

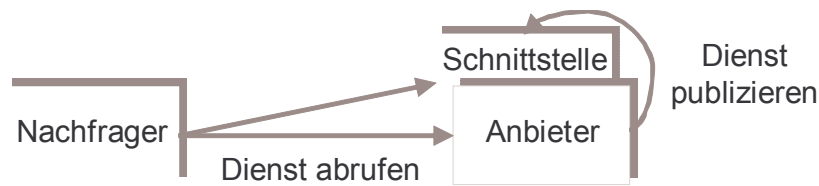
WSDL

Heutige Vorlesung

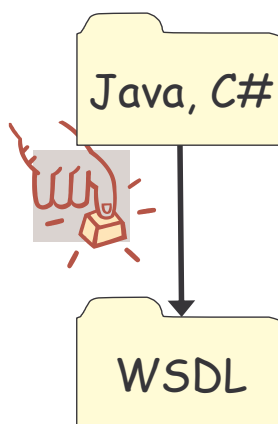
WSDL

- Wozu WSDL-Syntax verstehen?
- prinzipieller Aufbau
 - abstrakte Schnittstelle
 - Bindungen
- standardisierte Bindungen
 - SOAP-Bindung
 - HTTP-Bindung
- Vor- und Nachteile

Lernziel:
Google-WSDL
verstehen und
erweitern
können



- Client möchte bestimmten Web Service nutzen
- Client benötigt hierfür:
 - Struktur des Aufrufes: Name, Parameter, Ergebnis, Fehlermeldungen
 - Übertragungsprotokoll und Web-Adresse
- genau dies wird mit WSDL beschrieben
- ähnlich wie Java-IDL, jedoch wesentlich allgemeiner



- WSDL = zu veröffentlichende Schnittstellenbeschreibung (Vertrag)
- Nutzer des Web Service kennt nur WSDL, nicht Programm-Code
- ⇒ Web-Service-Anbieter sollte WSDL (Vertrag) verstehen!
- mögliche Probleme bei generierten WSDLs:
 - Fehlermeldungen nicht korrekt beschrieben
 - RPC-Parameter ungünstig beschrieben

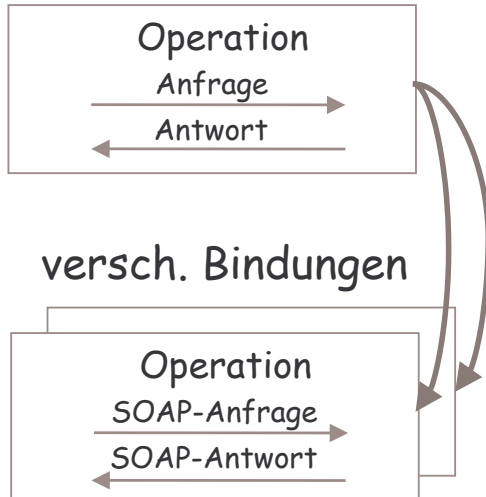
Prinzipieller Aufbau

Web Services Description Language



- beschreibt Netzwerkdienste als Kommunikationsendpunkte (Ports), die best. Nachrichten über best. Protokolle austauschen
- aktuelle Version 1.1 (2001)
- W3C-Note
- Version 2.0 Candidate Recommendation (2006)

