

## XML Vorlesung ETHZ, Sommersemester 2006

# Web Services

Erik Wilde

27.6.2006

<http://dret.net/lectures/xml-ss06/>

## Übersicht

- XML als Präsentationsschicht
  - Vergleich mit dem OSI Referenzmodell
  - *Abstract Syntax Notation One (ASN.1)*
- XML für RPC-Mechanismen
  - XML-RPC
  - *Simple Object Access Protocol (SOAP)*
- Web Services
  - SOAP, WSDL, UDDI, ...

## OSI vs. Internet

- Open Systems Interconnection (OSI)
  - ISO-Standardisierung um 1990 herum
  - führte ein Schichtenmodell ein
    - sieben Schichten für die modulare Modellierung
    - Modellierung von Diensten auf jeder Schicht
    - jede Schicht nutzt den Dienst der darunterliegenden Schicht
  - gedacht als komplette Ablösung des Internet
- Phase der Unsicherheit Anfang der 90er
  - Entscheidung OSI/Internet war unklar
  - letztlich Erfolg des Internet
    - einfache, bekannte und erprobte Technologien
    - stark zunehmender Erfolg durch Email und WWW

27.6.2006

XML Vorlesung ETHZ SS 2006

3

## OSI Presentation Layer (OSI Layer 6)

- die sechste Schicht des OSI-Modells
  - Abstraktion von verschiedenen Datenmodellen
  - Lehre aus der "Network Byte Order" des Internet
    - Big Endian oder Little Endian?
    - das Internet benutzt Big Endian
  - Applikationen tauschen Daten abstrakt aus
  - der OSI-Stack sorgt für Kompatibilität
- OSI-Dienstschnittstellen benutzen ASN.1
  - Datenaustausch über ASN.1
  - Konvertierung in die Transfersyntax im OSI Stack
  - Aushandlung der Transfersyntax

27.6.2006

XML Vorlesung ETHZ SS 2006

4

## Beispiel für ASN.1

```
Personnel Record ::= [APPLICATION 0] SET {
    name Name,
    title VisibleString,
    number EmployeeNumber,
    dateOfHire Date,
    nameOfSpouse Name,
    children SEQUENCE OF ChildInformation DEFAULT {} }
ChildInformation ::= SET {
    name Name,
    dateOfBirth Date }
Name ::= [APPLICATION 1] SEQUENCE {
    givenName VisibleString,
    initial VisibleString,
    familyName VisibleString }
EmployeeNumber ::= [APPLICATION 2] INTEGER
Date ::= [APPLICATION 3] VisibleString -- YYYY MMDD
```

27.6.2006

XML Vorlesung ETHZ SS 2006

5

## ASN.1 Encoding Rules

- ASN.1 definiert eine abstrakte Syntax
  - aber wie werden konkrete Daten codiert?
  - dies ist die Aufgabe von Encoding Rules
    - Abbildung ASN.1 abstrakte Syntax ↔ Octet Stream
- *Basic Encoding Rules (BER)*
  - einfachste Codierung: Tag/Length/Value
- *Packed Encoding Rules (PER)*
  - effiziente Codierung am Datentyp orientiert
  - u.U. Ausnutzung von jedem Bit (aligned vs. unaligned)
- *Distinguished Encoding Rules (DER)*
  - Untermenge von BER für eindeutige Codierung

27.6.2006

XML Vorlesung ETHZ SS 2006

6

## Beispiel für XML Instanz

```
<?xml version="1.0" encoding="UTF-8"?>
<Personnel Record>
  <name>
    <givenName>John</givenName>
    <initials>P</initials>
    <familyName>Smith</familyName>
  </name>
  <title>Director</title>
  <number>51</number>
  <dateOfHire>19710917</dateOfHire>
  <nameOfSpouse>
    <givenName>Mary</givenName>
    <initials>T</initials>
    <familyName>Smith</familyName>
  </nameOfSpouse>
  <children>
    <childInformation>
      <name>
        <givenName>Ralph</givenName>
        <initials>T</initials>
        <familyName>Smith</familyName>
      </name>
      <dateOfBirth>19571111</dateOfBirth>
    </childInformation>
    <childInformation>
      <name>
        <givenName>Susan</givenName>
        <initials>B</initials>
        <familyName>Jones</familyName>
      </name>
      <dateOfBirth>19590717</dateOfBirth>
    </childInformation>
  </children>
</Personnel Record>
```

