

## 7363 - Web-basierte Anwendungen

Eine Vertiefungsveranstaltung  
mit Schwerpunkt auf XML-Technologien

## WSDL

Web Services Description Language

- Warum eine Beschreibungssprache für Web Services?
  - Dokumentation der Schnittstelle
    - Systematischere, strukturierte Entwicklung und Pflege von WS
    - Analogien: C-Headerdateien, IDL, *reflection*, *type libraries*
  - Grundlage für Code-Generierung
    - Entlastung der Client-Entwicklung
  - Grundlage für flexiblen Einsatz
    - Schnittstellenänderungen lassen sich z.T. automatisch einarbeiten
    - Neue WS lassen sich schnell und leicht verwenden
    - Zusammenspiel mit UDDI: Finden, konfigurieren, nutzen

- Entwicklung von WSDL
  - IBM: NASSL, 2000
    - Network Accessibility Service Specification Language
    - Verwendete XML zur Beschreibung von WS Interfaces
    - Verwendet XML Schema zur Beschreibung von Datentypen
  - Microsoft: SCL, 2000
    - Service Contract Language
    - Verwendete XML zur Beschreibung von WS Interfaces
    - Verwendet XDR (Microsoft's "XML Data Reduced") zur Beschreibung von Datentypen
    - Nachfolger: SDL, zusammen mit Visual Studio.NET
  - Problem dabei:
    - Interoperabilität auf der Beschreibungsebene??
  - Lösung: WSDL

## WSDL

- Entwicklung von WSDL
  - Microsoft, IBM, Ariba: Erste WSDL-Version
    - Nutzung von XML Schema (bzw. Vorläufer), SOAP, MIME
  - WSDL 1.1, 15. März 2001
    - W3C Note: <http://www.w3.org/TR/wsdl>
  - WSDL 2.0
    - Status: Working Draft
    - Zuletzt aktualisiert: August 2004
    - Neu: Vererbungskonzept, per Attribut "extends"
    - Verändert: Begriff / Element "port" wird ersetzt durch "interface"

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 5

## WSDL

- WSDL 1.1
  - Dokumentation: <http://www.w3.org/TR/wsdl>
  - Namespace: <http://schemas.xmlsoap.org/wsdl/>
- Ansatz:
  - Beschreibung von WS als Sammlung von Netzwerk-Endpunkten (*ports / interfaces*)
  - Trennung der abstrakten Interface- und Nachrichtentyp-Beschreibung von der konkreten Implementierung (im Sinne von *bindings*, *encoding*, einzelne Nachrichten)
  - XML-Dokumententyp **definitions** als Grundlage
  - Ausgiebiger Gebrauch von XML Schema!
    - Teile des WSDL-Schemas sind von "Schema" übernommen, etwa die Elemente "include", "import" & das Attribut "targetNameSpace"

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 6

## WSDL

- WSDL: Anwendungsszenarien

```

graph TD
    subgraph "Download"
        A[Anwender] -- Download --> B[WSDL-Datei]
        B --> C[Codegenerator]
        C --> D[WS consumer]
        D -- Anwendung --> E[WS provider]
        E -- Veröffentlichung/Generierung --> B
    end
    subgraph "Design"
        F[Entwickler] -- Design --> G[WSDL-Datei]
        G --> H[Codegenerator]
        H --> I[WS consumer]
        H --> J[WS provider]
    end
  
```

Einfache Benutzerschnittstelle, etwa CLI

Skelett-Code, zu Ende zu entwickeln

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 7

## WSDL

- WSDL 2.0
  - Dokumentation: <http://www.w3.org/TR/wsdl20>
  - (vorl.)Namespace: <http://www.w3.org/2004/03/wsdl>
  - Status: Draft, zuletzt aktualisiert: August 2004
  - Geändert:
    - Port --> Interface
  - Neu:
    - Vererbungskonzept! Attribut "extends" in Element "interface"
- Empfehlung
  - Noch zu früh für den praktischen Einsatz, aber:
  - Entwicklung beobachten!

