

WSDL

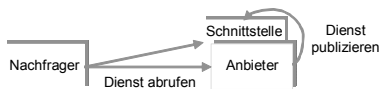
Heutige Vorlesung

- Prinzipieller Aufbau von WSDL-Beschreibungen
- Beschreibung von Protokoll-Bindungen in WSDL
- Vor- und Nachteile von WSDL

Lernziel

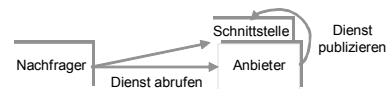
- Google-WSDL lesen und erweitern können

Wozu WSDL?



- Client möchte bestimmten Web-Dienst abrufen
- Client benötigt hierfür Beschreibung der Schnittstelle:
 - Struktur des Aufrufes: Name, Eingangsparameter, Ergebnis
 - Protokoll und Adresse
- ähnlich wie Java-IDL, jedoch unabhängig von Plattformen, Programmiersprachen und Protokollen

Wie wird WSDL verwendet?



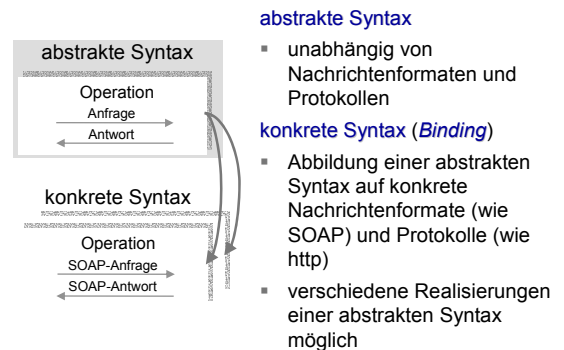
- aus WSDL-Beschreibung können automatisch Stubs generiert werden
- Stubs abstrahieren von WSDL, SOAP und Protokoll
- erleichtert Einbindung des Dienstes zur Entwicklungszeit erheblich
- theoretisch damit auch dynamische Einbindung zur Laufzeit möglich

Web Services Description Language



- XML-Standard
- beschreibt Netzwerkdienste als Kommunikationsendpunkte (*Ports*), die bestimmte Nachrichten austauschen
- *nicht* auf SOAP-Nachrichten beschränkt
- aktuelle Version 1.1 (2001)
- *kein* offizieller W3C-Standard, sondern *W3C-Note* von IBM/Microsoft)

Abstrakte vs. konkrete Syntax



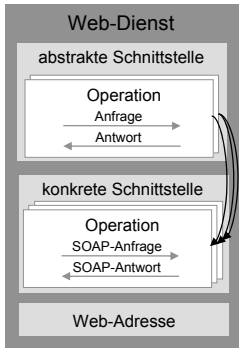
abstrakte Syntax

- unabhängig von Nachrichtenformaten und Protokollen

konkrete Syntax (*Binding*)

- Abbildung einer abstrakten Syntax auf konkrete Nachrichtenformate (wie SOAP) und Protokolle (wie http)
- verschiedene Realisierungen einer abstrakten Syntax möglich

Abstrakte und konkrete Schnittstelle

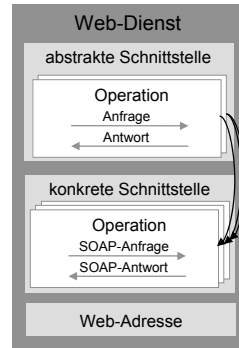


- **abstrakte Schnittstelle (Port Type):** Menge von abstrakten Operationen
- **konkrete Schnittstelle (Binding):** Abbildung auf konkrete Protokolle und Nachrichtenformate
- Dienst kann verschiedene Realisierungen haben: z.B. SOAP/HTTP- und SOAP-SMTP-Bindung