27.6.2006

XML Vorlesung ETHZ SS 2006

7

## ASN.1 XML Encoding Rules (XER)

- klassische ASN.1 Codierungen sind binär
  - Kommunikation zwischen OSI Layers
  - binäre Codierungen sind effizienter
- ASN.1 definiert eine Schemasprache
  - XML definiert mehrere Sprachen
    - DTDs für Schemas (langsam abgelöst durch XML Schema)
    - XML als Sprache für Instanzen (d.h. Dokumente)
  - Abbildung von ASN.1 auf XML ist möglich
- XER als Schnittstelle zwischen ASN.1 und XML
  - Codierung von ASN.1 Instanzen als XML Dokument
  - unglücklicherweise nicht kompatibel mit XML Schema

27.6.2006

XML Vorlesung ETHZ SS 2006

8

## XML ASN.1 Encoding Rules

- XML definiert Schema und Instanz
  - DTDs für das Schema (oder auch XML Schema)
  - XML Dokumente für die Instanzen
- XML Dokumente sind extrem voluminös
  - Relikt aus der Dokumentenverarbeitung mit SGML
  - Tags verursachen sehr hohe Redundanz
- alternative Codierungen wären möglich
  - im einfachsten Fall Zip von einem XML Dokument
    - Nachteil: die Struktur ist nicht mehr sichtbar
  - XML-spezifische Kompression
  - Abbildung auf ASN.1 und dann PER anwenden

27.6.2006

XML Vorlesung ETHZ SS 2006

9

## Vergleich ASN.1 und XML Encoding

Unaligned PER	84 Bytes
Aligned PER	94 Bytes
BER with definitive length	136 Bytes
BER with indefinite length	161 Bytes
XER	1420 Bytes

27.6.2006

XML Vorlesung ETHZ SS 2006

10

## XML vs. ASN.1 (Extremfall)

```
Simple-example ::= SEQUENCE {
  firstfield INTEGER (0..7),
  secondfield BOOLEAN,
  thirdfield INTEGER (8..11),
  fourthfield SEQUENCE {
    fourA BOOLEAN,
    fourB BOOLEAN } }
```

PER	1 Byte
BER	19 Bytes
XER	183 Bytes

```
<Simple-example>
  <firstfield>5</firstfield>
  <secondfield>false</secondfield>
  <thirdfield>10</thirdfield>
  <fourthfield>
    <fourA>false</fourA>
    <fourB>false</fourB>
  </fourthfield>
</Simple-example>
```

27.6.2006

XML Vorlesung ETHZ SS 2006

11

## Verhältnis von XML und ASN.1

- konkurrierende Schema-Sprachen
  - XML mit DTDs oder XML Schema
  - ASN.1 mit der ASN.1 Specification Language
- Vorteile von XML
  - wesentlich populärer
  - viel verbreitete und freie Software
    - beginnt sich als Standard-Software zu etablieren
- Vorteile von ASN.1
  - Kompatibilität mit XML über XER
  - kompaktere Codierungen mit BER oder PER

27.6.2006

XML Vorlesung ETHZ SS 2006

12

## Microsoft's .NET und XML

- Datenmodell von Programmiersprachen
  - oftmals Datenaustausch über den Stack
  - Datenformat durch Compiler implementiert
- Datenmodell fest in der Sprache verankert
  - einfache Datentypen (Integer, Boolean, ...)
  - komplexe Datentypen (sprachspezifisch)
- .NET definiert ein gemeinsamen Datenmodell
  - mehrere Sprachen, ein Datenmodell (XML)
  - dynamisches Zusammenspiel vieler Sprachen
    - aus einer Java-Routine eine C Prozedur aufrufen
    - diese später mit einer neueren C# Implementierung ersetzen