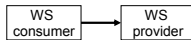
17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 8

Nachtrag zu SOAP: **Operationstypen**

– Je nach Anzahl und Reihenfolge von **input**-, **output**-, und **fault**-Elementen, die zwischen zwei SOAP-Knoten ausgetauscht werden, unterscheidet man folgende 4 abstrakte Operationstypen:

– **One-way**

- WS consumer sendet



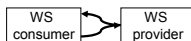
– **Request-response**

- WS consumer sendet, WS provider antwortet



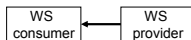
– **Solicit-response**

- WS provider agiert, WS consumer reagiert



– **Notification**

- WS provider agiert



– In der Praxis dominieren die Typen "One-way" und "Request-response". (Publikumsfrage: Warum wohl?)

**WSDL: Eine beispiel-orientierte Einführung**

Die WSDL-Datei vom Babelfish-Service mit Auszügen aus dem Google-API

**definitions**

WSDL-Dokumentenelement

– **types**

- **schema**

Optionaler Container

Schemata für Dokumenten- (EDI-) Modus

– **message**

- **part**

Abstrakter Teil: Nachrichtenaufbau

Teilnachricht, etwa ...Request, ...Response

– **portType**

- **operation**

Abstrakter Teil: Zuordnung der Nachrichtenarten zu logischen Ports/Interfaces

– **binding**

- **operation**

Konkreter Teil: Anbindung eines Port-Typen an Transportprotokoll & Codierung

– **service**

- **port**

Konkreter Teil:

Assoziation eines konkreten Ports mit einem Binding. Achtung: Typen müssen zueinander passen.

Lernen von WSDL am Beispiel

– "Babelfish-Service"

- Grundlagen: SOAP 1.1, WSDL 1.1, RPC-Stil, HTTP *binding*
- Input: Zwei Strings
  - Quell- und Zielsprache, ISO-codiert, etwa: "en\_de"
  - String mit zu übersetzendem Text, max. 5k Zeichen
- Output (Rückgabewert): Ein String
  - Der übersetzte Text

– "Google-Service"

- Analog zu Babelfish, aber mit komplexeren Datentypen

– Vorgehen

- Analyse der WSDL-Dateien dieser Dienste (*bottom-up*)
- Erstellung einer Babelfish Client-Anwendung
- Test

## WSDL

- Die WSDL-Datei zum WS "Babelfish"
  - Das Dokumentenelement
    - Wenig Bemerkenswertes, nur viele Namensraum-Deklarationen
    - In rot: Default-Namensraum (WSDL-Namensraum)

```

<?xml version="1.0"?>
<definitions name="BabelFishService"
  xmlns:tns="http://www.xmethods.net/sd/BabelFishService.wsdl"
  targetNamespace=
    "http://www.xmethods.net/sd/BabelFishService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

<!-- Hier der eigentliche Inhalt, s.u. -->

</definitions>

```

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 13

## WSDL

- Beschreibung eines Web Service
  - Das "service"-Element

```

<service name="BabelFishService">
  <documentation>Translates text of up to 5k in length,
  between a variety of languages.</documentation>
  <port name="BabelFishPort" binding="tns:BabelFishBinding">
    <soap:address location=
      "http://services.xmethods.net:80/perl/soaplite.cgi"/>
  </port>
</service>

```

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 14

## WSDL

- Beschreibung eines Web Service
  - Das "service"-Element
    - Der Dienst erhält einen Namen
    - Ihm wird ein (zunächst abstraktes) Interface (Port) zugewiesen
    - Das "binding" dazu erfolgt
      - durch Angabe eines Binding-Typs (zunächst nur ein Name)
      - durch Angabe einer konkreten Adresse (hier: ein URL)
  - Das Unter-Element "documentation"
    - Universelles Unter-Element - nutzen Sie es!
    - Es wirkt als Container, nimmt also neben *char data* auch beliebige (eigene) Unter-Elemente sowie Attribute auf.

