RESTful Web Services

W3L AG
info@W3L.de

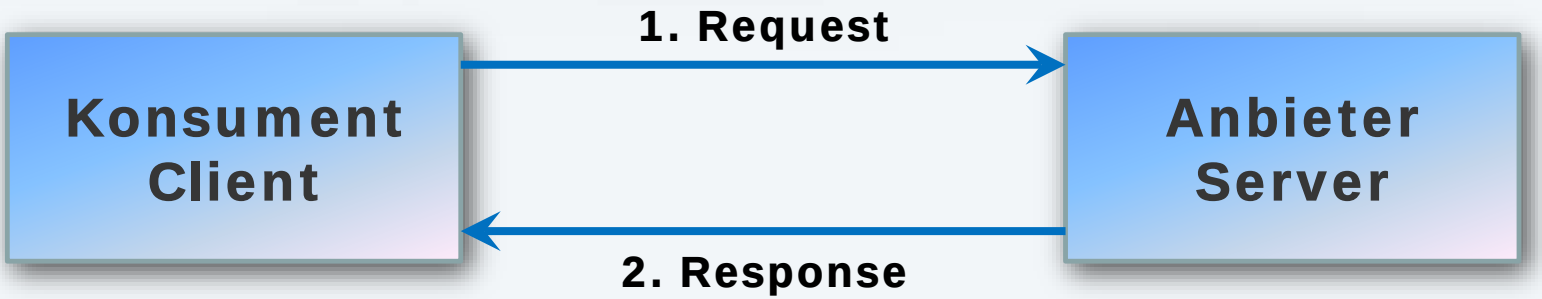
2014

Inhaltsverzeichnis

zurück

Was ist ein Web Service?

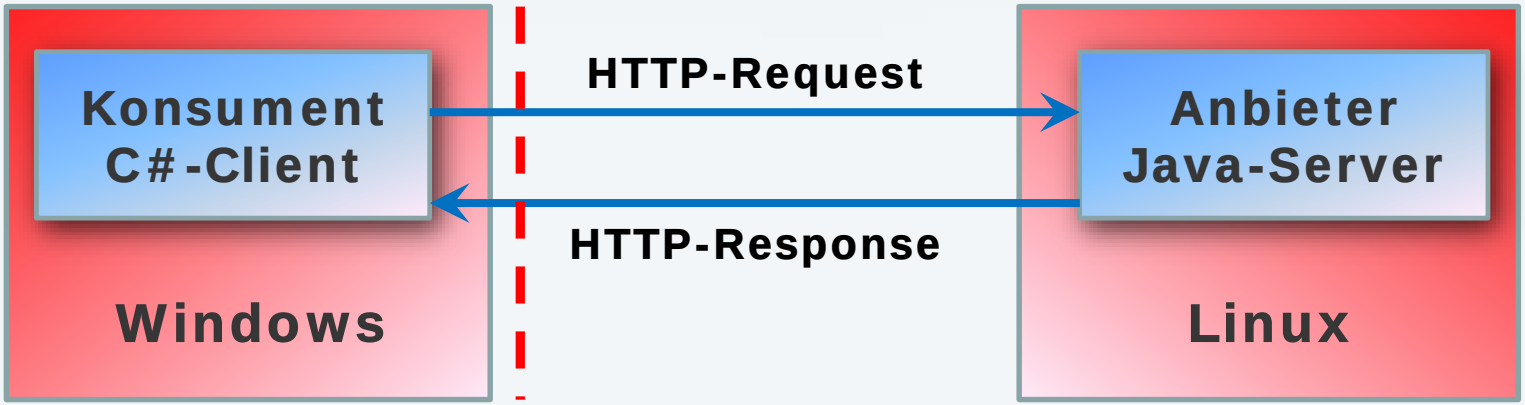
- **per Netzwerk bereitgestellte Softwareanwendung**
  - zum Abruf eines über das Web verfügbaren Datenangebots
  - ermöglicht automatische Informationsbeschaffung ohne User-Eingriff



- **Anwendungsbereiche**
  - B2B – Abwicklung von Geschäftsprozessen über Unternehmensgrenzen hinweg
  - Zugang zu Geodaten
  - Bereitstellung externer Funktionen
  - Übertragung von TÜV-Seminaren zur Certqua

Was ist ein Web Service?

