

Organisatorisches

→ »Aktuelles« auf der Webseite der Veranstaltung

Was erwarte ich?

- Sie haben Folien der Vorlesung und Übungen verstanden und können dieses Wissen anwenden.

Konzepte und Technologien

- beherrschen Sie *aktiv*

Beispiel

- gegeben: XSLT-Stylesheet + XML-Dokument
- gesucht: Ergebnis-Dokument

Beispiel

- Konzept Erweiterung von SOAP-Nachrichten erläutern

Syntax

- beherrschen Sie *passiv* und soweit *aktiv*, dass Sie *kleine Änderungen* vornehmen können

Beispiel

- gegeben: DTD + XML-Schema
- Frage: äquivalent?

Beispiel

- gegeben: WSDL-Beschreibung
- Aufgabe: Bindung SOAP/SMTP hinzufügen

WSDL

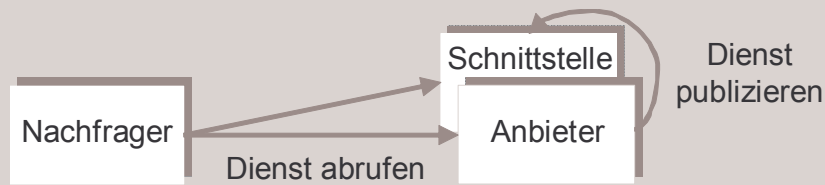
Heutige Vorlesung

WSDL

- Prinzipieller Aufbau
 - abstrakte Schnittstelle
 - Bindungen
- standardisierte Bindungen
 - SOAP über HTTP
 - HTML über HTTP
- Bewertung

Lernziel: Google-
WSDL lesen und
erweitern können

Wozu WSDL?

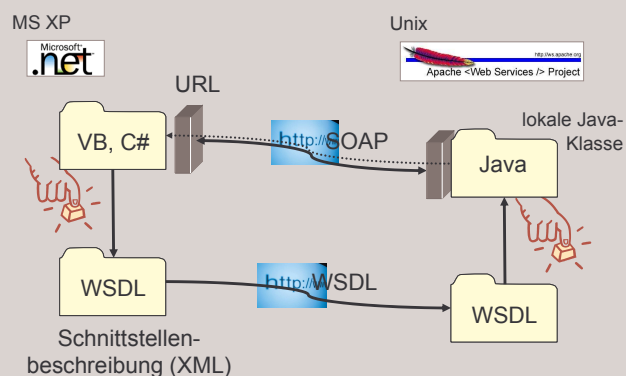


- Client möchte bestimmten Web Service nutzen
- Client benötigt hierfür Beschreibung der Schnittstelle:
 - Struktur des Aufrufes: Name, Eingangsparameter, Ergebnis
 - Übertragungsprotokoll und Web-Adresse
- genau dies kann mit WSDL beschrieben werden
- ähnlich wie Java-IDL, jedoch wesentlich allgemeiner und kryptischer

Wie wird WSDL verwendet?



- aus WSDL-Beschreibung können automatisch Stubs generiert werden
- Stubs abstrahieren von SOAP und Übertragungsprotokoll



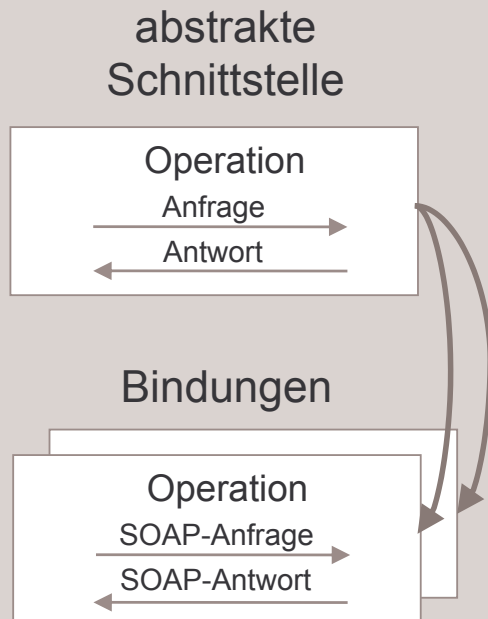
- erleichtert Einbindung des Dienstes zur Entwicklungszeit erheblich
- theoretisch damit auch dynamische Einbindung zur Laufzeit möglich