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 15

## WSDL

- Beschreibung eines Web Service
  - Das "binding"-Element

```

<binding name="BabelFishBinding" type="tns:BabelFishPortType">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="BabelFish">
    <soap:operation soapAction=
      "urn:xmethodsBabelFish#BabelFish"/>
    <input><soap:body
      use="encoded" namespace="urn:xmethodsBabelFish"
      encodingStyle=
        "http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output><soap:body
      use="encoded" namespace="urn:xmethodsBabelFish"
      encodingStyle=
        "http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>
</binding>

```

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 16

**WSDL**

- Beschreibung eines Web Service
  - Das **"binding"**-Element
    - Das Binding erhält einen Namen
    - Ihm wird ein (noch zu spezifizierender) Interfacetyp (Porttyp) zugewiesen
    - Einzelheiten beschreiben Unter-Elemente
  - Das Unter-Element **"soap:binding"**
    - Hier werden SOAP-Stil (hier „rpc“) und Transportprotokoll (HTTP, erkennbar an einem reservierten Namensraum-URL) festgelegt.
    - Attribut „style“: Werte sind
      - „rpc“ SOAP-RPC Stil, eine Konvention zum Inhalt von „body“: Je ein Unterelement, abgeleitet von Methodennamen (\*Request, \*Response, \*Fault)
      - „document“ Dokumenten- oder EDI-Stil: Inhalt von „body“ kann frei gewählt werden, ggf. beschrieben durch Schema in „types“

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 17

**WSDL**

- Beschreibung eines Web Service
  - Das (hier einzige) Unter-Element **"operation"**
    - Beschreibung, nach welchen Regeln Übergabeparameter „verpackt“ sind. Hier: SOAP *encoding* gemäß SOAP 1.1.
    - Attribut „use“: Werte sind
      - encoded Gemäß SOAP 1.1, 1.2 oder anderen Verfahren (Ruby scheint eine ASP.NET-Variante zu kennen). Parameter werden gemäß Inhalt von „encodingStyle“ codiert!
      - literal Keine Codierung – Dokumententeile werden einfach durchgereicht und nicht als Parameter interpretiert.
  - Die Kombinationen der Attribute „style“ und „use“
    - rpc / encoded Häufigster Fall: SOAP-RPC, meist via http
    - document / literal Typisch für Dokumentenmodus
    - rpc / literal Grundsätzlich möglich
    - document / encoded Grundsätzlich möglich

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 18

**WSDL**

- Beschreibung eines Web Service
  - Das **"portType"**-Element

```

<portType name="BabelFishPortType">
  <operation name="BabelFish"
    parameterOrder="translationmode sourcedata" >
    <input message="tns:BabelFishRequest" />
    <output message="tns:BabelFishResponse" />
  </operation>
</portType>

```

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 19

**WSDL**

- Beschreibung eines Web Service
  - Das **"portType"**-Element
    - Hier wird nun der Interfacetyp näher beschrieben
    - Ihm werden evtl. mehrere Operationen zugewiesen.
  - Das Unter-Element **"operation"**
    - Wir kennen es schon als Unterelement von "binding"
    - Im vorliegenden Kontext benennen seine Unterelemente den eigentlichen Aufbau der Body-Inhalte (hier: sowohl *input* als auch *output*, da wir ein RPC-Szenario verwenden)
    - Die Inhalte der Attribute "message" sind Elementnamen des Namensraums, der den für diesen Service verwendeten Elementen zugewiesen wurde.
    - Das Attribut „parameterOrder“ ist optional und nicht im originalen BabelFish-Beispiel enthalten. Es bringt eine determinierte Reihenfolge in die sonst hash-artige Parameterliste (parts).