© Klaus Schild, 2004

7

Port

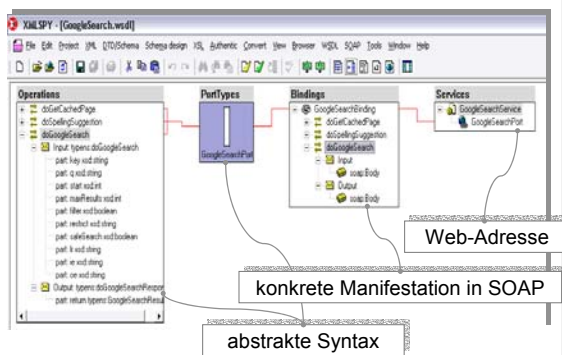


- **Web-Dienst (Service):** besteht aus mindestens einem Kommunikationsendpunkt (Port)
- Port = konkrete Schnittstelle + Web-Adresse

© Klaus Schild, 2004

8

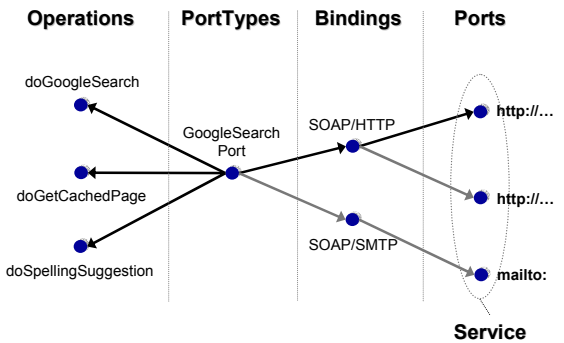
Die WSDL-Beschreibung von Google



© Klaus Schild, 2004

9

Grundstruktur



© Klaus Schild, 2004

10

XML-Syntax

```
<?xml version="1.0"?>
<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  <types>...</types>
  <message name="doGoogleSearch">...</message>
  <message name="doGoogleSearchResponse">...</message>
  <portType name="GoogleSearchPort">...</portType>
  <binding name="GoogleSearchBinding"
    type="typens:GoogleSearchPort">
  ...</binding>
  <service name="GoogleSearchService">...</service>
</definitions>
```

© Klaus Schild, 2004

11

XML-Syntax

```
<?xml version="1.0"?>
<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  <types>...</types>
  <message name="doGoogleSearch">...</message>
  <message name="doGoogleSearchResponse">...</message>
  <portType name="GoogleSearchPort">...</portType>
  <binding name="GoogleSearchBinding"
    type="typens:GoogleSearchPort">
  ...</binding>
  <service name="GoogleSearchService">...</service>
</definitions>
```

- **definitions:** Wurzel-Element
- **types:** Definition von Datentyp (normalerweise mit XML-Schema)
- **portType:** abstrakte Schnittstelle, d.h. Menge von Operationen, die abstrakte Nachrichten austauschen
- **message:** abstrakte Nachricht
- **binding:** Abbildung eines portTypes auf konkrete Protokolle und Nachrichtenformate
- **service:** Menge von ports (jeweils binding + Web-Adresse)

© Klaus Schild, 2004

12

Dokumentwurzel

```
<?xml version="1.0"?>
<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  ...
</definitions>
```

- Wurzel-Element immer definitions aus dem Namensraum für WSD (kein W3C-Namensraum)
- WSDL-Beschreibung kann Namen haben.
- WSDL-Beschreibung kann eigenen Ziel-Namensraum definieren.

Datentypen

```
<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>...</types>
  <message name="doGoogleSearch">...</message>
  <message name="doGoogleSearchResponse">...</message>
  <portType name="GoogleSearchPort">...</portType>
  <binding name="GoogleSearchBinding"
    type="typens:GoogleSearchPort">
    ...
  </binding>
  <service name="GoogleSearchService">...</service>
</definitions>
```

Datentypen

```
<types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:GoogleSearch">
    ...
  </schema>
</types>
```

- Datentypen, die beim Austausch von Nachrichten relevant sind
- Verwendung von XML-Schema empfohlen, jedes andere Typsystem aber auch erlaubt
- Beachte: XML-Schema kann auch verwendet werden, wenn die Nachrichten *nicht* in XML übertragen werden.