Prinzipieller Aufbau

Web Services Description Language



- XML-Sprache
- beschreibt Netzwerkdienste als Kommunikationsendpunkte (*Ports*), die bestimmte Nachrichten austauschen
- aktuelle Version 1.1 (2001)
- W3C-Note (©Ariba, IBM und Microsoft)



abstrakte Schnittstelle

- Beschreibung der Schnittstelle unabhängig von
 - Nachrichtenformaten wie SOAP
 - Übertragungsprotokollen wie HTTP

Bindungen

- Realisierung der abstrakten Schnittstelle mit unterschiedlichen Nachrichtenformaten und Übertragungsprotokollen

Beispiel Google (fiktiv)

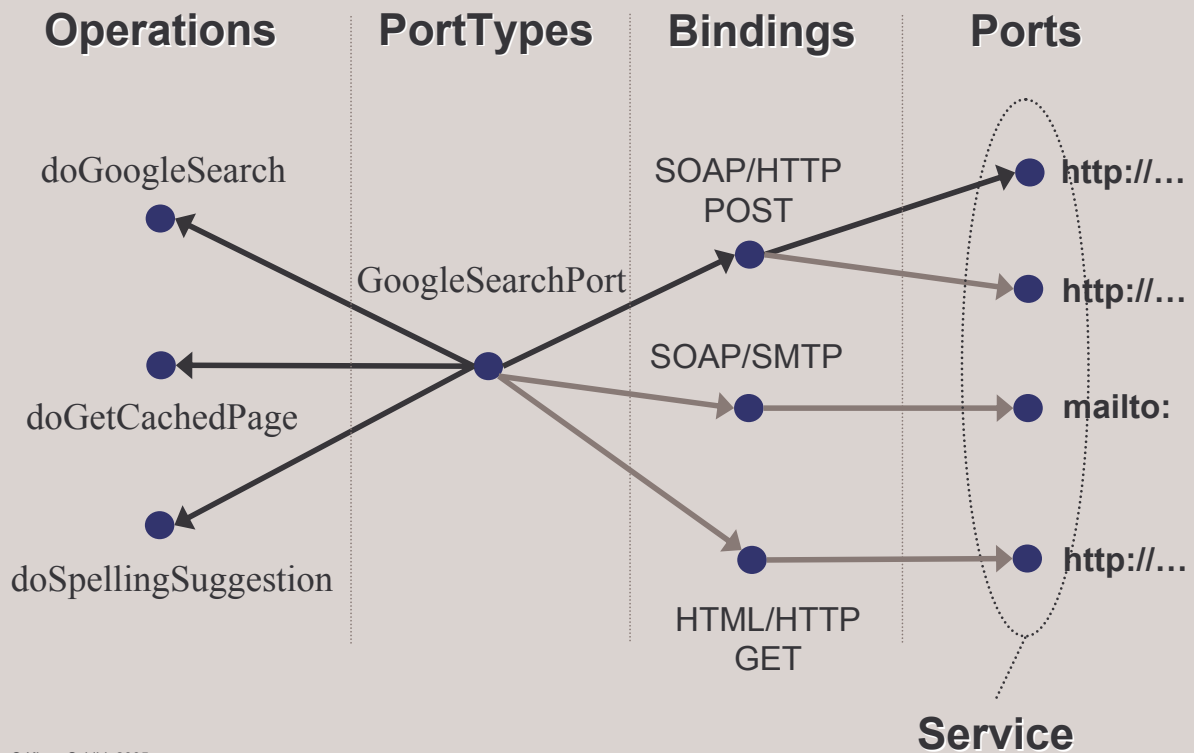
eine **abstrakte Schnittstelle**

- Name der Operation: doGoogleSearch
- Eingangsparameter: key:string, q:string, ...
- Rückgabewert: doGoogleSearchResponse:complexType