27.6.2006

XML Vorlesung ETHZ SS 2006

13

## XML-RPC

- extrem einfacher RPC Mechanismus
  - Codierung von Datentypen in XML
    - vordefinierte Typenmenge
  - Austausch über XML und HTTP
- keine weitergehende Infrastruktur
  - Aufrufsemantik unklar
  - Frage der Serviceauffindung unklar
- eher Fingerübung als komplette Infrastruktur
  - sehr einfach zu verwenden
  - sehr klare Limitierungen für ernsthafte Anwendungen

27.6.2006

XML Vorlesung ETHZ SS 2006

14

## XML-RPC Request

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-Length: 177
```

```
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getState</methodName>
  <params>
    <param>
      <value><i 4>41</i 4</value>
    </param>
  </params>
</methodCall>
```

## XML für Web Services

- was sind Web Services?
  - Dienste basierend auf Web-Technologien
  - nicht unbedingt inhaltlich auf das Web bezogen
    - viele Anwendungen in geschlossenen Benutzergruppen
  - XML als Datenformat
- Web Services für B2B Anwendungen
  - allgemein verfügbare Software
  - allgemein angewendete Technologie
  - allgemein vorhandenes Know-How
  - allgemein akzeptiertes Vorgehen



## SOAP 1.1

- *Simple Object Access Protocol (SOAP)*
- Message Modell
  - Header
  - Body
  - Attachments
- Unabhängig vom Transport-Protokoll
  - Bindings für HTTP
  - SMTP, RMI, JMS etc. denkbar aber nicht in SOAP definiert
- RPC oder one way (Messaging)

27.6.2006

XML Vorlesung ETHZ SS 2006

17

## SOAP 1.2

- aktuelle Version von SOAP
- nicht überall implementiert
- anderes Datenmodell als SOAP 1.1
  - SOAP 1.1 basiert auf XML 1.0
  - SOAP 1.2 basiert auf dem Infoset
- stärkere Betonung von Messaging
  - *Representational State Transfer (REST)* vs. RPC
- bessere Integration von binären Daten
  - Transport als Attachments von SOAP Messages
  - transparente Integration in das Infoset

27.6.2006

XML Vorlesung ETHZ SS 2006

18

## SOAP Request

```
POST /EchoService/EchoService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: 369
SOAPAction: "http://bsquare.ws/echoservice/Echo"
```

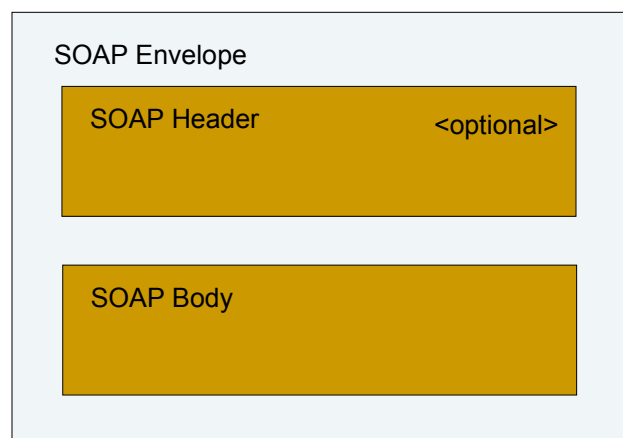
```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Echo xmlns="http://bsquare.ws/echoservice/">
      <echoString>hello</echoString>
    </Echo>
  </soap:Body>
</soap:Envelope>
```

27.6.2006

XML Vorlesung ETHZ SS 2006

19

## SOAP Aufbau



27.6.2006

XML Vorlesung ETHZ SS 2006

20

## SOAP Header und Body

- SOAP Envelope umfasst alles
- SOAP Header ist Information auf dem Umschlag
  - optional (Umschlag kann leer sein)
  - Routing-Information (Absender, Pfad, ...)
- SOAP Body sind die Nutzdaten
  - muss existieren
  - eigentlicher Inhalt der SOAP Message
  - kann beliebig komplex strukturiert sein
    - ist immer eine XML-Struktur
    - kann auf binäre Daten verweisen (SOAP Attachments)

27.6.2006

XML Vorlesung ETHZ SS 2006

21

## SOAP Response

```

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Fri, 01 Feb 2002 16:21:09 GMT
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Length: 358