Datentyp für Google-Suchresultat

```
<types>
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:typens="urn:GoogleSearch"
    targetNamespace="urn:GoogleSearch">
    <xsd:complexType name="GoogleSearchResult">
      <xsd:all>
        <xsd:element name="estimatedTotalResultsCount" type="xsd:int"/>
        <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:all>
    </xsd:complexType>
  </schema>
</types>
```

- komplettes XML-Schema
- Ziel-Namensraum (normalerweise identisch mit Ziel-Namensraum von WSDL)

Abstrakte Nachrichten

```
<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>...</types>
  <message name="doGoogleSearch">...</message>
  <message name="doGoogleSearchResponse">...</message>
  <portType name="GoogleSearchPort">...</portType>
  <binding name="GoogleSearchBinding"
    type="typens:GoogleSearchPort">
    ...
  </binding>
  <service name="GoogleSearchService">...</service>
</definitions>
```

- abstrakte Nachrichten, vom Web-Dienst empfangen oder gesendet

Abstrakte Nachrichten

```
<message name="doGoogleSearchResponse">
  <part name="return" type="typens:GoogleSearchResult"/>
</message>
```

- definiert abstrakte Nachricht mit Namen (Referenz)
- Definitionen werden in portType oder binding verwendet
- können verschiedene logische Bestandteile (**part**) haben, z.B.:
 - Parameter eines entfernten Prozeduraufrufs
- jedes dieser Bestandteile hat ebenfalls einen Namen
- Reihenfolge der logischen Bestandteile unerheblich

Nachrichten mit strukturiertem Inhalt

zwei unterschiedliche Modellierungen

1. nur *ein* Bestandteil (part), diesem wird komplexer Datentyp zugeordnet:

```
<message name="doGoogleSearchResponse">
  <part name="return" type="typens:complexType"/>
</message>
```

typens:complexType könnte z.B. 2 Parameter enthalten

2. *mehre* Bestandteile (parts), denen jeweils ein Element aus types zugeordnet wird:

```
<message name="doGoogleSearchResponse">
  <part name="param1" element="typens:param1"/>
  <part name="param2" element="typens:param2"/>
</message>
```

© Klaus Schild, 2004

19

Wie abstrakt ist eine abstrakte Nachricht?

- message beschreibt *abstrakte* Syntax
- Konkrete Schnittstelle (binding) bildet abstrakte Syntax auf bestimmte Protokolle und Nachrichtenformate ab.
- Dennoch kann abstrakte Nachricht einer konkreten Realisierung (binding) sehr ähnlich sein.

Wie abstrakt eine abstrakte Syntax ist, zeigt also erst die konkrete Schnittstelle (binding).

© Klaus Schild, 2004

20

Abstrakte Schnittstelle

```
<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/"
<types>...</types>
<message name="doGoogleSearch">...</message>
<message name="doGoogleSearchResponse">...</message>
<portType name="GoogleSearchPort">...</portType>
<binding name="GoogleSearchBinding"
  type="typens:GoogleSearchPort">
  ...
</binding>
<service name="GoogleSearchService">...</service>
</definitions>
```

© Klaus Schild, 2004

21

Abstrakte Schnittstelle

```
<message name="doGoogleSearch">...</message>
<message name="doGoogleSearchResponse">...</message>
<portType name="GoogleSearchPort">
  <operation name="doGoogleSearch">
    <input message="typens:doGoogleSearch"/>
    <output message="typens:doGoogleSearchResponse"/>
  </operation>
  ...
</portType>
```

- definiert abstrakte Schnittstelle als Menge von abstrakten Operationen (operations)
- Definition wird in binding verwendet

© Klaus Schild, 2004

22

Abstrakte Schnittstelle

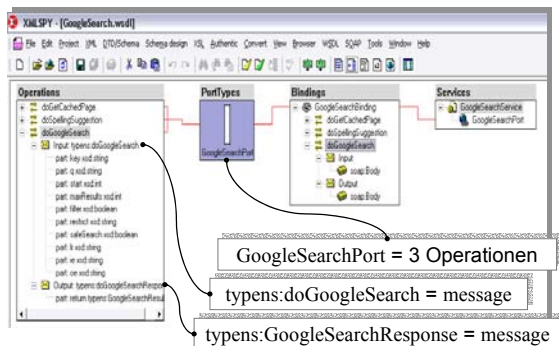
```
<message name="doGoogleSearch">...</message>
<message name="doGoogleSearchResponse">...</message>
<portType name="GoogleSearchPort">
  <operation name="doGoogleSearch">
    <input message="typens:doGoogleSearch"/>
    <output message="typens:doGoogleSearchResponse"/>
  </operation>
  ...
</portType>
```