drei unterschiedlich **Bindungen**

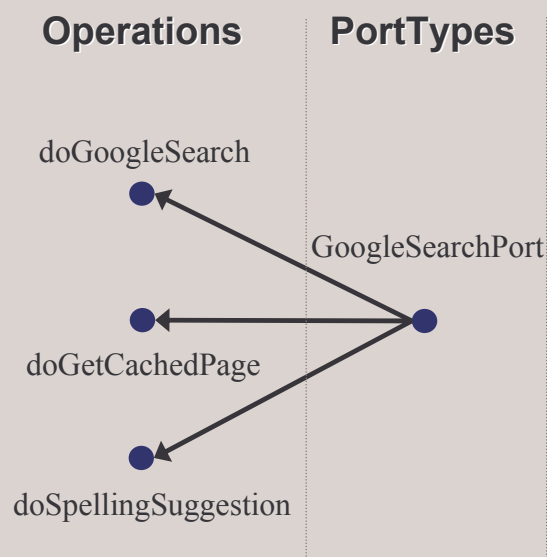
- synchrone SOAP-Schnittstelle: SOAP/HTTP-POST
- asynchrone SOAP-Schnittstelle: SOAP/SMTP
- klassische Browser-Schnittstelle: HTML/HTTP-GET

Grundstruktur



11

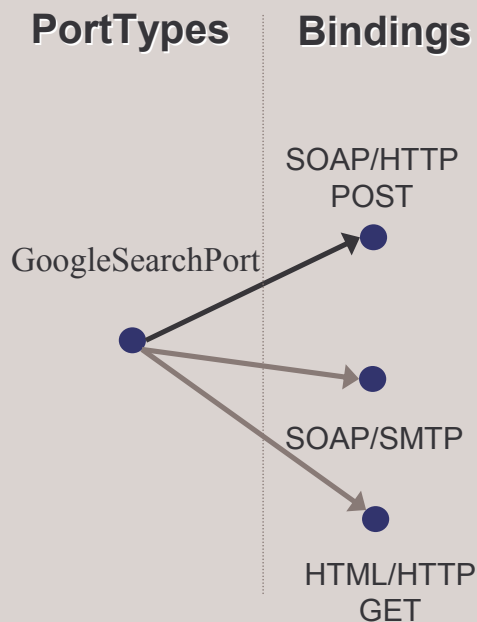
Abstrakte Schnittstellen



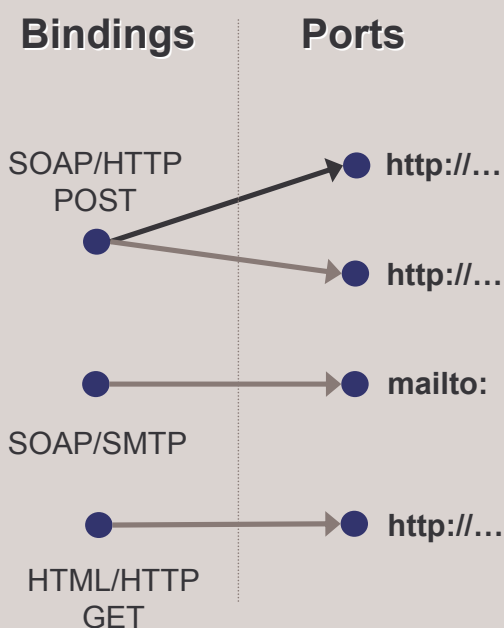
- in WSDL **portType** genannt
- in nächster WSDL-Version **interface** genannt
- Menge von abstrakten Operationen
- jede abstrakte Operation beschreibt Eingangs- und Ausgangsnachricht
- meist nur ein portType, aber auch mehrere möglich

