REST: Eine leichtgewichtige und einfachere Alternative zu Web Services

W3L AG
info@W3L.de

2009

Inhalt

- Einführung
- Grundprinzipien der REST-Architektur
- Beispiel
- Entwurf von REST-Anwendungen
- REST mit Java
- Zusammenfassung

zurück

Einführung

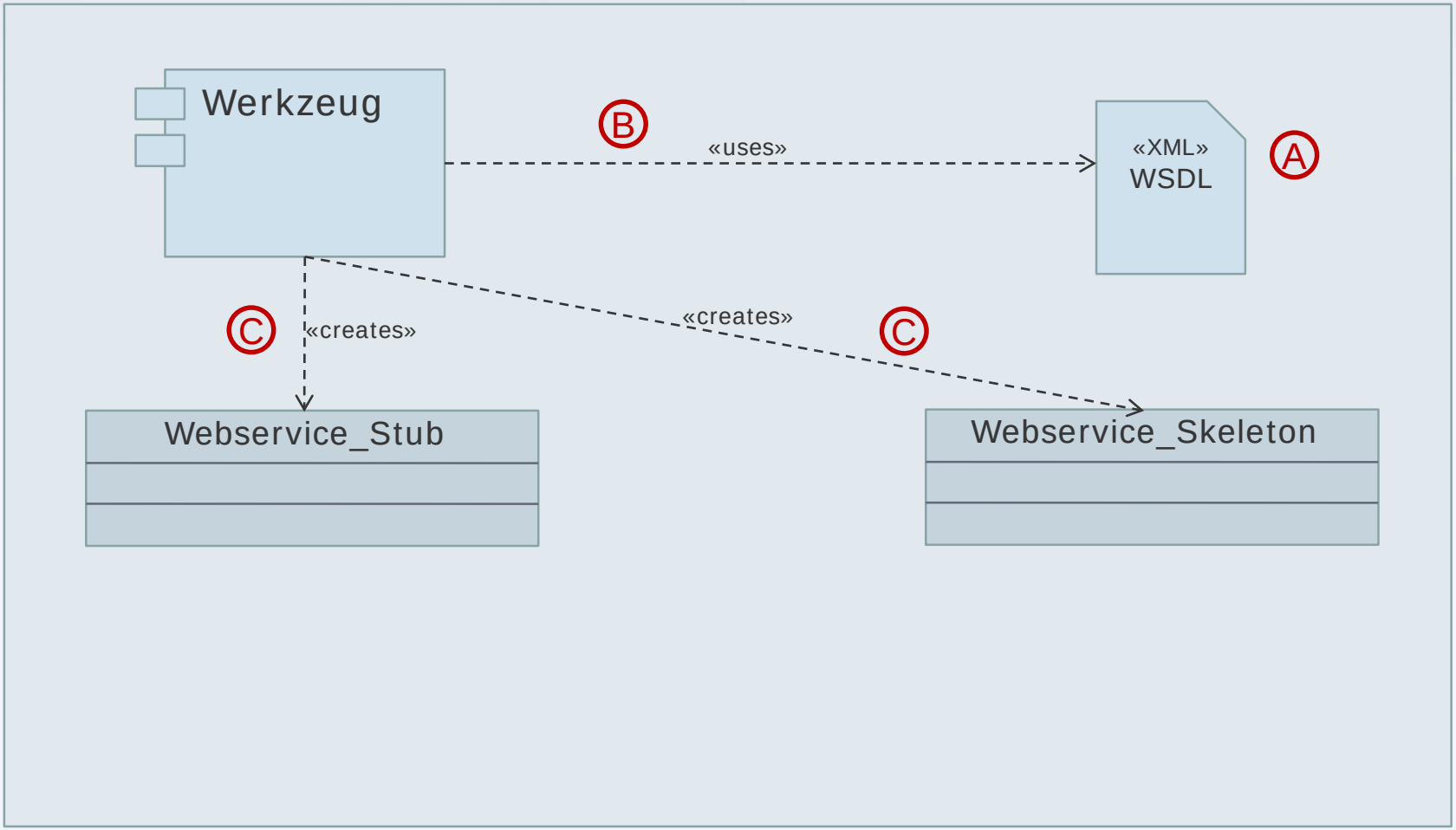
- **Web Services**
  - Stück Software, welches über eine Schnittstelle mit bereitgestellten Operationen verfügt.
  - Wunsch nach hoher Wiederverwendung durch Interoperabilität.
  - Kommunikation über ein Netzwerk
    - Dazu gehört auch das Internet!
  - W3C Spezifikationen und Empfehlungen
  - Andere Ansätze verfolgen ähnliche Ziele: CORBA!
  - Anwendungsgebiete
    - RPC (Kommunikationseinheit Operationsaufruf)
    - SOA (Kommunikationseinheit Nachricht)