abstrakte Schnittstelle



abstrakte Schnittstelle

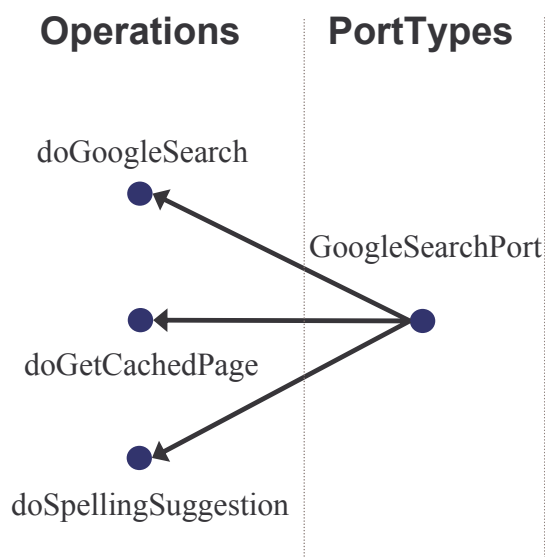
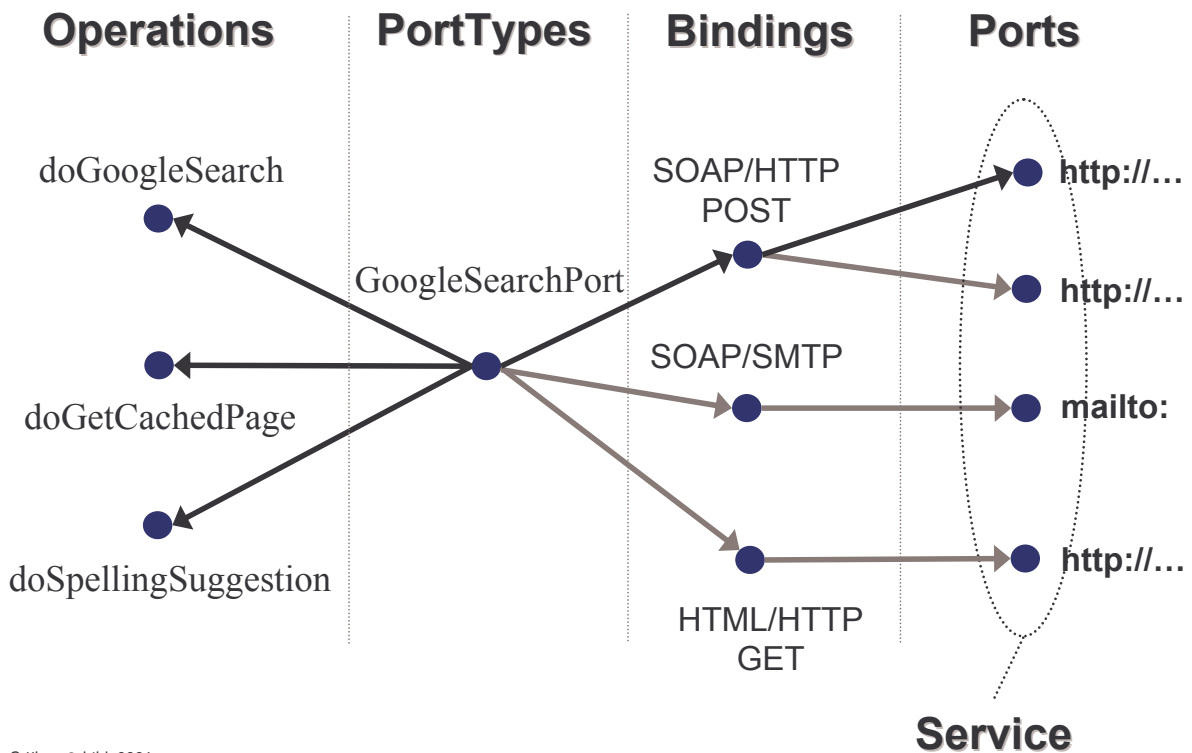
- Beschreibung der Schnittstelle unabhängig von
 - Nachrichtenformaten wie SOAP
 - Übertragungsprotokollen wie HTTP

Bindung

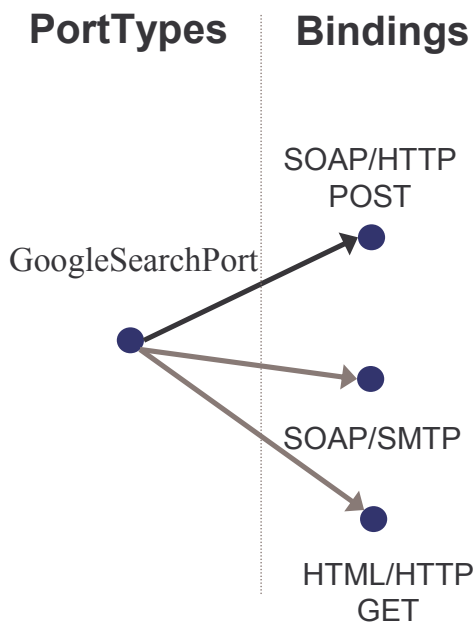
- Realisierung einer abstrakten Schnittstelle mit best. Nachrichtenformat und Übertragungsprotokoll

Beispiel Google (fiktiv)