© Klaus Schild, 2005

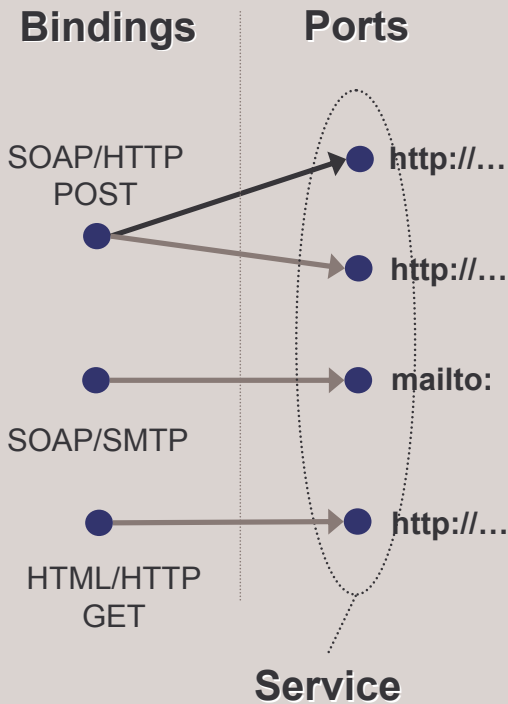
12



- in WSDL **binding** genannt
- für jede abstrakte Schnittstelle (portType) mindestens eine Bindung
- Bindung = Realisierung mit konkretem Nachrichtenformat und Übertragungsprotokoll
- ein portType kann also mit unterschiedlichen Bindungen realisiert sein

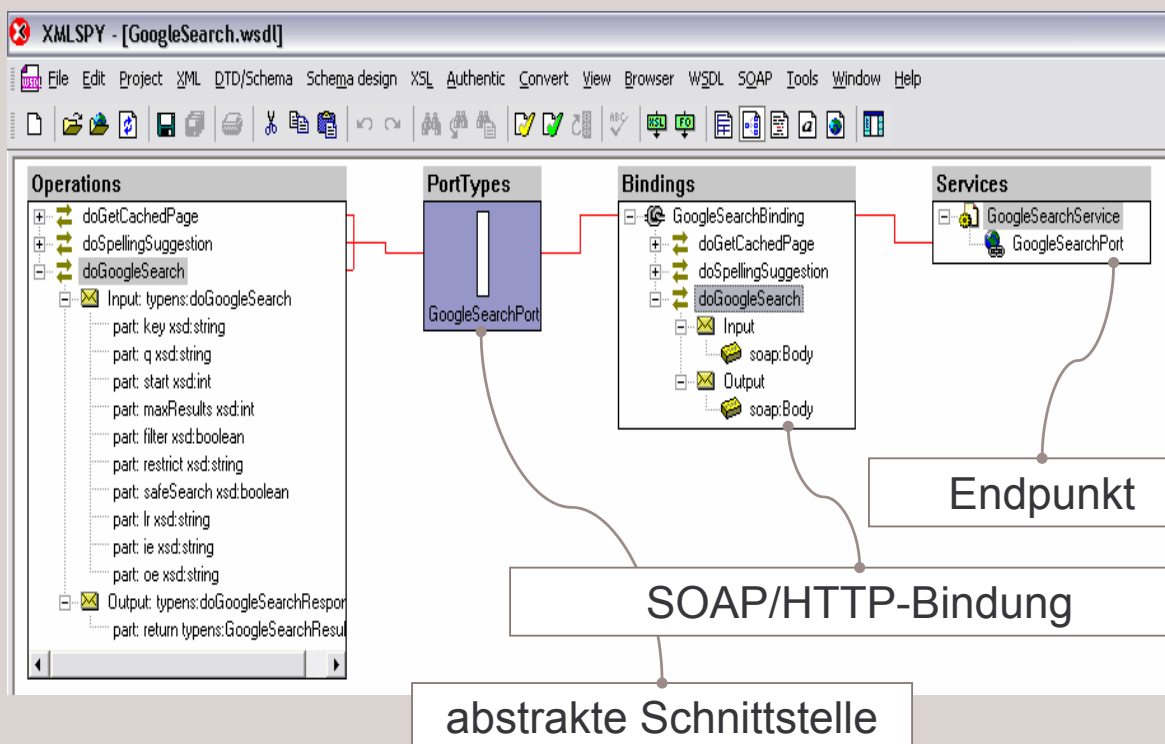


- in WSDL **port** genannt
- in nächster WSDL-Version **endpoint** genannt
- port = Bindung + Web-Adresse
- für jede Bindung (binding) mindestens ein port
- ein binding kann also über unterschiedliche Web-Adressen zugänglich sein