- **operation:** definiert einfaches Interaktionsmuster mit Eingangs- und Ausgangs-Nachrichten.
- Definition wird in binding verwendet
- verwendet keine Datentypen, sondern Nachrichten

© Klaus Schild, 2004

23

Beispiel



© Klaus Schild, 2004

24

Mögliche Interaktionsmuster

```
<operation name="...">
  <input message="..."/>
</operation>
```

Einweg (*one way, fire and forget*)

```
<operation name="...">
  <input message="..."/>
  <output message="..."/>
</operation>
```

Anfrage-Antwort (*request-response*)

```
<operation name="...">
  <output message="..."/>
</operation>
```

Benachrichtigung (*notification*)

```
<operation name="...">
  <output message="..."/>
  <input message="..."/>
</operation>
```

Benachrichtigung-Antwort (*notification-response*)

Abstrakte Interaktionsmuster

- Anfrage-Antwort-Muster müssen *nicht* mit *einer* Netzwerkcommunication (z.B. HTTP request/response) realisiert werden.
- auch Realisierung z.B. mit zwei unabhängigen Kommunikationen (z.B. E-Mails) möglich
- entfernter Prozeduraufruf daher auch mit SMTP realisierbar
- Realisierung wird erst in der konkreten Schnittstelle (binding) festgelegt

Komplexe Interaktionsmuster

- Registrierung zum Börsenticker
- ← Bestätigung der Registrierung
- ← aktueller Börsenkurs (Benachrichtigung)



```
<operation name="...">
  <input message="..."/>
  <output message="..."/>
  <output message="..."/>
</operation>
```

In WSDL *nicht* erlaubt!

Entfernte Prozeduraufrufe

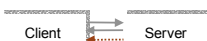
```
<operation name="doGoogleSearch" parameterOrder="key q start ...">
  <input message="typens:doGoogleSearch"/>
  <output message="typens:doGoogleSearchResponse"/>
</operation>
```

- Reihenfolge der Bestandteile (parts) unerheblich
- bei entfernten Prozeduraufrufen Reihenfolge der Input-Parameter (parts) aber häufig wichtig
- Reihenfolge kann mit `parameterOrder` festgelegt werden.
- zusätzliche Konvention: In/Out-Parameter erscheinen sowohl in input- als auch in output-Nachricht.

Fehlermeldungen

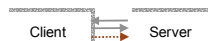
Anfrage-Antwort

```
<operation name="...">
  <input message="..."/>
  <output message="..."/>
  <fault message="..."/>
</operation>
```



Benachrichtigung-Antwort

```
<operation name="...">
  <output message="..."/>
  <input message="..."/>
  <fault message="..."/>
</operation>
```



- Statt Antwort kann auch Fehler auf Anwendungsebene gemeldet werden.
- nur eine Fehlermeldung; kann aber komplexen Datentyp mit unterschiedlichen Fehlertypen (`xsd:choice`) haben.

Konkrete Schnittstelle

```
<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>...</types>
  <message name="doGoogleSearch">...</message>
  <message name="doGoogleSearchResponse">...</message>
  <portType name="GoogleSearchPort">...</portType>
  <binding name="GoogleSearchBinding"
    type="typens:GoogleSearchPort">
    ...
  </binding>
  <service name="GoogleSearchService">...</service>
</definitions>
```

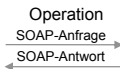
Konkrete Schnittstelle



abstrakte Syntax



konkrete Syntax



- **konkrete Schnittstelle (Binding):** Abbildung einer abstrakten Schnittstelle (portType) auf konkrete Nachrichtenformate und Protokolle
- jede abstrakte Operation des PortType wird abgebildet
- für ein PortType verschiedene Abbildungen (Bindings) möglich