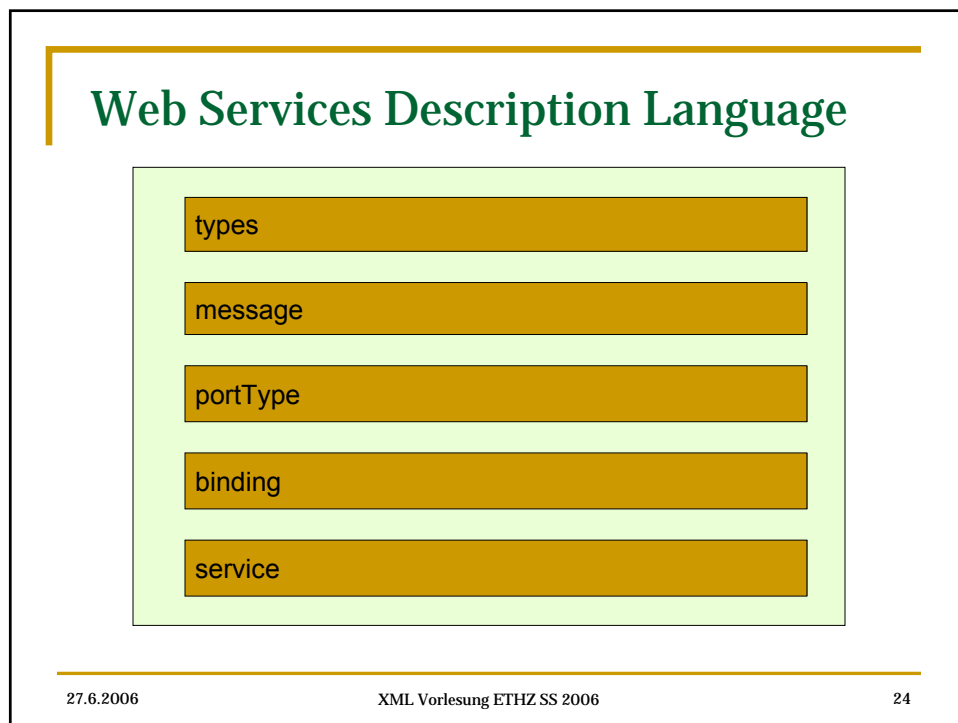
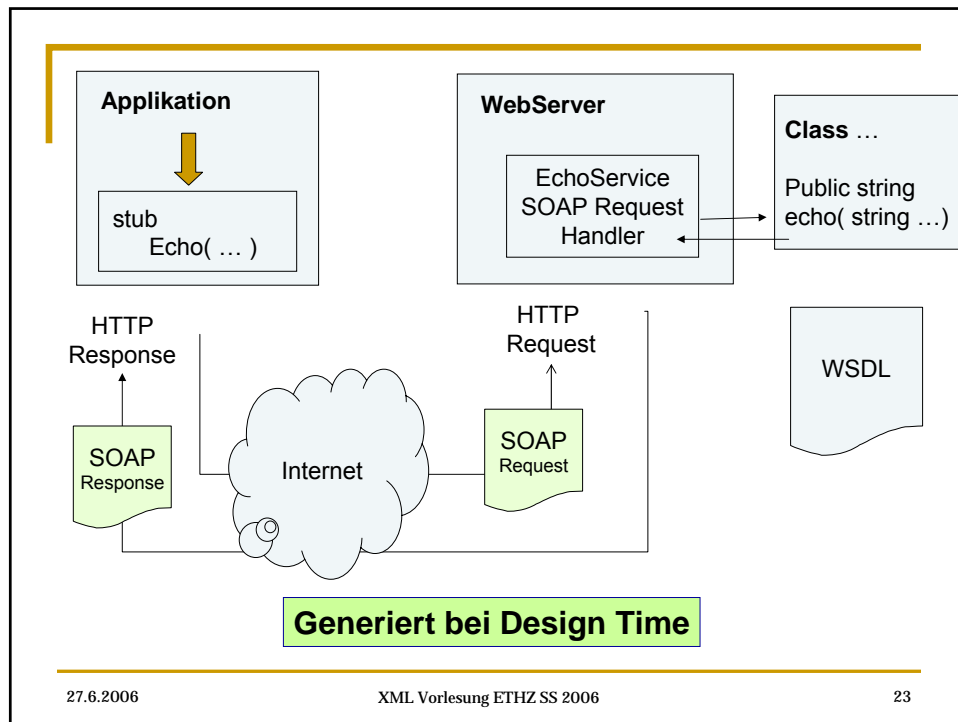
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <EchoResponse xmlns="http://bsquare.ws/echoservice/">
      <EchoResult>Echo [17:21] hello</EchoResult>
    </EchoResponse>
  </soap:Body>
</soap:Envelope>

```

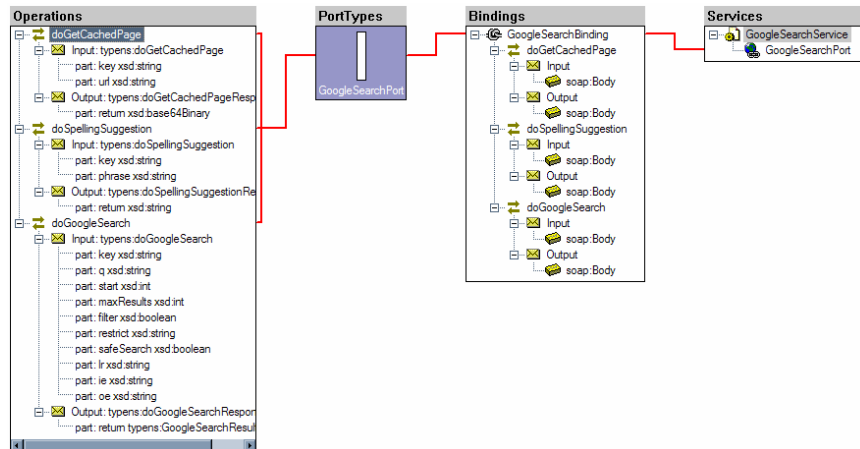
27.6.2006

XML Vorlesung ETHZ SS 2006

22



## WSDL Visualisierung (Google WSDL)



27.6.2006

XML Vorlesung ETHZ SS 2006

25

## Beispiel WSDL Types (Google)