- Menge von ports bilden zusammen einen **Service**
- ports können auch in mehrere Services gruppiert werden

Die WSDL-Beschreibung von Google™



```
<?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>
```

```
<?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 Datentypen
- können für Spezifikation von abstrakten Nachrichten verwendet werden

message

- abstrakte Nachricht
- werden für die Spezifikation der abstrakten Schnittstelle verwendet

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

- Wurzel-Element definitions aus entsprechendem Namensraum
- WSDL-Beschreibung kann Namen haben.
- WSDL-Beschreibung kann eigenen Ziel-Namensraum definieren.
- SOAP-Nachricht kann auf diesen Ziel-Namensraum verweisen.

```
<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>
```

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

- Datentypen für die Spezifikation von abstrakten Nachrichten
- Verwendung von XML-Schema empfohlen, theoretisch 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>
```

- vollständiges XML-Schema
- Ziel-Namensraum normalerweise identisch mit Ziel-Namensraum 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>
```

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

- definiert abstrakte Nachricht mit eindeutigen Namen
- Definitionen werden in portType und binding verwendet
- setzen sich aus logischen Bestandteilen (**parts**) zusammen
- ein part kann z.B. ein Parameter eines RPCs sein
- jedes dieser Bestandteile hat ebenfalls einen eindeutigen Namen
- Reihenfolge der logischen Bestandteile unerheblich

zwei unterschiedliche Modellierungen

1. ein 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 parts

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

zwei unterschiedliche Modellierungen

1. ein part, diesem wird kom

```
<message name="doGoogleS
  <part name="return" type
</message>
```

typens:complexType könnt

2. mehre parts

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

Unterschiede

- parts
reihenfolgeunabhängig
- parts können in Bindung
unterschiedlich behandelt
werden, z.B.:
ein part in Body der SOAP-
Nachricht, das andere part
in den Header

