

Middleware – Cloud Computing

Web-Services

Wintersemester 2025/26

Tobias Distler, Christian Berger

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Systemsoftware)



Lehrstuhl für Informatik 4
Systemsoftware



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Web-Services

- Motivation

- Web Services Description Language (WSDL)

- Hypertext Transfer Protocol (HTTP)

- SOAP

- Representational State Transfer (REST)

- Vergleich zwischen SOAP und REST

■ 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

Motivation

Web Services Description Language (WSDL)

Hypertext Transfer Protocol (HTTP)

SOAP

Representational State Transfer (REST)

Vergleich zwischen SOAP und REST

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*)

■ Literatur



Web Services Description Language (WSDL) Version 2.0

<http://www.w3.org/TR/wsdl20/>



David C. Fallside and Priscilla Walmsley

XML Schema Part 0: Primer Second Edition, 2004.

■ 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]

■ Anwendungsspezifische 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>
```

■ 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]"/>
```

■ 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:[Methode]" 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
-
- Alternative: **HTTP als Anwendungsprotokoll** (z. B. mittels REST)

■ Beschreibung der **Kommunikationsendpunkte** eines **Web-Services**

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

Web-Services

Motivation

Web Services Description Language (WSDL)

Hypertext Transfer Protocol (HTTP)

SOAP

Representational State Transfer (REST)

Vergleich zwischen SOAP und REST

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, Paul Leach 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
PUT	Registrieren von Daten unter der übergebenen Ressourcenadresse
DELETE	Löschen der adressierten Ressource
POST	Übermittlung von Daten an die adressierte Ressource (Beispiele) – Weitergabe von Formulardaten – Anfügen eines Datensatzes an eine Datenbank

[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]

Web-Services

Motivation

Web Services Description Language (WSDL)

Hypertext Transfer Protocol (HTTP)

SOAP

Representational State Transfer (REST)

