

## Web-Services

Motivation

Web Services Description Language (WSDL)

Hypertext Transfer Protocol (HTTP)

SOAP

Representational State Transfer (REST)

Zusammenfassung



- Ziel: Universeller Zugriff auf Cloud-Dienste durch
  - Endnutzer einer Anwendung
  - Administrator des Cloud-Diensts
  - Andere (Cloud-)Anwendungen
- Web-Services (mögliche Definition)

„A **Web service** is a software system designed to support **interoperable machine-to-machine interaction over a network**. It has an interface described in a machine-processable format (specifically **WSDL**). Other systems interact with the Web service in a manner prescribed by its description using **SOAP messages**, typically conveyed **using HTTP** with an XML serialization in conjunction with other Web-related standards.“

[Web Services Architecture – W3C Working Group Note 11, <http://www.w3.org/TR/ws-arch/>]
- Herausforderungen
  - Woher weiß ein Nutzer, wie er mit einem Dienst kommunizieren soll?
  - Welcher Teil der Kommunikation lässt sich automatisiert implementieren?
  - Wie lässt sich ein Web-Service skalierbar realisieren?



# Web Services Description Language (WSDL)

- Überblick
  - Beschreibungssprache für die Funktionalität von Web-Services
  - Repräsentation als XML-Dokument
  - Ziel: Automatische Erzeugung von Stubs für Zugriff auf Web-Services
- Bestandteile einer WSDL-Beschreibung (*Description*)
  - Datentypen (*Types*)
  - Schnittstellen (*Interfaces*)
  - Abbildung auf Kommunikationsprotokolle (*Bindings*)
  - Dienste (*Services*)

[Hinweis: Zwischen den WSDL-Versionen 1.1 und 2.0 wurden einige Elemente neu benannt. Des Weiteren wurde das message-Element entfernt. Die im Rahmen der Vorlesung verwendeten Begriffe sind der Version 2.0 entnommen.]

## Literatur



**Web Services Description Language (WSDL) Version 2.0**

<http://www.w3.org/TR/wsd120/>



David C. Fallside and Priscilla Walmsley

**XML Schema Part 0: Primer Second Edition, 2004.**



# Datentypen

- Standarddatentypen aus der XML Schema Definition (XSD)

```
<types>
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              targetNamespace="[URI des Typ-Namensraums]">
    [Definitionen dienstspezifischer Datentypen]
  </xsd:schema>
</types>
```

[xmlns: XML namespace]
- Dienstspezifische Datentypen
  - Spezifizierung komplexer Datenstrukturen
  - Beispiel: Zusammengesetzter Datentyp aus Zeichenkette und Double

```
<xsd:element name="[Name des Datentyps]">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="[Variablenname]" type="xsd:string">
      <xsd:element name="[Variablenname]" type="xsd:double">
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```



## Schnittstellen

- Beschreibung der Methoden und Fehlermeldungen

```
<interface name="[Schnittstellename]">
  <operation>[...]</operation>
  <fault [...]/>
  [Definitionen weiterer Operationen und Fehlermeldungen]
</interface>
```

- Methoden

- Festlegung des Kommunikationsmusters (z. B. in-out, in-only,...)
- Zuordnung von Nachrichtenformaten zu Operationen
- Beispiel: Methode mit Anfrage-Antwort-Interaktion

```
<operation name="[Methodenname]"
  pattern="http://www.w3.org/ns/wsdl/in-out">
  <input messageLabel="In" element="[Datentyp der Anfrage]"/>
  <output messageLabel="Out" element="[Datentyp der Antwort]"/>
</operation>
```

- Fehlermeldungen

```
<fault name="[Fehlername]" element="[Datentyp der Fehlermeldung]"/>
```



## Abbildung auf Kommunikationsprotokolle

- Beispiel: Abbildung auf SOAP über HTTP

```
<binding name="[Abbildungsname]"
  interface="tns:[Schnittstellename]"
  type="http://www.w3.org/ns/wsdl/soap"
  wsoap:protocol="http://[...]/soap/bindings/HTTP/">

  <operation ref="tns:[Methodenname]"
    wsoap:mep="http://[...]/soap/mep/request-response"/>
  [Auflistung weiterer Operationen]