```
<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>
```

portType

```
<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

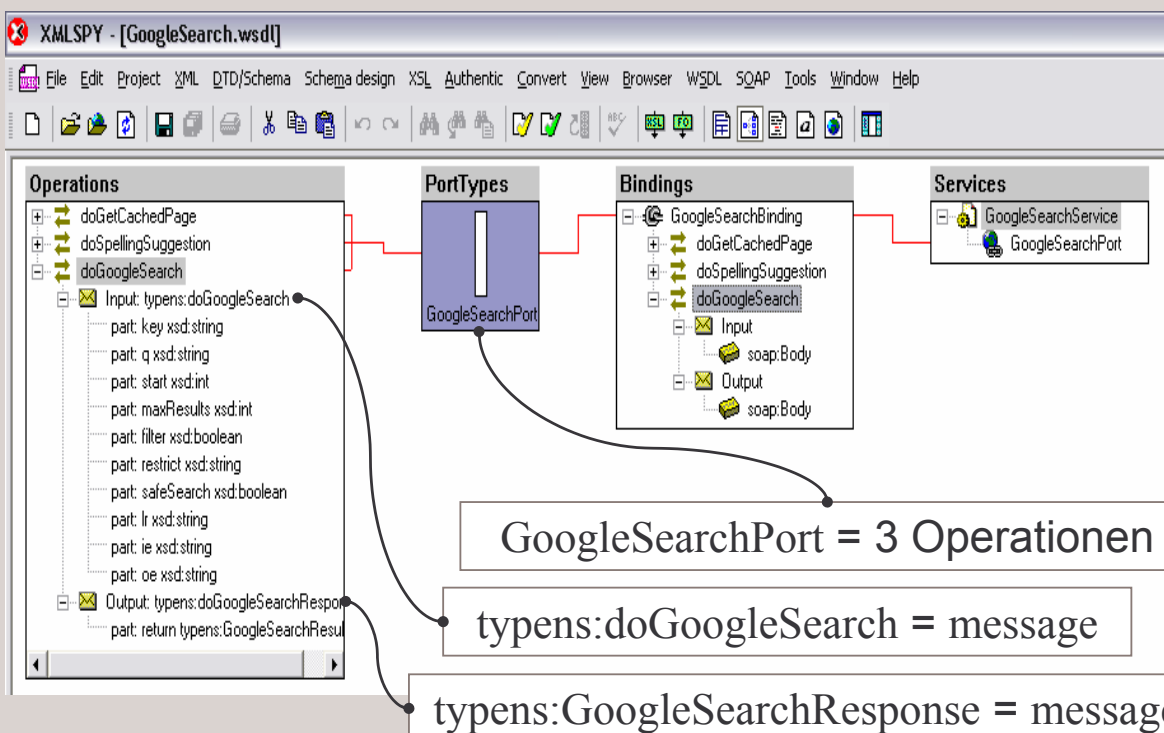
Operation



```
<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 einfaches Interaktionsmuster mit Eingangs- und Ausgangs-Nachrichten.
- Definition wird in binding verwendet
- verwendet keine Datentypen, sondern Nachrichten

Beispiel Google



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

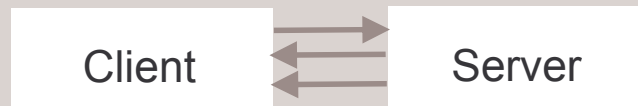


- Anfrage-Antwort-Muster müssen nicht mit *einer* Netzwerkkommunikation (z.B. mit HTTP) realisiert werden.
- auch Realisierung mit zwei unabhängigen Kommunikationen (z.B. E-Mails) möglich
- entfernter Prozeduraufruf daher auch mit SMTP realisierbar
- wird erst in der Bindung (*binding*) festgelegt

Komplexe Interaktionsmuster



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



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

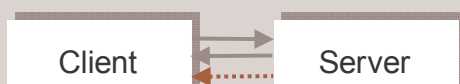
In WSDL *nicht* erlaubt!

Fehlermeldungen



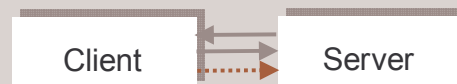
Anfrage-Antwort

```
<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 gemeldet werden
- nur eine Art von Fehlermeldung:
kann aber komplexen Datentyp mit unterschiedlichen Fehlertypen (xsd:choice) haben.

```
<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>
```

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
```

Erweiterungselement

```
<operation name="doGoogleSearch">
```

Erweiterungselement

```
<input>
```

Erweiterungselement

```
</input>
```

```
<output>
```

Erweiterungselement

```
</output>
```

```
</operation>
```

...

```
</binding>
```

- definiert eine Bindung
- wird in Spezifikation eines service verwendet
- **name**: eindeutiger Name der Bindung
- **type**: die abgebildete abstrakte Schnittstelle (portType)
- mehrere binding-Elemente für eine abstrakte Schnittstelle erlaubt

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
```

Erweiterungselement

```
<operation name="doGoogleSearch">
```

Erweiterungselement

```
<input>
```

Erweiterungselement

```
</input>
```

```
<output>
```

Erweiterungselement

```
</output>
```

```
</operation>
```

...

```
</binding>
```

- eigentliche Bindung mit sog. **Erweiterungselementen** (*extensibility elements*) kodiert
- Informationen über Bindung auf allen Ebenen:
 - Bindung allgemein
 - einzelnen Operationen
 - Input- und Output-Nachrichten
 - Fehlermeldungen

- Platzhalter in der WSDL-Grammatik für spezielle Bindungen
- WSDL 1.1 standardisiert zwei Bindungen:
 - SOAP über HTTP
 - HTML über HTTP

werden gleich vorgestellt,
vorher noch letzten Teil
einer WSDL-Beschreibung

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>
```

Service



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

- **Service** = Menge von Ports
- **Port** = Bindung zusammen mit Web-Adresse
- Ports eines Service semantisch äquivalente Alternativen einer abstrakten Schnittstelle (PortType)



Bindung SOAP über HTTP

SOAP-Bindung