Konkrete Schnittstelle



```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">  
  Erweiterungselement  
  <operation name="doGoogleSearch">  
    Erweiterungselement  
    <input>  
      Erweiterungselement  
    </input>  
    <output>  
      Erweiterungselement  
    </output>  
  </operation>  
  ...  
</binding>
```

- definiert konkrete Schnittstelle
- Definition wird in service benutzt
- type: die abgebildete abstrakte Schnittstelle (portType)
- mehrere binding-Elemente für eine abstrakte Schnittstelle erlaubt

Erweiterungselemente



```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">  
  Erweiterungselement  
  <operation name="doGoogleSearch">  
    Erweiterungselement  
    <input>  
      Erweiterungselement  
    </input>  
    <output>  
      Erweiterungselement  
    </output>  
  </operation>  
  ...  
</binding>
```

- Abbildung mit sog. Erweiterungselementen kodiert
- Informationen über die Abbildung auf allen Ebenen:
 - Binding selbst
 - Operation
 - Input- und Output-Nachricht
 - Fehlermeldung

Erweiterungselemente



- engl. *extensibility elements*
- Platzhalter für spezielle Bindings
- also Platzhalter in der WSDL-Grammatik für entsprechende Erweiterungen
- WSDL 1.1 definiert drei Bindings:
 - SOAP-Binding
 - HTTP-Binding
 - MIME-Bindingwerden gleich vorgestellt, vorher noch letzten Teil einer WSDL-Beschreibung

Web-Dienst (Service)



```
<definitions name="GoogleSearch"  
  targetNamespace="urn:GoogleSearch"  
  ...  
  xmlns="http://schemas.xmlsoap.org/wsdl/"  
<types>...</types>  
<message name="doGoogleSearch">...</message>  
<message name="doGoogleSearchResponse">...</message>  
<portType name="GoogleSearchPort">...</portType>  
<binding name="GoogleSearchBinding"  
  type="typens:GoogleSearchPort">  
  ...  
</binding>  
<service name="GoogleSearchService">...</service>  
</definitions>
```

Web-Dienst (Service)



```
<service name="GoogleSearchService">  
  <port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">  
    Erweiterungselement (Web-Adresse)  
  </port>  
  ...  
</service>
```

- Web-Dienst = Menge von Kommunikationsendpunkten (Ports)
- mehrere service-Elemente erlaubt, falls unterschiedliche Web-Dienste angeboten werden, z.B.: GoogleSearchService und GoogleClassificationService

Kommunikationsendpunkt

```
<port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">
  Erweiterungselement (Web-Adresse)
</port>
```

- definiert einen benannten Kommunikationsendpunkt: Binding + genau eine Web-Adresse
- Web-Adresse mit einem Erweiterungselement spezifiziert, z.B.:

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

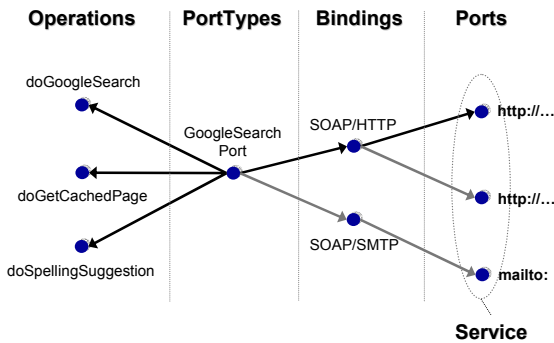
Welche Ports bilden einen Service?

```
<service name="GoogleSearchService">
  <port...>...</port>
  <port...>...</port>
</service>
<service name="GoogleClassificationService">
  <port...>...</port>
</service>
```

Ports eines Web-Dienstes (service)

- kommunizieren *nicht* untereinander
- Ports mit derselben abstrakten Schnittstelle, aber versch. Bindings oder Web-Adressen: semantisch äquivalente Alternativen