  <fault ref="tns:[Fehlername]" wsoap:code="[SOAP-Fehler-Code]"/>
  [Auflistung weiterer Fehlermeldungen]
</binding>
```

- Festlegung des Kommunikations- und Transportprotokolls
- Angabe des Kommunikationsmusters für Methoden [mep: message exchange pattern]
- Abbildung der Fehlermeldungen

- Weiteres Beispiel: Verwendung von HTTP als Anwendungsprotokoll



## Dienste

- Beschreibung der Kommunikationsendpunkte des Web-Service

```
<service name="[Web-Service-Name]"
  interface="tns:[Schnittstellename]">
  <endpoint name="[Endpunktname]"
    binding="tns:[Abbildungsname]"
    address="[URL des Web-Service]"/>
  [Definitionen weiterer Endpunkte]
</service>
```

- Angabe der Kommunikationsprotokolle der Endpunkte
- Bekanntmachung der zu kontaktierenden Endpunktadressen

- WSDL-Dokumente enthalten also Antworten auf folgende Fragen:

- Welche Methoden bietet der Dienst an?
- An wen muss sich ein Client wenden, um die Methoden zu verwenden?
- Welche Nachrichten muss ein Client hierfür senden?

→ Automatisierte Generierung von Stubs für Web-Services möglich



## Hypertext Transfer Protocol (HTTP)


- Protokoll für Zugriff auf *Ressourcen* über ein Netzwerk

- Zumeist TCP/IP als zuverlässiges Transportprotokoll
- Textbasierter Nachrichtenaustausch

- Aufbau der Anfrage- und Antwortnachrichten

- Header
  - Methodenname und Ressourcen-ID (Anfrage) bzw. Statusmeldung (Antwort)
  - HTTP-Versionsnummer
  - Liste von Schlüssel-Wert-Paaren (z. B. Content-Length: 4711)
- Body (optional): Nutzdaten

- Literatur

 Tim Berners-Lee, Roy Fielding, and Henrik Frystyk  
**Hypertext Transfer Protocol – HTTP/1.0, RFC 1945, 1996.**

 Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter et al.  
**Hypertext Transfer Protocol – HTTP/1.1, RFC 2616, 1999.**



## ■ Überblick über die wichtigsten HTTP-Methoden

Methoden	Beschreibung
GET	Lesender Zugriff auf die adressierte Ressource
HEAD	Unterschied zu GET: Antwort enthält keinen Body
POST	Modifizierender Zugriff auf die adressierte Ressource (Beispiele) <ul style="list-style-type: none"><li>– Übermittlung von Formulardaten</li><li>– Anfügen eines Datensatzes an eine Datenbank</li></ul>
PUT	Registrieren von Daten unter der übergebenen Ressourcenadresse
DELETE	Löschen der adressierten Ressource

[RFC 2616]

## ■ Kategorien von Statusmeldungen

Klasse	Kategorie	Beschreibung
1xx	Informell	Anfrage wurde empfangen, Bearbeitung erfolgt
2xx	Erfolg	Anfrage wurde empfangen, verstanden und akzeptiert
3xx	Weiterleitung	Weitere Aktionen notwendig
4xx	Client-Fehler	Anfrage war fehlerhaft
5xx	Server-Fehler	Anfrage war korrekt, aber im Server lag ein Fehler vor

[RFC 2616]



## ■ Überblick

- Standard-Kommunikationsprotokoll für Web-Services
- Versand von Nachrichten mittels separatem Transportprotokoll
- Hinweis zur Namensgebung
  - Ursprünglich: *Simple Object Access Protocol*
  - Inzwischen nur noch als Abkürzung verwendet

## ■ Bestandteile des SOAP-Nachrichten-Frameworks

- Nachrichtenaufbau
- Verarbeitungsmodell
- Abbildung auf Transportprotokolle
- Erweiterbarkeitsmodell

## ■ Literatur

-  **SOAP Version 1.2**  
<http://www.w3.org/TR/soap12/>



# Nachrichtenaufbau

## ■ Repräsentation als XML-Dokument

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    [Header-Blöcke]
  </soap:Header>
  <soap:Body>
    [Body-Daten]
  </soap:Body>
</soap:Envelope>
```