```
<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>
```

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

```
<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>
```

```
<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**: gibt an, dass abstrakte Schnittstelle (portType) mit SOAP realisiert ist
- **transport**: Übertragungsprotokoll
- **Beachte**: HTTP meint hier HTTP-POST
- hier auch möglich: transport="http://.../soap/smtp"
- **style**: entfernter Prozeduraufruf (**rpc**) oder Messaging (**document**)

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 lediglich Struktur des SOAP-Nachrichteninhalts (Body) fest, darüber hinaus *keine* Bedeutung

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>
```

Erweiterungselement soap:body und soap:header

```
</output>
```

```
<fault>
```

Erweiterungselement soap:fault

```
</fault>
```

```
</operation>
```

```
</binding>
```

```
<operation name="doGoogleSearch">
  ...
  <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

```
<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
- Google verwendet Kodierungsverfahren von SOAP, benutzt z.B. SOAP-Arrays

soap:header



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

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

- Teile der abstrakten Nachricht im Header Block der SOAP-Nachricht
- für jeden Header Block ein soap:header-Element
- 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>
```

- Jedem Port muss genau eine Web-Adresse (soap:address) zugeordnet sein.
- Beachte: Die Web-Adresse muss zum Transportprotokoll der Bindung passen.

Bindung HTML über HTTP

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 realisiert
- auch möglich, eine HTTP-Schnittstelle zu definieren, die *nicht* SOAP verwendet

Beispiel



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

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

```
Host: api.google.com
```

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

```
Content-Length: nnnn
```

Antwort soll HTML-Dokument sein

HTML über HTTP-GET



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

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">  
  <http:binding verb="GET"/>  
  <operation name="doGoogleSearch">  
    <http:operation location="doGoogleSearch"/> (relative Adresse)  
    <input><http:urlEncoded/></input>  
    <output><mime:content part="docs" type="text/html"/></output>  
  </operation>  
</bindi
```

⇒ browser-basiertes Google mit WSDL beschrieben!

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

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">  
  <http:binding verb="GET"/>  
  <operation name="doGoogleSearch">  
    <soap:operation soapAction="doGoogleSearch"/>  
    <input><http:urlEncoded/></input>  
    <output><soap:body use="literal"/></output>  
  </operation>  
</binding>
```

⇒ REST-kompatible Schnittstelle SOAP/HTTP-GET

Unterschied beachten!

SOAP/HTTP-POST

- **soap:binding**
- **transport="http://schemas.xmlsoap.org/soap/http"**
- **<input><soap:body use="literal"/></input>**
- **<output><soap:body use="literal"/></output>**

SOAP/HTTP-GET

- **http:binding**
- **verb="GET"**
- **<input><http:urlEncoded/></input>**
- **<output><soap:body use="literal"/></output>**

Bewertung

Vor- und Nachteile

-
- + plattformunabhängiger XML-Standard
 - + allgemein akzeptiert und etabliert
 - + Syntax der Schnittstelle kann genau festgelegt werden.
 - + Unterschiedliche Realisierungen einer abstrakter Schnittstelle möglich
z.B. SOAP über HTTP und SMTP
 - verschiedene Protokoll-Bindungen (wie HTTP vs. SMTP) können unterschiedliche Semantik haben
 - *keine* komplexen Interaktionsmuster
 - *keine* qualitativen Aspekten (*quality of service*)
 - *keine* Sicherheitsaspekte

WSDL

- ☑ Prinzipieller Aufbau
- ☑ standardisierte Bindungen
- ☑ Bewertung

heutige Übung

- Google-WSDL erweitern (keine Folien hierzu)
- Gelegenheit, Fragen zu stellen!

Vorlesung nächste Woche

- Web Services in der Praxis
- Ausblick: Semantic Web