Einführung

- **Neuerdings zwei Lager**
  - Big Web Services
    - WSDL, XML und SOAP
    - Java oder .NET
    - Spring, Apache Axis 2, Apache CXF
  - **REST**ful Web Services
    - Entsprechen auch der W3C Definition von Web Services
    - Schnittstelle nicht erweiterbar
    - Ressourcen-orientiert

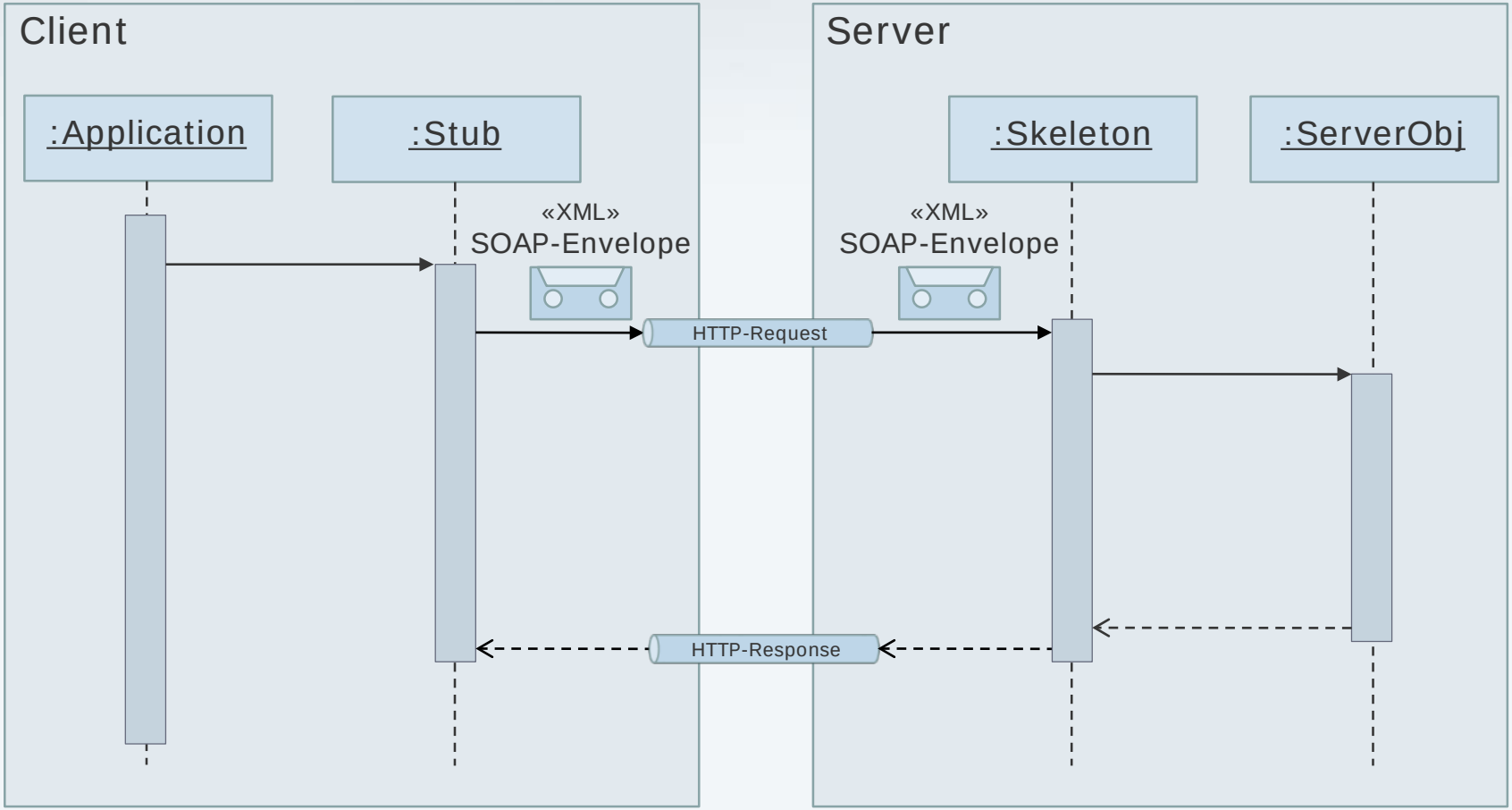
Einführung

- **Big Web Services**
  - Vorgehensweise *(Contract First)*



Einführung

- **Big Web Services**
  - Kommunikationsprozess



Einführung

- **Motivation**
  - *»The motivation for developing REST was to create an architectural model for how the Web should work, such that it could serve as the guiding framework for the Web protocol standards.«*
    Roy T. Fielding [Fielding 2000]

- **Worin besteht der Bezug zu Web Services?**
  - REST ist ein Architektur-Modell für die WWW-Anwendung!
  - REST lässt sich aber auch in anderen Kontexten einsetzen!

Einführung

- **Was ist REST?**
  - REpresentational State Transfer!
  - Der Begriff wurde durch Roy T. Fielding in seiner Dissertation eingeführt.
  - Roy T. Fielding ist
    - Kernentwickler einiger Web-Standards und
    - ehemaliger Vorsitzender der Apache Software-Foundation.
  - Software-Architektur für verteilte Hymedia-Anwendungen.
    - Anwendung besteht aus einer Menge von Ressourcen, die adressiert werden können.
