name="return" type="ns1:vector"/>
  </message>
  <portType name="GetVortraegePortType">
    <operation name="GetVortraege">
      <input name="GetVortraege0Request" message="tns:GetVortraege0Request"/>
      <output name="GetVortraege0Response" message="tns:GetVortraege0Response"/>
    </operation>
  </portType>
  <binding name="GetVortraegeBinding" type="tns:GetVortraegePortType">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetVortraege">
      <soap:operation soapAction="" style="rpc"/>
      <input name="GetVortraege0Request">
        <soap:body use="encoded" namespace="urn:pdv.doag.webservices.GetVortraege"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
      </input>
      <output name="GetVortraege0Response">
        <soap:body use="encoded" namespace="urn:pdv.doag.webservices.GetVortraege"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
      </output>
    </operation>
  </binding>
  <service name="GetVortraege">
    <port name="GetVortraegePort" binding="tns:GetVortraegeBinding">
      <soap:address location="http://localhost:8080/soap/servlet/rpcrouter"/>
    </port>
  </service>
</definitions>

```

*Abb. 7 Beispiel WSDL Beschreibung*

Die meisten Werkzeuge und IDE sind in der Lage die WSDL zu einem bestimmten Service automatisch zu generieren.

Wie findet man nun eine WSDL Definitionen? Einerseits kann die Adresse (z.B Datei oder URL) direkt angegeben werden, falls sie bekannt ist. Zentrale Repositories wie z.B. xmethods.net oder www.salcentral.com ermöglichen das Auffinden über die Beschreibung des Services.

Ein standardisiertes Verfahren zum Suchen und Finden von Web Services stellt UDDI dar.

## **UDDI**

Web Services stellen kein grundsätzlich neues Paradigma dar, sondern führen nur

## **Zukunft**

Web Services stellen kein grundsätzlich neues Paradigma dar, sondern führen nur

Entwicklungen z

Die Seiten werden in Schwarz/Weiß gedruckt , achten Sie darauf dass sie nicht zu klein abgebildet werden

*Abb. 1 Times New Roman kursiv*

Die Seitenzahl wird von uns eingefügt!

### **Kontaktadresse:**

**Sven Tissot**

Dorotheenstraße 64  
D-22301 Hamburg

Telefon: +49(0)40-69213 266  
Fax: +49(0)40-69213 278  
E-Mail: [tissot@pdv-tas.de](mailto:tissot@pdv-tas.de)  
Internet: <http://www.pdv-tas.de>