## ■ Kapselung von Informationen im Wurzelement soap:Envelope

- Header (optional): SOAP-Header-Blöcke mit
  - Kontextinformationen für Nutzdaten
  - Kontrollflussinformationen für Kommunikationspartner
- Body
  - Nutzdaten
  - Fehlermeldungen



# Verarbeitungsmodell

# Grundprinzip

## ■ SOAP-Knoten

- Kategorien
  - Sender (*Initial SOAP sender*)
  - Zwischenstation (*SOAP intermediary*)
  - Empfänger (*Ultimate SOAP receiver*)
- Adressierung über Universal Resource Identifier (URI)
- Weitergabe von Nachrichten entlang eines Pfads aus SOAP-Knoten
  - Zwischenstationen dürfen Header-Blöcke lesen, hinzufügen oder löschen
  - Festlegung des Adressaten eines Header-Blocks mittels role-Attribut
    - Endgültiger Empfänger (*ultimateReceiver*)
    - Nächste Zwischenstation bzw. endgültiger Empfänger (*next*)
    - Kein Adressat (*none*)

## ■ Schritte zur Verarbeitung eines Header-Blocks

- Entfernen des Blocks von der ursprünglichen Nachricht
- Verarbeitung der im Block enthaltenen Informationen gemäß der referenzierten Block-Spezifikation → Semantik ist anwendungsabhängig
- Bei Bedarf: Hinzufügen eines Blocks für die nächste Zwischenstation



## ■ Fault-Element im Body einer SOAP-Nachricht

```
<soap:Body>
  <soap:Fault>
    <soap:Code>[Fehler-Code]</soap:Code>
    <soap:Reason>[Fehlerbeschreibung]</soap:Reason>
    <soap:Detail>[Anwendungsspezifische Informationen]</soap:Detail>
  </soap:Fault>
</soap:Body>
```

## ■ Fehler-Code

```
<soap:Code>
  <soap:Value>[Hinweis auf Fehlerursache]</soap:Value>
  <soap:Subcode>[Fehler-Subcode]</soap:Subcode>
</soap:Code>
```

- Beispiele für Fehlerfälle
  - \* Header-Block wurde nicht verstanden
  - \* Aufbau eines Header-Blocks entspricht nicht den Regeln
- Angabe von untergeordneten Fehler-Codes möglich



## ■ HTTP

- Übertragung der SOAP-Nachrichten im Body von HTTP-Nachrichten
- Eigener Content-Type: application/soap+xml
- Methoden
  - HTTP-GET
    - \* SOAP-Nachricht als Teil der Antwort
    - \* Rein lesende Anfragen
  - HTTP-POST
    - \* Anfrage-Antwort-Interaktion: Implizite Zuordnung der SOAP-Nachrichten
    - \* Modifizierende Anfragen
- Signalisierung von SOAP-Fehlern: 500 Internal Server Error

## ■ SMTP

- Übertragung der SOAP-Nachrichten im E-Mail-Text oder als Anhang
- Explizite Zuordnung der Nachrichten bei Anfrage-Antwort-Interaktion erforderlich, z. B. durch Hinzufügen der Message-Id der Anfrage-Mail im In-reply-to-Feld des E-Mail-Header der Antwort-Mail



## Representational State Transfer (REST)

### ■ Intention

Entwicklung eines Regelwerks für hochskalierbare Systeme zur Bereitstellung von Anwendungen und Diensten über das Internet

„The name „**Representational State Transfer**“ is intended to evoke an image of **how a well-designed Web application behaves**: a network of web pages (a virtual state-machine), where the user progresses through the application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user [...]“

[Fielding, Architectural Styles and the Design of Network-based Software Architectures.]

### ■ Architekturkonzept

- Sammlung von Grundprinzipien, keine konkrete Systemimplementierung
- Netzwerkbasierter Ansatz
- Anwendungsbeispiel: World Wide Web (WWW)

### ■ Literatur

Roy Fielding  
**Architectural Styles and the Design of Network-based Software Architectures**, *Dissertation*, 2000.