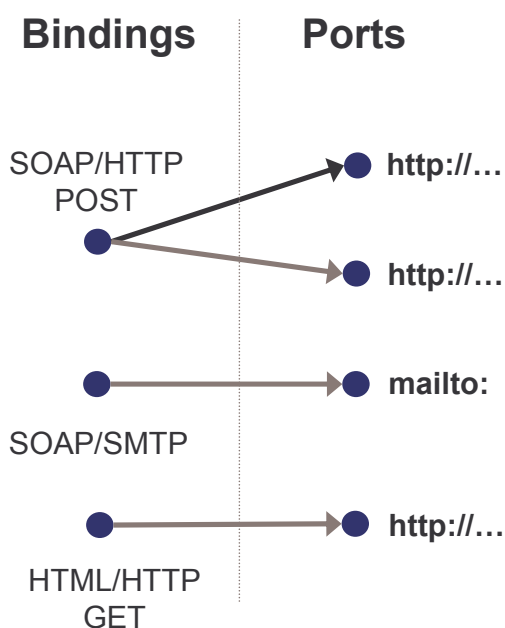
- ein Dienst (**abstrakte Schnittstelle**):
 - Name der Operation: doGoogleSearch
 - Eingangsparameter: key:string, q:string, ...
 - Rückgabewert: doGoogleSearchResponse (komplexer Datentyp)
- eine Beschreibung (**WSDL**), aber vier Zugriffsmöglichkeiten (**Bindungen**):
 1. SOAP/HTTP-POST
 2. SOAP/HTTP-GET (Rest)
 3. SOAP/SMTP (asynchron)
 4. HTML/HTTP-GET (Browser)



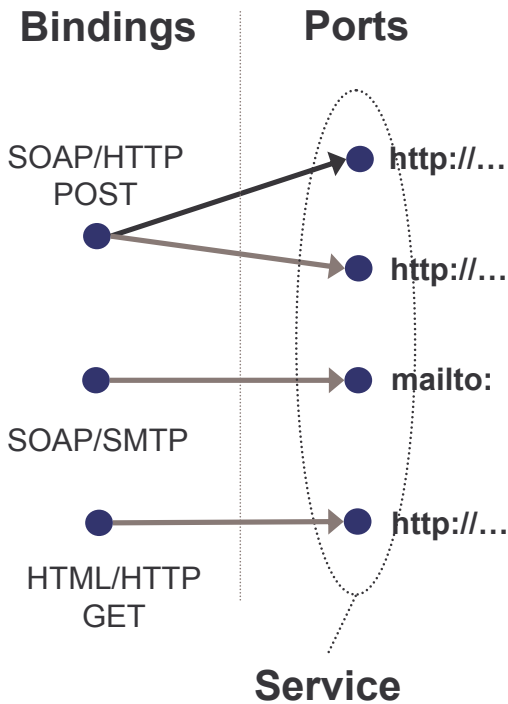
- in WSDL 1.1 portType genannt
- in WSDL 2.0 interface genannt
- portType = Menge von abstrakten Operationen
- jede abstrakte Operation beschreibt Eingangs- und Ausgangsnachricht
- meist nur ein portType, aber in WSDL 1.1 auch mehrere möglich



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

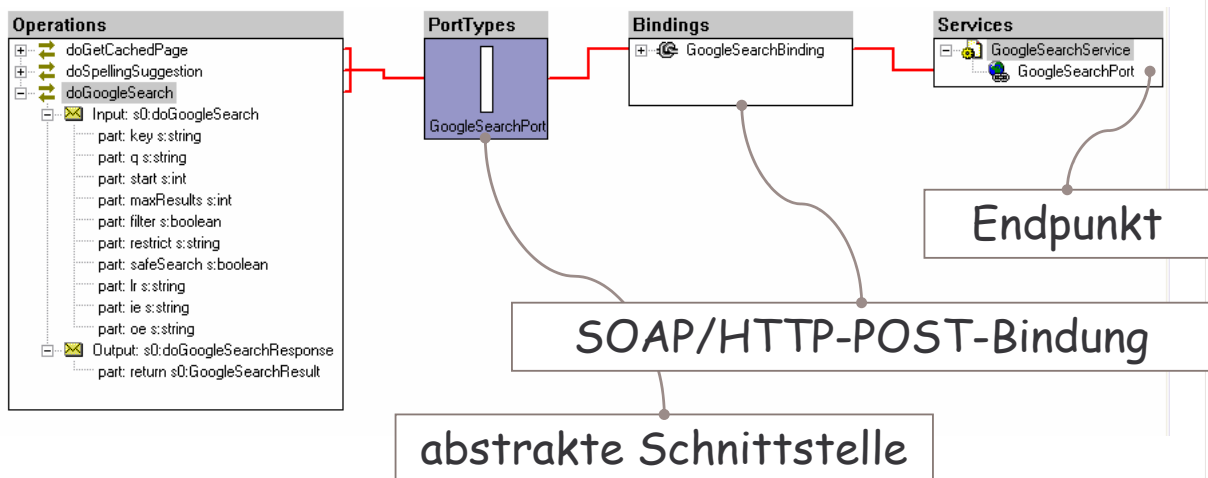


- in WSDL 1.1 **port** genannt
- in WSDL 2.0 **endpoint** genannt
- **port** = Bindung + Web-Adresse
- für jede Bindung (binding) mindestens ein port
- ein binding kann also über unterschiedliche Web-Adressen zugänglich sein