```

<wsdl:types>
  <xsd:schema targetNamespace="urn:GoogleSearch">
    <xsd:complexType name="GoogleSearchResult">
      <xsd:all>
        <xsd:element name="documentFiltering" type="xsd:boolean"/>
        <xsd:element name="searchComments" type="xsd:string"/>
        <xsd:element name="estimatedTotalResultsCount" type="xsd:int"/>
        <xsd:element name="estimateExact" type="xsd:boolean"/>
        <xsd:element name="resultElements" type="typens:ResultElementArray"/>
        <xsd:element name="searchQuery" type="xsd:string"/>
        <xsd:element name="startIndex" type="xsd:int"/>
        <xsd:element name="endIndex" type="xsd:int"/>
        <xsd:element name="searchTips" type="xsd:string"/>
        <xsd:element name="directoryCategories"
          type="typens:DirectoryCategoryArray"/>
        <xsd:element name="searchTime" type="xsd:double"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:schema>
  ...

```

27.6.2006

XML Vorlesung ETHZ SS 2006

26

## Beispiel WSDL Message (Google)

```

<message name="doGoogleSearch">
  <part name="key" type="xsd:string"/>
  <part name="q" type="xsd:string"/>
  <part name="start" type="xsd:int"/>
  <part name="maxResults" type="xsd:int"/>
  <part name="filter" type="xsd:boolean"/>
  <part name="restrict" type="xsd:string"/>
  <part name="safeSearch" type="xsd:boolean"/>
  <part name="lr" type="xsd:string"/>
  <part name="ie" type="xsd:string"/>
  <part name="oe" type="xsd:string"/>
</message>
<message name="doGoogleSearchResponse">
  <part name="return" type="typens:GoogleSearchResult"/>
</message>
...

```

27.6.2006

XML Vorlesung ETHZ SS 2006

27

## Beispiel WSDL PortType (Google)