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 20

**WSDL**

- Beschreibung eines Web Service
  - Das "message"-Element

```

<message name="BabelFishRequest">
  <part name="translationmode" type="xsd:string"/>
  <part name="sourcedata" type="xsd:string"/>
</message>

<message name="BabelFishResponse">
  <part name="return" type="xsd:string"/>
</message>

```

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 21

**WSDL**

- Beschreibung eines Web Service
  - Das "message"-Element
    - Bei SOAP-RPC existiert nur ein Unter-Element von "Body" und dies trägt den Namen der Methode/Prozedur.
    - Das Element "message" beschreibt genau dieses Unter-Element von "Body", sowohl im *request*- als auch im *response*-Fall.
  - Das Unter-Element "part"
    - Unter-Elemente von SOAP-RPC Methodenelementen sind bekanntlich die Teile eines *struct*. Jedes solche Teil wird hier als "part" beschrieben, mit Namen und Datentyp.
    - Interessant ist der Datentyp: In einfachen Fällen ist er einer der XML Schema-Datentypen, aber auch komplexe Datentypen aus eigenen Schemata (s.u.) werden hier oftmals verwendet.

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 22

Fachhochschule Wiesbaden - Fachbereich Informatik

## WSDL: Weitere Elemente

types: Komplexe Datentypen  
(Auszüge aus der WSDL-Datei von Google)  
import & include

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 23

**WSDL**

- Beschreibung eines Web Service
  - Das "types"-Element

```

<types>
  <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
    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="estimateIsExact" 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> <!-- etc. -->

```

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 24

**WSDL**

- Beschreibung eines Web Service
  - Das "types"-Element

```

<xsd:complexType name="ResultElement">
  <xsd:all>
    <xsd:element name="summary" type="xsd:string"/>
    <xsd:element name="URL" type="xsd:string"/>
    <xsd:element name="snippet" type="xsd:string"/>
    <xsd:element name="title" type="xsd:string"/>
    <xsd:element name="cachedSize"
      type="xsd:string"/>
    <xsd:element name="relatedInformationPresent"
      type="xsd:boolean"/>
    <xsd:element name="hostName" type="xsd:string"/>
    <xsd:element name="directoryCategory"
      type="typens:DirectoryCategory"/>
    <xsd:element name="directoryTitle"
      type="xsd:string"/>
  </xsd:all>
</xsd:complexType> <!-- Fortsetzung -->

```

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 25

**WSDL**

- Beschreibung eines Web Service
  - Das "types"-Element

```

<xsd:complexType name="ResultElementArray">
  <xsd:complexContent>
    <xsd:restriction base="soapenc:Array">
      <xsd:attribute ref="soapenc:arrayType"
        wsdl:arrayType="typens:ResultElement[]" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<!-- usw., schließlich: -->
</xsd:schema>
</types>

```

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 26

**WSDL**

- Beschreibung eines Web Service
  - Das "types"-Element
    - Dies ist eine Hülle für Schemadokumente
    - Entweder schreibt man gleich ein (kleines) Schemadokument hier hinein, oder man importiert es.
    - Beispiele finden Sie in der WSDL-Datei von Google!
  - Die Elemente "include" und "import"
    - Sie gestatten (analog zu Schema) die Verteilung eines Schemadokuments auf mehrere Dateien bzw. Ressourcen.
    - Verwenden Sie "import", um WSDL-Bestandteile aus anderen Namensräumen einzubeziehen.
    - "include" (nur WSDL 2.0!) wirkt dagegen wie das direkte Einbeziehen eines Textblocks.

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 27

Fachhochschule Wiesbaden - Fachbereich Informatik

**WSDL: Test & Demo**

17.05.2005 H. Werntges, FB Informatik, FH Wiesbaden 28



- Test: Erzeugung einer Babelfish-Clientanwendung
  - Werkzeug / Entwicklungsumgebung:
    - Ruby 1.8.x, SOAP4R 1.5.2 (das *separate* Paket!)
  - Code-Generator:
    - `wSDL2ruby.rb` aus dem *separat* erhältlichen SOAP4R-Paket
  - Vorgehen:
    - Dateien generieren
    - Client-Datei mit einem minimalen CLI (*command line interface*) ausstatten
    - Ggf. Umgebungsvariablen setzen, hier:

```
$ export SOAP_USE_PROXY=on
$ export HTTP_PROXY=$http_proxy
```
    - On-line Demo auf einem der Linuxcluster-Rechner!