- **Vorteile**
  - nicht an Übertragungsprotokoll gebunden
  - Interoperabilität

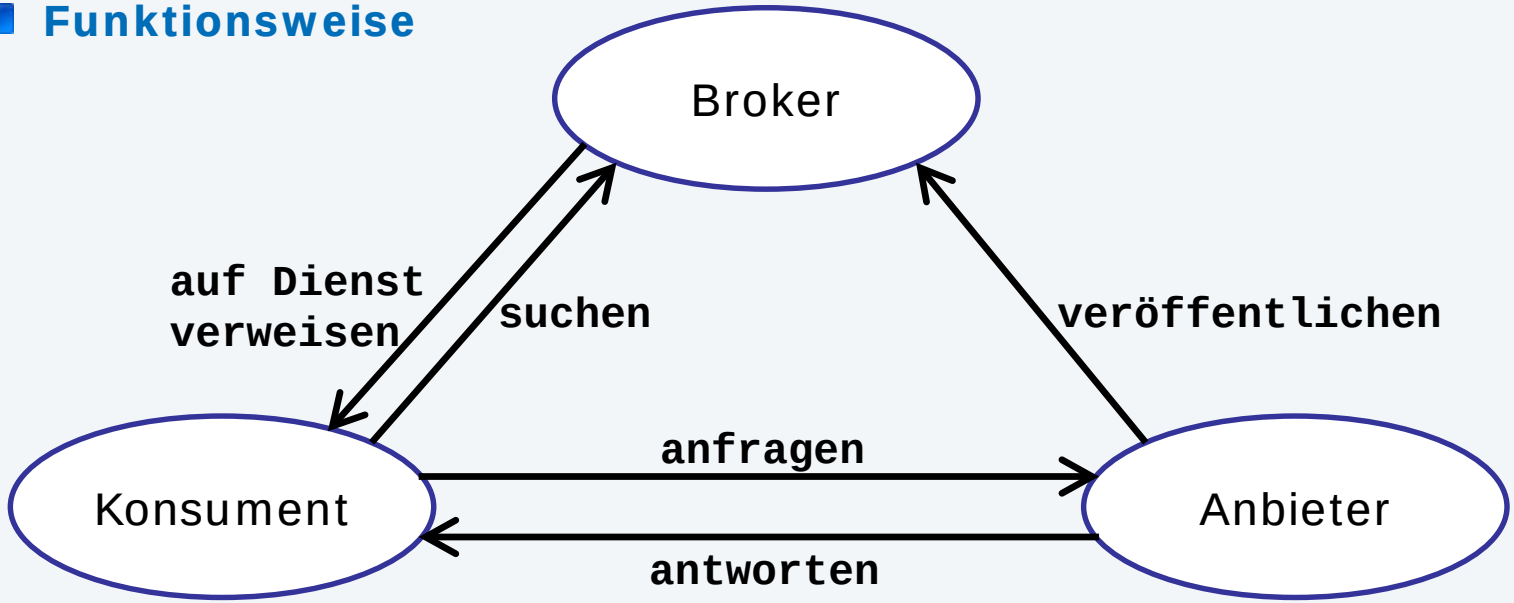


- **Nachteile**
  - Performance
  - Sicherheitsaspekte
  - externe Bibliotheken

Techniken zur Implementierung eines Web Services

SOAP

- **Netzwerkprotokoll zum Datenaustausch**
  - XML-Format
- **Aufbau einer SOAP-Nachricht**
  - XML-Element Envelope mit Header und Body
- **Funktionsweise**