```

<portType name="GoogleSearchPort">
  <operation name="doGetCachedPage">
    <input message="typens:doGetCachedPage"/>
    <output message="typens:doGetCachedPageResponse"/>
  </operation>
  <operation name="doSpellingSuggestion">
    <input message="typens:doSpellingSuggestion"/>
    <output message="typens:doSpellingSuggestionResponse"/>
  </operation>
  <operation name="doGoogleSearch">
    <input message="typens:doGoogleSearch"/>
    <output message="typens:doGoogleSearchResponse"/>
  </operation>
</portType>

```

27.6.2006

XML Vorlesung ETHZ SS 2006

28

## Beispiel WSDL Binding (Google)

```
<binding name="GoogleSearchBinding"
  type="typens:GoogleSearchPort">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="doGetCachedPage">
    <soap:operation soapAction="urn:GoogleSearchAction"/>
    <input>
      <soap:body use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:GoogleSearch"/>
    </input>
    <output>
      <soap:body use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:GoogleSearch"/>
    </output>
  </operation>
  ...
```

27.6.2006

XML Vorlesung ETHZ SS 2006

29

## Beispiel WSDL Service (Google)

```
<service name="GoogleSearchService">
  <port name="GoogleSearchPort"
    binding="typens:GoogleSearchBinding">
    <soap:address location="http://api.google.com/search/beta2"/>
  </port>
</service>
```

27.6.2006

XML Vorlesung ETHZ SS 2006

30

## Google SOAP Request

```
<SOAP-ENV:Envelope
  xml ns: SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xml ns: SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xml ns: xsi="http://www.w3.org/2001/XMLSchema-instance"
  xml ns: xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <m:doGoogleSearch xml ns:m="urn:GoogleSearch"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <key xsi:type="xsd:string">String</key>
      <q xsi:type="xsd:string">String</q>
      <start xsi:type="xsd:int">0</start>
      <maxResults xsi:type="xsd:int">0</maxResults>
      <filter xsi:type="xsd:boolean">1</filter>
      <restrict xsi:type="xsd:string">String</restrict>
      <safeSearch xsi:type="xsd:boolean">1</safeSearch>
      <lr xsi:type="xsd:string">String</lr>
      <ie xsi:type="xsd:string">String</ie>
      <oe xsi:type="xsd:string">String</oe>
    </m:doGoogleSearch>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Semantik des Google SOAP Request

**2. Search Request Format**

**2.1 Search Parameters**

This table lists all the valid name-value pairs that can be used in a search request and describes how these parameters will modify the search results.

Name	Description
key	Provided by Google, this is required for you to access the Google service. Google uses the key for authentication and logging.
q	(See <a href="#">Query Terms</a> section for details on query syntax.)
start	Zero-based index of the first desired result.
maxResults	Number of results desired per query. The maximum value per query is 10. <i>Note:</i> If you do a query that doesn't have many matches, the actual number of results you get may be smaller than what you request.
filter	Activates or deactivates automatic results filtering, which hides very similar results and results that all come from the same Web host. Filtering tends to improve the end user experience on Google, but for your applications you may prefer to turn it off. (See <a href="#">Automatic Filtering</a> section for more details.)
restrict	Restricts the search to a subset of the Google Web index, such as a country like "Ukraine" or a topic like "Linux." (See <a href="#">Restricts</a> for more details.)
safeSearch	A Boolean value which enables filtering of adult content in the search results. See <a href="#">SafeSearch</a> for more details.
lr	<i>Language Restrict</i> - Restricts the search to documents within one or more languages.
ie	<i>Input Encoding</i> - this parameter has been deprecated and is ignored. All requests to the APIs should be made with UTF-8 encoding. (See <a href="#">Input and Output Encodings</a> section for details.)
oe	<i>Output Encoding</i> - this parameter has been deprecated and is ignored. All requests to the APIs should be made with UTF-8 encoding. (See <a href="#">Input and Output Encodings</a> for details.)