Grundstruktur



SOAP-Binding



SOAP-Bindung von WSDL

```
<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  <types>...</types>
  <message name="doGoogleSearch">...</message>
  <message name="doGoogleSearchResponse">...</message>
  <portType name="GoogleSearchPort">...</portType>
  <binding name="GoogleSearchBinding"
    type="typens:GoogleSearchPort">
    ...
  </binding>
  <service name="GoogleSearchService">...</service>
</definitions>
```

SOAP-Bindung: Erweiterungselemente

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
  Erweiterungselement soap:binding
  <operation name="doGoogleSearch">
    Erweiterungselement soap:operation
    <input>
      Erweiterungselemente soap:body und soap:header
    </input>
    <output>
      Erweiterungselemente soap:body und soap:header
    </output>
    <fault>
      Erweiterungselement soap:fault
    </fault>
  </operation>
</binding>
```

- Erweiterungselemente beschreiben Abbildung auf SOAP-Nachricht
- Beachte: WSDL 1.0 benutzt SOAP 1.1

soap:binding

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
  Erweiterungselement soap:binding
  <operation name="doGoogleSearch">
    Erweiterungselement soap:operation
    <input>
      Erweiterungselemente soap:body und soap:header
    </input>
    <output>
      Erweiterungselemente soap:body und soap:header
    </output>
    <fault>
      Erweiterungselement soap:fault
    </fault>
  </operation>
</binding>
```

© Klaus Schild, 2004

43

soap:binding

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
  <soap:binding xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="doGoogleSearch">
    ...
  </operation>
</binding>
```

- **soap:binding**: besagt, dass portType mit SOAP realisiert wird
- **transport**: wie SOAP-Nachrichten übertragen werden
- **Beachte**: HTTP meint hier HTTP-POST
- auch möglich: transport="http://schemas.xmlsoap.org/soap/smtp"
- **style**: entfernte Prozeduraufrufe (rpc) oder Messaging (document)

© Klaus Schild, 2004

44

SOAP-Stile

style="rpc"

```
<body>
  <procedure-name>
  <part-1>...<part-1>
  ...
  <part-n>...<part-n>
  </procedure-name>
</body>
```

style="document"

```
<body>
  <part-1>...<part-1>
  ...
  <part-n>...<part-n>
</body>
```

- legt Struktur des SOAP-Nachrichteninhalts (body) fest, darüber hinaus *keine* Bedeutung
- SOAP-Stil in soap:binding gilt für *alle* Operationen, kann aber in jeder einzelnen Operation überschrieben werden.

© Klaus Schild, 2004

45

soap:body und soap:header

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
  Erweiterungselement soap:binding
  <operation name="doGoogleSearch">
    Erweiterungselement soap:operation
    <input>
      Erweiterungselemente soap:body und soap:header
    </input>
    <output>
      Erweiterungselemente soap:body und soap:header
    </output>
    <fault>
      Erweiterungselement soap:fault
    </fault>
  </operation>
</binding>
```

© Klaus Schild, 2004

46

soap:body

```
<operation name="doGoogleSearch">
  <soap:operation soapAction="urn:GoogleSearchAction">
  <input>
    <soap:body use="literal"/>
  </input>
  <output>...</output>
</operation>
```

Komplette input-Nachricht wird *unverändert* in SOAP-Body übernommen.

- **soap:body**: Abbildung einer abstrakten input- oder output-Nachricht auf den SOAP-Nachrichteninhalt (body)
- **use="literal"**: Abstrakte Nachricht wird unverändert übernommen

© Klaus Schild, 2004

47

soap:body

```
<input>
  <soap:body use="encoded"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
</input>
```

Komplette input-Nachricht wird *kodiert* in den SOAP-Body übernommen.

- **use="encoded"**: Abstrakte Nachricht wird mit Hilfe eines bestimmten Verfahrens (encodingStyle) kodiert (wie in SOAP).
- Google verwendet Kodierungsverfahren von SOAP, z.B. SOAP-Arrays für Suchresultate.