## Herangehensweise

### ■ An- und Herausforderungen

- Geografische Verteilung
- Weltweite Skalierbarkeit
- Hohe Anzahl von Dienst Anbietern und -nutzern
- Unabhängiges Hinzufügen und Entfernen von Komponenten
- Heterogene Hardware, Software, Datenformate,...
- Organisationsübergreifende Architektur
- Vollständige Verfügbarkeit des Gesamtsystems nicht gegeben
- Verwaltung unterschiedlicher Informationen

### ■ Formulierung von Grundprinzipien für den Aufbau eines Systems

- Aufteilung in Client und Server
- Zustandslose Interaktion
- Zwischenspeicherung von Antwortnachrichten
- Einheitliche Schnittstellen
- Transparente indirekte Kommunikation
- Code-on-Demand



## Aufteilung in Client und Server

- Trennung der Belange
  - Server
    - Bereitstellung des Diensts
    - Verwaltung der Nutzdaten
  - Client
    - Schnittstelle für Zugriff auf den vom Server angebotenen Dienst
    - Nutzung des Diensts mittels Senden von Anfragen an den Server
- Vorteile
  - Beide Seiten können unabhängig voneinander (von potenziell unterschiedlichen Organisationen) weiterentwickelt werden, solange die Schnittstelle zwischen ihnen stabil bleibt
  - Erhöhte Portabilität in heterogenen Umgebungen
- Nachteile
  - Asymmetrische Aufgaben-/Lastverteilung nicht für alle Dienste geeignet
  - Server-Seite kann bei schlechter Skalierbarkeit zum Flaschenhals werden



## Zustandslose Interaktion

- Kommunikation per Nachrichtenaustausch
  - Verwendung von bidirektionalen Datenströmen
  - Kleine bzw. mittelgroße Nachrichten für Kontrollflussdaten
  - Große Nachrichten für Anwendungsdaten
- Zustandslose Kommunikation
  - Zustandstragende Nachrichten
    - Kein Vorhalten von Kontextinformation auf Server-Seite
    - Zustand der Interaktion wird vollständig in den Nachrichten selbst verwaltet
  - Vorteile
    - Geringerer Ressourcenverbrauch und bessere Skalierbarkeit auf Server-Seite
    - Einfachere parallele Bearbeitung von Anfragen
    - Bessere Unterstützung für indirekte Kommunikation (z. B. mittels Proxies)
  - Nachteile
    - Eventuell Mehraufwand durch vielfaches Senden von Kontextinformationen
    - Server kann nur in begrenztem Umfang konsistentes Verhalten der Anwendung garantieren → Problematisch bei unterschiedlichen Client-Versionen



## Zwischenspeicherung von Antwortnachrichten

- Ziele
  - Entlastung der Server-Seite: Minimierung der Client-Server-Interaktionen
  - Reduzierung der über das Netzwerk zu übertragenden Daten
- Grundprinzip
  - Speichern von Antworten mit Vermerk zu welcher Anfrage sie gehörten
  - Wiederverwendung von Antwortnachrichten bei äquivalenten Anfragen
  - Antwortnachrichten enthalten implizite oder explizite Hinweise, ob sie zwischengespeichert werden können/dürfen oder nicht
- Mögliche Speicherorte
  - Client
  - Server
  - Dedizierte Cache-Server
- Probleme
  - Identifizierung von äquivalenten Anfragen
  - Semantikunterschiede zwischen geteilten und Client-spezifischen Caches



## Einheitliche Schnittstellen

## Bedingungen

- Identifizierung von *Ressourcen*: Universal Resource Identifiers (URIs)
  - Ressource als abstraktes Konzept
    - Referenz ist nicht an die Existenz eines Objekts gebunden
    - Zwei URIs können auf dasselbe zeigen, jedoch nicht dasselbe meinen  
[Beispiel: neuester Sicherungspunkt vs. Sicherungspunkt von letzter Woche]
  - URIs sollten sich so selten wie möglich ändern
    - Unabhängigkeit vom Inhalt einer Ressource
    - Unabhängigkeit vom Ort einer Ressource
- Manipulation von Ressourcen mittels *Repräsentationen*
  - Repräsentation einer Ressource muss nicht notwendigerweise das Format aufweisen, in dem die Ressource selbst verwaltet wird
    - Vorverarbeitung auf Senderseite
    - Anhängen von beschreibenden Metadaten
    - Bereitstellung von Originaldaten und Prozeduren für deren Verarbeitung
  - Modifikation einer Ressource mittels Weitergabe einer entsprechenden Repräsentation mit den gewünschten Änderungen