- Menge von ports bilden zusammen einen **Service**
- ports können auch in verschiedene Services gruppiert werden
- ports eines Service = semantisch äquivalente Alternativen

Die WSDL-Beschreibung von Google™

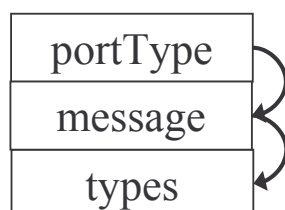


```
<?xml version="1.0"?>
<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  ....
</definitions>
```

- Wurzel-Element definitions aus entsprechendem Namensraum
 - Namensraum von definitions = Version
 - WSDL-Beschreibung kann Ziel-Namensraum definieren.
- ⇒ SOAP-Nachricht kann auf diesen Ziel-Namensraum verweisen.

Hilfsdefinitionen: types & message

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



types

- Definition von Datentypen
- werden für Spezifikation von abstrakten Nachrichten verwendet

message

- Definition einer abstrakten Nachricht
- werden für Spezifikation der abstrakten Schnittstelle verwendet

```
<types>
```

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

```
...
```

```
</schema>
```

```
</types>
```

- Datentypen für Spezifikation von abstrakten Nachrichten
- XML-Schema als Typsystem empfohlen, theoretisch jedes andere Typsystem aber auch erlaubt
- Beachte: XML-Schema kann auch verwendet werden, wenn Nachrichten nicht in XML übertragen werden.

Beispiel: Google™-Suchresultat

```
<types>
```

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

```
<xsd:complexType name="GoogleSearchResult">
```

```
<xsd:all>
```

```
<xsd:element name="estimatedTotalResultsCount" type="xsd:int"/>
```

```
<xsd:element name="resultElements" type="tns: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>
  ...
</definitions>
```

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

- Name muss innerhalb der WSDL eindeutig sein
- setzen sich aus logischen Bestandteilen zusammen:
≥1 parts
- part kann z.B. ein Parameter eines RPCs sein
- jedes part hat eindeutigen Namen
- Reihenfolge der logischen Bestandteile unerheblich

zwei unterschiedliche Modellierungen

1. mehrere parts:

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

2. ein part mit komplexen Datentyp:

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

tns:complexType könnte z.B. 2 Parameter enthalten

zwei unterschiedliche Modellierungen

1. mehrere parts:

```
<message name="doGoogleSea
  <part name="param1" eleme
  <part name="param2" eleme
</message>
```

2. ein part mit komplexen Da

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

tns:complexType könnte z.

Unterschiede

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

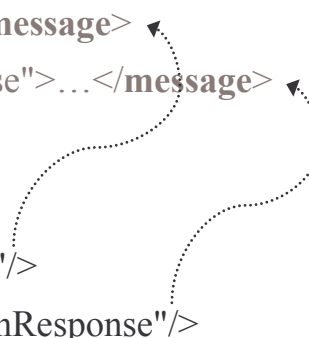
```
<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  ...
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>...</types>
  <message name="doGoogleSearch">...</message>
  <message name="doGoogleSearchResponse">...</message>
  <portType name="GoogleSearchPort">...</portType>
  ...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

Abstrakte Schnittstelle: portType

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

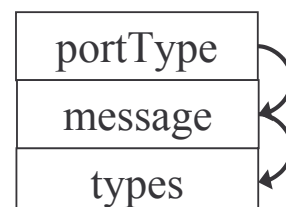

- abstrakte Schnittstelle = Menge von abstrakten Operationen (operations)

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



- definiert einfaches Interaktionsmuster mit Eingangs- und Ausgangs-Nachrichten.
- wichtig: verwendet keine Datentypen, sondern abstrakte Nachrichten

```
<message name="doGoogleSearchResponse">
  <part name="return" type="tns:GoogleSearchResult"/>
</message>
...
<portType>
  <operation name="doGoogleSearch">
    <input message="tns:doGoogleSearch"/>
    <output message="tns:doGoogleSearchResponse"/>
  </operation>
  ...
</portType>
```



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

Einweg (one way)

```
<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 mit zwei unabhängigen Kommunikationen (z.B. E-Mails) möglich
- Realisierung wird erst in der Bindung (binding) festgelegt

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



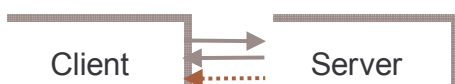
```
<operation name="...">  
  <input message="..."/>  
  <output message="..."/>  
  <output message="..."/>  
</operation>
```

In WSDL
nicht erlaubt!