Techniken zur Implementierung eines Web Services

SOAP

- **Beispiel**
  - Anfrage an Datenbank nach Mitarbeiter mit angegebener ID

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Body>
    <m:MitarbeiterInDatenbank xmlns:m="http://www.db.de/soap">
      1574
    </m:MitarbeiterInDatenbank>
  </s:Body>
</s:Envelope>
```

  - Auswertung der Nachricht durch den Server

```
startElement(„MitarbeiterInDatenbank“, […])
```

  - Antwort des Servers mit einer Soap-Nachricht

Techniken zur Implementierung eines Web Services

REST – Representational State Transfer

- **fünf Prinzipien für REST-Dienste**
  - Zustandslosigkeit
  - Adressierbarkeit

```
http://www.beispiel.de/rest-service
http://www.beispiel.de/rest-service/artikel/1
http://www.beispiel.de/rest-service/kunde/532
```

  - Repräsentationen
  - Operationen
  - Verwendung von Hypermedia

REST und SOAP im Vergleich

- **REST**
  - generisches Interface
    - GET, POST, PUT, DELETE
  - Verwendung von Standards
    - URI, HTTP, XML, ...
  - mehrere Nachrichtenformate
    - XML, JSON, ...
  - WADL zur Beschreibung
  - direkt adressierbare Ressourcen
- **SOAP**
  - kein generisches Interface
    - Methoden selbst definieren
  - Schaffen eigener Standards
    - WSDL, UDDI, ...
  - festgelegtes Nachrichtenformat
    - XML
  - WSDL zur Beschreibung
  - Adressierung per Dispatcher

REST und SOAP im Vergleich

Vorteile von REST - Beispiele

- **Suche nach Produkt „Web Services“ in Online-Shop**
  - URI Web-Anwendung/Web Service (SOAP):
    - `http://www.online-shop.de/angebot`
  - URI REST-Anwendung:
    - `http://www.online-shop.de/angebot/artikel/549`

- **Zugriff auf Ressource von verschiedenen Clients**
  - spezielle Anwendung liest aus Ressource für sie relevante Daten und verwertet sie
  - per Browser lässt sich eine HTML-Repräsentation der Ressource zur Einsicht/Überwachung öffnen

REST und SOAP im Vergleich

	SOAP	REST
ANFRAGE	<?xml version="1.0"?> <s:Envelope xmlns:s= "http://www.w3.org/2003/05/soap-envelope"> <s:Body> <m:MitarbeiterInDatenbank xmlns:m="http://www.db.de/soap"> 1574 </m:MitarbeiterInDatenbank> </s:Body> </s:Envelope>	GET /mitarbeiter/1574 HTTP/1.1
AUSWERTUNG	startElement("MitarbeiterInDatenbank", […]) {…}	@GET @Path("mitarbeiter/{id}") @Produces("text/xml") getMitarbeiter(@PathParam("id") String id) {…}
ANTWORT	<?xml version="1.0"?> <s:Envelope xmlns:s= "http://www.w3.org/2003/05/soap-envelope"> <s:Header> <m:RID xmlns:m="http://www.db.de/soap"> #40215781 </m:RID> </s:Header> <s:Body> <m:mitarbeiter id="1574" xmlns:m="http://www.db.de/soap"> <m:nachname>Meier</m:nachname> <m:vorname>Sven</m:vorname> </m:mitarbeiter> </s:Body> </s:Envelope>	HTTP/1.1 200 OK Content-Type: text/xml Content-Length: 119 <?xml version="1.0"?> <m:mitarbeiter id="1574" xmlns:m="http://www.db.de/rest"> <m:nachname>Meier</m:nachname> <m:vorname>Sven</m:vorname> </m:mitarbeiter>

Realisierung von RESTful Web Services in Java

JAX-RS

- **Java API for RESTful Web Services**

- **Veröffentlichungen in der Java Platform Enterprise Edition**
  - JAX-RS 1.1 in Java EE6
  - JAX-RS 2.0 in Java EE7

- **Referenzimplementierungen für JAX-RS**
  - Restlet
  - RESTEasy
  - Apache Wink
  - Jersey

  - Implementierung von Spring

Implementierung von RESTful Web Services mit JAX-RS

Server

- **Tomcat-Server hinzufügen**
- **dynamisches Web Projekt erstellen**
- **Web Projekt dem Server hinzufügen**

- **Klasse der übergebenen Objekte erzeugen**
  - Person.java
  - Annotation zur XML-Ausgabe
  - parameterloser Konstruktor
- **Ressourcen-Klasse erzeugen**
  - RestRessource.java
  - Bibliothek „jsr311-api-1.1.jar“ importieren
  - Methoden implementieren und annotieren

Implementierung von RESTful Web Services mit JAX-RS

Server

- **Annotationen**
  - @Path: Pfad, über den auf eine Ressource zugegriffen werden kann
  - @GET, @POST: HTTP-Methoden
  - @Produces: den Media-Typen der Ausgabe festlegen; z.B. XML, HTML, Text
  - @Consumes: den Media-Typen der Eingabe festlegen
  - @PathParam: in der URL übergebene Parameter einlesen
  - @QueryParam: in der URL übergebene Parameter in Form von Name-Wert-Paaren einlesen
  - @FormParam: per Formular übergebene Parameter einlesen

Implementierung von RESTful Web Services mit JAX-RS

Server

- **Ressourcen registrieren**
  - Einbinden der Bibliothek `jaxrs-ri` (Jersey)
  - Konfiguration von web.xml