Vergleich zwischen SOAP und REST

■ Überblick

- **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/>

■ 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

- Weitergabe von Nachrichten entlang eines **Pfads aus SOAP-Knoten**

- Kategorien von SOAP-Knoten

- Sender
 - Zwischenstation
 - Empfänger

- Zwischenstationen dürfen Header-Blöcke lesen, hinzufügen oder löschen

- **Fehlermeldungen**

- 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>
```

- Beispiel: Aufbau eines Header-Blocks entspricht nicht den Regeln

■ HTTP

- Übertragung der SOAP-Nachrichten im Body von HTTP-Nachrichten
- Eigener Content-Type: `application/soap+xml`
- Signalisierung von SOAP-Fehlern: 500 Internal Server Error
- Senden von SOAP-Nachrichten
 - Unidirektional per HTTP-GET
 - Bidirektional per HTTP-POST
- **Implizite Zuordnung** von Nachrichten bei Anfrage-Antwort-Interaktion
 - Abhängigkeit zwischen HTTP-POST-Anfrage und ihrer Antwort
 - SOAP-Nachricht in der Antwort gehört zu SOAP-Nachricht in der Anfrage

■ SMTP

- Übertragung der SOAP-Nachrichten im E-Mail-Text oder als Anhang
- **Explizite Zuordnung** von Nachrichten bei Anfrage-Antwort-Interaktion
 - Keine von vornherein bestehende Abhängigkeit zwischen zwei E-Mails
 - Lösung: Message-Id der Anfrage im In-reply-to-Feld des Antwort-Headers

Web-Services

Motivation

Web Services Description Language (WSDL)

Hypertext Transfer Protocol (HTTP)

SOAP

Representational State Transfer (REST)

Vergleich zwischen SOAP und REST

Representational State Transfer (REST)

- Herausforderungen im **World Wide Web (WWW)**
 - Geografische Verteilung
 - Heterogene Hardware, Software, Datenformate,...
 - Vollständige Verfügbarkeit des Gesamtsystems nicht gegeben
- **Formulierung von Grundprinzipien** (Beispiele)
 - Aufteilung in Client und Server
 - Zustandslose Interaktion
 - Einheitliche Schnittstellen

„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.]

■ Literatur



Roy Fielding

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

- **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 vom 1.10.2025]
 - Manipulation von Ressourcen mittels **Repräsentationen**
 - Repräsentation kann vom internen Format einer Ressource abweichen
 - Ressourcenmodifikation durch Weitergabe einer veränderten Repräsentation
- **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
 - Nachteil: Mehraufwand durch vielfaches Senden von Kontextinformationen

Web-Services

Motivation

Web Services Description Language (WSDL)

Hypertext Transfer Protocol (HTTP)

SOAP

Representational State Transfer (REST)

Vergleich zwischen SOAP und REST

■ SOAP

- **Modellierung der zu verwendenden 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
- 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

Proc. of the 17th International World Wide Web Conference (WWW’08), S. 805–814, 2008.

Anwendungsbeispiel: Amazon Simple Storage Service

- **Amazon Simple Storage Service** (Amazon S3)
 - Web-Service zur Speicherung von Daten in der Amazon-Cloud
 - Ausfallsicherheit durch Replikation auf mehrere Standorte
- Organisation in **Buckets (Verzeichnisse)** und **Objects (Nutzdaten)**
 - Ein Bucket kann mehrere Objekte enthalten
 - Keine Schachtelung von Buckets vorgesehen
- Web-Service APIs
 - SOAP
 - REST
- Literatur



Amazon Simple Storage Service
<http://aws.amazon.com/s3/>



Amazon Simple Storage Service API Reference
<http://awsdocs.s3.amazonaws.com/S3/latest/s3-api.pdf>

■ Erstellen eines (Text-)Objekts in einem Bucket: Anfragenachricht

```
<PutObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>[Bucket-Name]</Bucket>
  <Key>[Objektnamen]</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <ContentLength>[Größe der Nutzdaten]</ContentLength>
  <AccessControlList>[Spezifizierung von Zugriffsrechten]</AccessControlList>
  <AWSAccessKeyId>[Zugangsschlüssel-ID]</AWSAccessKeyId>
  <Timestamp>[Zeitstempel]</Timestamp>
  <Signature>[Signatur]</Signature>
</PutObject>
```

- Übertragung der Nutzdaten im Anhang an den SOAP-Envelope
- Ersetzung der Daten, falls ein Objekt desselben Namens bereits existiert

■ Erstellen eines Objekts in einem Bucket: Antwortnachricht (Body)

```
<PutObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectResponse>
    <ETag>[MD5-Hash der Nutzdaten des Objekts]</ETag>
    <LastModified>[Zeitstempel der letzten Speicheroperation dieses Objekts]</LastModified>
  </PutObjectResponse>
</PutObjectResponse>
```

■ Zugriff auf Objekte

```
<GetObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>[Bucket-Name]</Bucket>
  <Key>[Objektnamen]</Key>
  <GetMetadata>[true | false]</GetMetadata>
  <GetData>[true | false]</GetData>[...]
</GetObject>
```

- Getrennte Abfrage von Daten und Metadaten möglich
- Rückgabe der Daten im SOAP-Body oder als SOAP-Anhang

■ PUT: Erstellen eines (Text-)Objekts in einem Bucket

```
PUT /[Objektname] HTTP/1.1
Host: [Bucket-Name].s3.amazonaws.com
Date: [Zeitstempel]
Authorization: AWS [Zugangsschlüssel-ID]:[Signatur]
Content-Type: text/plain
Content-Length: [Größe der Nutzdaten]
```

- Nutzdaten des Objekts im Nachrichten-Body
- Hierarchische Objektnamen möglich (z.B. dir/subdir/file.txt)

■ Zugriff auf Objekte

- Varianten
 - GET: Abfrage der Nutz- und Metadaten
 - HEAD: Abfrage der Metadaten
- Komplexere Leseanfragen durch Hinzufügen von Header-Feldern

■ DELETE: Löschen von Objekten