Fehlermeldungen

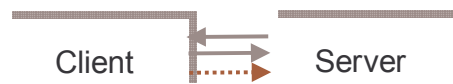
Anfrage-Antwort

```
<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

```
<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="tns:GoogleSearchPort">
  ...
  </binding>
  <binding ...>...</binding>
  ...
</definitions>
```

Grundstruktur von binding

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

Erweiterungselement

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

Erweiterungselement

```
<input>
```

Erweiterungselement

```
</input>
```

```
<output>
```

Erweiterungselement

```
</output>
```

```
</operation>
```

...

```
</binding>
```

- definiert eine Bindung
- **name**: eindeutiger Name der Bindung
- **type**: die zu realisierende abstrakte Schnittstelle (portType)
- mehrere binding-Elemente für eine abstrakte Schnittstelle erlaubt

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

Erweiterungselement

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

Erweiterungselement

```
<input>
```

Erweiterungselement

```
</input>
```

```
<output>
```

Erweiterungselement

```
</output>
```

```
</operation>
```

...

```
</binding>
```

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

Erweiterungselemente

- Platzhalter in der WSDL-Grammatik
- WSDL 1.1 standardisiert drei Bindungen:
 1. SOAP
 2. HTTP
 3. MIME
- bevor SOAP- und HTTP-Binding vorgestellt wird, noch den letzten Teil einer WSDL-Beschreibung

```
<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="tns:GoogleSearchPort">
  ...
  </binding>
  <service name="GoogleSearchService">...</service>
</definitions>
```

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

- **Service** = Menge von Ports
- **Port** = Bindung + Web-Adresse
- Ports eines Service sollen semantisch äquivalente Alternativen einer abstrakten Schnittstelle sein



SOAP-Bindung

SOAP-Bindung

```
<binding name="GoogleSearchBinding" type="tns: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 portType → SOAP-Nachricht
- Beachte: WSDL 1.1 benutzt SOAP 1.1

```
<binding name="GoogleSearchBinding" type="tns: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="tns: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 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

Abbildung der abstrakten Nachrichten

```
<binding name="GoogleSearchBinding" type="tns: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>
```

- **soap:body**: Wie wird abstrakte input- bzw. output-Nachricht auf SOAP-Body abgebildet?
- **use="literal"**: abstrakte Nachricht wird unverändert übernommen

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

- **use="encoded"**: Abstrakte Nachricht wird mit Hilfe eines bestimmten Verfahrens (`encodingStyle`) kodiert.
- hier Kodierungsverfahren von SOAP (→ RPC-Struktur, einschl. SOAP-Arrays)

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

```
...
```

```
<input>
```

```
  <soap:body parts="q start maxResults ..." use="encoded" .../>
```

```
  <soap:header message="tns:doGoogleSearch" part="key"
    use="literal"/>
```

```
</input>
```

```
<output>...</output>
```

```
</operation>
```

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

- Teile der abstrakten Nachricht → SOAP-Header
- für jeden Header Block ein soap:header-Element
- Struktur von soap:header analog zu soap:body.

soap:address

```
<binding name="GoogleSearchBinding" type="tns: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="tns:GoogleSearchBinding">
```

```
  <soap:address location="http://api.google.com/search/beta2"/>
```

```
</port>
```

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

HTTP-Bindung

Beispiel für HTTP-Bindung (fiktiv)

- HTTP-GET-Anfrage kodiert alle Parameter in 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

```
<binding name="GoogleSearchBinding" type="tns:GoogleSearchPort"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/">
  <http:binding verb="GET"/>
  <operation name="doGoogleSearch">
    ...
    <input><http:urlEncoded/></input>
    <output><mime:contentType="text/html"/></output>
  </operation>
</binding>

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

⇒ browser-basiertes Google mit WSDL beschrieben!

```
<binding name="GoogleSearchBinding" type="tns:GoogleSearchPort">
  <http:binding verb="GET"/>
  <operation name="doGoogleSearch">
    ...
    <input><http:urlEncoded/></input>
    <output><mime:mimeXml/></output>
  </operation>
</binding>
```

- + plattformunabhängig
- + 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

Wie geht es weiter?

WSDL

- ☑ Prinzipieller Aufbau
- ☑ SOAP- und HTTP-Bindungen
- ☑ Vor- und Nachteile

heutige Übung

- SOAP
- Hinweise zur Klausur

Vorlesung nächste Woche

- Web Services in der Praxis