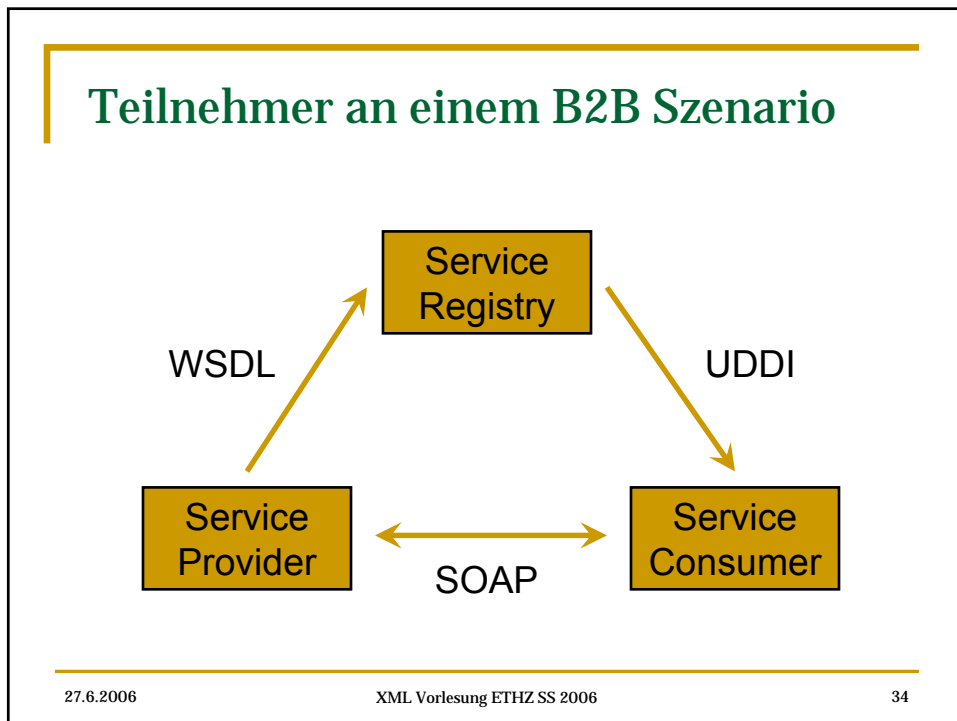
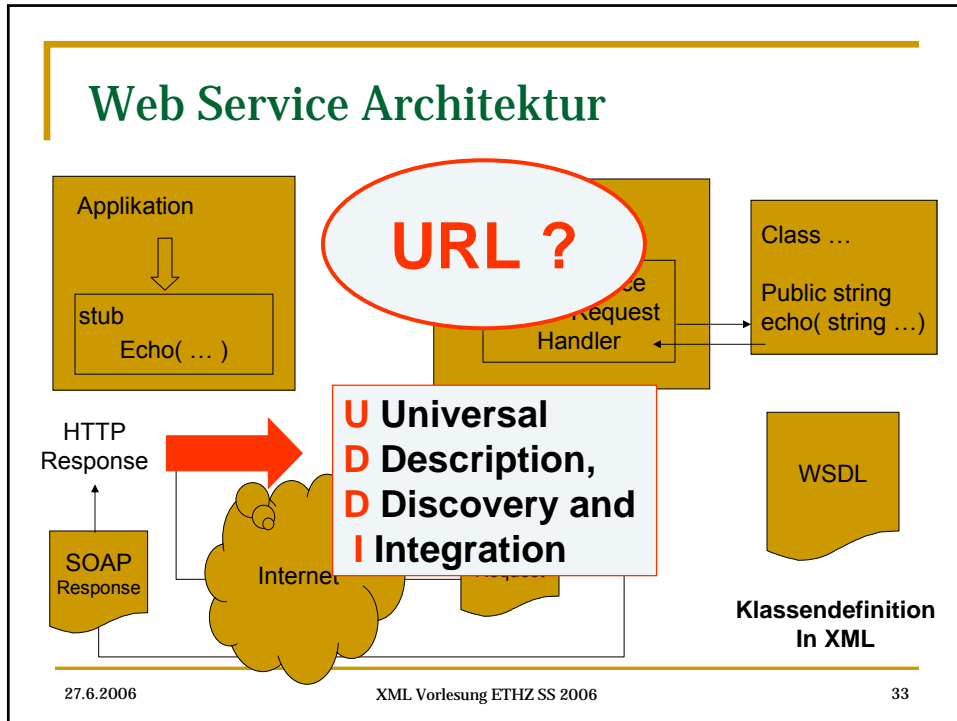
**2.2 Query Terms - <q>**

**Default Search**

By default, Google only returns pages that include all of the terms in the query string. There is no need to include "AND" between terms. Keep in mind that the order of the terms in the query will impact the search results.