- Selbstbeschreibende Nachrichten
  - Jede Nachricht enthält Informationen darüber, wie sie zu interpretieren ist
  - Beispiele
    - Angabe eines MIME-Typs (z. B. `text/html`)
    - Hinweis, ob eine Antwort im Cache zwischengespeichert werden darf
  - Vorteil: Unterstützung von Proxies und Caches
  - Nachteil: Umfangreichere Nachrichten
- Zustandstransitionen mittels Hyperlinks
  - Information zum Voranschreiten in den nächsten Anwendungszustand ist Bestandteil der Repräsentation des vorherigen Anwendungszustands
  - Menge der möglichen Folgezustände von der Anwendung bestimmt
  - Zustandsübergang erfolgt nachdem der Nutzer einen der möglichen Folgezustände ausgewählt hat



- Zugriff auf Ressourcen
  - Web-Seiten, Bilder, Videos,...
  - Referenzierung durch URIs
  - Zustandsübergänge per Hyperlinks
- Einsatz von HTTP
  - Jede Nachricht enthält Host-Name & URI
  - Eigene Header-Felder (z. B. `Cache-Control`) für Caching-Unterstützung
  - HTTP als Anwendungsprotokoll
    - Beschränkung auf die vorhandenen HTTP-Standardoperationen
    - Einhaltung der spezifizierten Semantiken
      - \* GET und HEAD sind rein lesende Methoden
      - \* POST, PUT und DELETE sind modifizierende Methoden
- Beispiele für Abweichungen von den REST-Grundprinzipien
  - Einsatz von Client-IDs in URIs
  - Verwendung von Cookies



## Entwicklung von Anwendungen im Vergleich

- SOAP
  - Modellierung der verwendeten Nachrichten
  - Konzipierung des Nachrichtenaustauschs (synchron/asynchron)
  - Auflistung der von der Anwendung angebotenen Operationen (→ WSDL)
- REST
  - Identifizierung der zu referenzierenden Ressourcen
  - Entwicklung einer geeigneten URI-Struktur
  - Definition der Operationssemantiken für jede Ressource
    - Welche der HTTP-Operationen darf auf der Ressource ausgeführt werden?
    - Welche Auswirkung hat eine bestimmte Operation auf die Ressource?
  - Formulierung von Beziehungen zwischen Ressourcen
    - Festlegung der in der Anwendung möglichen Zustandsübergänge
    - Erstellung von Hyperlinks
  - Spezifizierung der Datenrepräsentation für jede Ressource
- Literatur
  -  Cesare Pautasso, Olaf Zimmermann, and Frank Leymann  
**Restful Web Services vs. “Big” Web Services: Making the Right Architectural Decision**  
*Proceedings of the 17th Intl. World Wide Web Conference (WWW '08), S. 805–814, 2008.*



## Zusammenfassung

- Web Services Description Language (WSDL)
  - Beschreibung von Datentypen, Nachrichtenformaten und Schnittstellen
  - Abbildung auf Kommunikationsprotokolle
  - Ermöglicht automatisierte Generierung von Stubs für Web-Services
- SOAP
  - XML-basiertes Kommunikationsprotokoll für Web-Services
  - Schrittweise Weiterleitung von Nachrichten über Zwischenstationen
  - Transportprotokolle: HTTP, SMTP,...
- Representational State Transfer (REST)
  - Sammlung von Grundprinzipien für die Entwicklung von Web-Services
  - Grundkonzept des World Wide Web
  - Selbstbeschreibende Nachrichten
  - Hohe Skalierbarkeit durch zustandslose Interaktion
  - HTTP als Anwendungsprotokoll