© Klaus Schild, 2004

48

soap:header

```
<operation name="doGoogleSearch">
  <soap:operation soapAction="urn:GoogleSearchAction"/>
  <input>
    <soap:body parts="q start" use="encoded" .../>
    <soap:header parts="key maxResults filter restrict ..."
      use="literal"/>
  </input>
  <output>...</output>
</operation>
```

input-Nachricht wird auf SOAP-Header und -Body verteilt.

- Teile der abstrakten Nachricht werden im SOAP-Briefkopf repräsentiert.
- Struktur von soap:header analog zu soap:body.

soap:address

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
  <soap:binding xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  ...
</binding>
<port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">
  <soap:address location="http://api.google.com/search/beta2"/>
</port>
```

- Jeder Kommunikationsendpunkt muss genau eine Web-Adresse (soap:address) haben.
- Beachte: Die Web-Adresse muss zum Transportprotokoll des SOAP-Bindings passen.

HTTP-Binding

SOAP über HTTP

- Bindung für SOAP über HTTP haben wir bereits gesehen:

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
  <soap:binding xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  ...
</binding>
```

- hiermit wird SOAP über HTTP-POST-Schnittstelle realisiert
- es gibt aber auch Möglichkeit, eine HTTP-Schnittstelle zu definieren, die *nicht* SOAP verwendet

HTTP-GET-Anfrage

- HTTP-GET-Anfrage kodiert alle Parameter des RPCs in der URL:

```
GET /search/beta2/doGoogleSearch?key=45675353&q=Anfrage&...
HTTP/1.1
```

Host: api.google.com

Content-Type: text/xml; charset="utf-8"

Content-Length: nnnn

Antwort als HTML-Dokument

HTTP-GET-Binding ohne SOAP

```
<service name="GoogleSearchService">
  <port name="GoogleSearchPort2"
    binding="typens:GoogleSearchBinding2">
    <http:address xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
      location="http://api.google.com/search/beta2"/>
  </service>
<binding name="GoogleSearchBinding2" type="typens:GoogleSearchPort">
  <http:binding verb="GET"/>
  <operation name="doGoogleSearch">
    <http:operation location="doGoogleSearch"/>
    <input><http:urlEncoded/></input>
    <output><mime:content part="docs" type="text/html"/></output>
  </operation>
</binding>
```

relative Adresse

Browser-basiertes Google mit WSDL beschrieben



Bewertung



Vorteile

- + plattformunabhängiger XML-Standard
- + etabliert
- + Syntax der Schnittstelle kann genau festgelegt werden.
- + Unterschiedliche Realisierungen einer abstrakter Schnittstelle möglich
z.B. SOAP über HTTP und SMTP



Nachteile

- verschiedene Protokoll-Bindungen (wie HTTP vs. SMTP) können unterschiedliche Semantik haben
- *keine* explizite Unterscheidung zwischen synchron/asynchron
- *keine* komplexen Interaktionsmuster
- *keine* qualitativen Aspekten (*quality of service*)
- *keine* Sicherheitsaspekte



Wie geht es weiter?

- ☑ Prinzipieller Aufbau von WSDL-Beschreibungen
- ☑ Protokoll-Bindungen in WSDL
- ☑ Vor- und Nachteile von WSDL
- ☑ Google-WSDL lesen und erweitern können

Heute 16:15

- Tutorium

Nächste Woche

- Wie wird WSDL verwendet?
- Anforderungen aus der Praxis



Anhang



soap:operation

```

<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
  Erweiterungselement soap:binding
  <operation name="doGoogleSearch">
    Erweiterungselement soap:operation
    <input>
      Erweiterungselemente soap:body und soap:header
    </input>
    <output>
      Erweiterungselement soap:body und soap:header
    </output>
    <fault>
      Erweiterungselement soap:fault
    </fault>
  </operation>
</binding>

```

soap:operation



```
<operation name="doGoogleSearch">
  <soap:operation soapAction="urn:GoogleSearchAction"/>
  <input>...</input>
  <output>...</output>
</operation>
```

- soapAction: legt für Operation den SOAPAction-Briefkopf in der HTTP-Anfrage fest
- style: SOAP-Standard-Stil kann für die Operation überschrieben werden.