**Stop Words**





## UDDI

A vertical stack of four colored boxes representing the protocol stack for UDDI. From top to bottom: a yellow box labeled 'UDDI', an orange box labeled 'SOAP', a light green box labeled 'XML', and a dark brown box labeled 'HTTP über TCP/IP'.

- Universal Description, Discovery and Integration
- Verzeichnis und Klassifikation von Web Services
- selbst ein Web Service

---

27.6.2006 XML Vorlesung ETHZ SS 2006 35

## UDDI Datenstruktur

The diagram illustrates the UDDI data structure using class-like boxes and relationships:

- businessEntity** (yellow box):
  - name
  - description
  - contacts
  - categories
  - businessKey
- businessService** (yellow box):
  - name
  - description
  - serviceKey
- bindingTemplate** (yellow box):
  - bindingKey
  - description
  - accessPoint
- tModel** (yellow box):
  - tModelkey
  - name
  - description
  - overview
  - categories

Relationships:

- A **businessEntity** (1) is associated with **businessService** (0..\*) via a diamond-headed line.
- A **businessService** (1) is associated with **bindingTemplate** (0..\*) via a diamond-headed line.
- A **bindingTemplate** (0..n) is associated with a **tModel** (1) via a dashed arrow.

---

27.6.2006 XML Vorlesung ETHZ SS 2006 36

## UDDI Komponenten

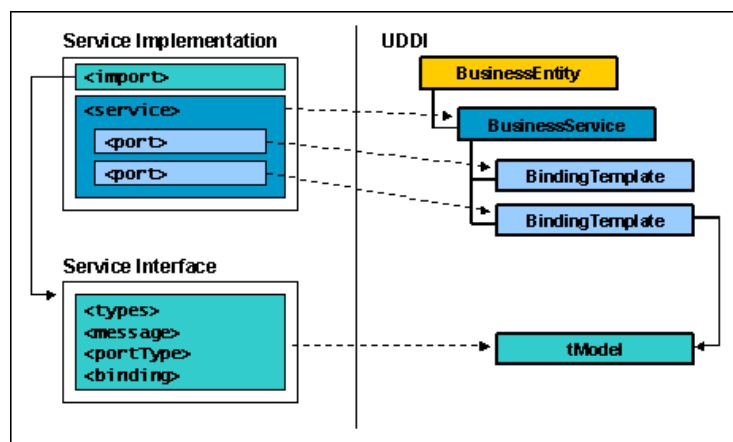
- businessEntity
  - Firma und/oder Person, die Dienste anbietet
- businessService
  - logische Klassifizierung von Diensten
- bindingTemplate
  - verschiedene Arten, einen Dienst zu benutzen
- tModel
  - Implementierung eines Dienstes
  - u.U. Verweis auf Beschreibung eines Dienstes
  - ermöglicht das Auffinden von bekannten Diensten

27.6.2006

XML Vorlesung ETHZ SS 2006

37

## Abbildung WSDL → UDDI



aus: <http://www-106.ibm.com/developerworks/webservices/library/ws-wsdl/>

27.6.2006

XML Vorlesung ETHZ SS 2006

38

## Zusammenfassung

- XML als Präsentationsschicht des Internet
  - eher gewachsen als designt
  - für viele Anwendungen mässig geeignet
    - Datenvolumen in B2B-Szenarien ist problematisch
- ASN.1 als 15 Jahre alte akademische Lösung
  - in kleinen Bereichen erfolgreich
  - als allgemeine Präsentationsschicht ein Misserfolg
- XML und ASN.1 als verschiedene Ansätze
  - XML als der Zufallserfolg aus der Dokumentenwelt
  - ASN.1 als durchgeplant, aber zu kompliziert