```
<servlet>
  <servlet-name>RESTService</servlet-name>
  <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>jersey.config.server.provider.packages</param-name>
    <param-value>restserver</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>RESTService</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

Implementierung von RESTful Web Services mit JAX-RS

Client

- **JSP**
  - GET-Operationen können durch JavaScript angestoßen werden
    - @PathParam
  - POST-Operationen können per Formular angestoßen werden
    - @FormParam
  - Client muss die Klasse Person nicht kennen

- **Java-Client**
  - Paket javax.ws.rs.client (seit Java EE7)
  - Einbinden der Bibliothek `jaxrs-ri.2.5.1` (Jersey)
  - Client- und WebTarget-Objekt erzeugen

Web Application Description Language

wadl2java

- **automatisch generierte WADL-Datei**
  - im Beispiel:
    `http://localhost:8082/RestServer/rest/application.wadl`

- **Tool wadl2java**
  - generiert aus der WADL-Datei Java-Klassen
  - Konsoleneingabe:

```
wadl2java –o C:\Users\...\REST –p de.wadl2java
                        http://localhost:8082/RestServer/rest/application.wadl
```

Quellen und Verweise

- **Orientation in Objects: REST Web Services**
  - http://www.oio.de/public/xml/rest-webservices.htm
- **JAXenter: Rest – Der bessere Web Service?**
  - http://jaxenter.de/artikel/REST-bessere-Web-Service-167838
- **Galileo Computing: RESTful und SOAP Web-Services**
  - http://openbook.galileocomputing.de/java7/1507_13_001.html
- **Jersey: RESTful Web Services in Java**
  - http://jersey.java.net/

Inhouse-Schulungen

Wir bieten Inhouse-Schulungen und Beratung durch unsere IT-Experten und –Berater.

Schulungsthemen

- Softwarearchitektur (OOD)
- Requirements Engineering (OOA)
- Nebenläufige & verteilte Programmierung

Gerne konzipieren wir auch eine individuelle Schulung zu Ihren Fragestellungen.

Sprechen Sie uns an!
Tel. 0231/61 804-0, info@W3L.de

W3L-Akademie

In Zusammenarbeit mit der Fachhochschule Dortmund bieten wir

zwei Online-Studiengänge

- B.Sc. Web- und Medieninformatik
- B.Sc. Wirtschaftsinformatik

und 7 Weiterbildungen im IT-Bereich an.

Besuchen Sie unsere Akademie!
http://Akademie.W3L.de