    - Auf diesen Ressourcen können nur fest vorgegebene Operationen aufgerufen werden.
    - Die Antwort auf einen Operationsaufruf kann eine Menge von Zustandsübergängen zurückliefern.
    - Zustandsübergang findet im Client statt!
    - Representational State Transfer!
  - Bestes Beispiel ist: World Wide Web!

Einführung

- **Was ist REST?**
  - REST-Grundprinzipien
    - Zustandslose Client/Server-Kommunikation
    - Identifizierbare Ressourcen als zentrale Dekompositionseinheit
    - Ressourcen-Repräsentation
    - Uniforme Schnittstelle
    - Hypermedia

Grundprinzipien der REST-Architektur

- **Zustandlose Kommunikation**
  - Kommunikation zwischen Client und Server erfolgt bei REST zustandlos
  - Es gibt keine Session bzw. kein Server-seitiges Zustandsverwaltung.
  - Alle Informationen einer Anfrage müssen mit der Anfrage gesendet werden.
  - Konsequenzen
    - Server muss keine Ressourcen vorhalten.
    - Vorteile für Skalierbarkeit, Lastverteilung, Ausfallsicherheit, usw.

Grundprinzipien der REST-Architektur

- **Identifizierbare Ressourcen**
  - Ressourcen sind das zentrale Konzept in REST.
    - REST-Architekturen werden daher oft auch als **ROA**‘s bezeichnet.
    - ***resource-oriented-architecture***
  - Eine Ressource ist jedes wiedererkennbare Gebilde innerhalb einer Anwendung.
    - Beispiele folgen…

Grundprinzipien der REST-Architektur

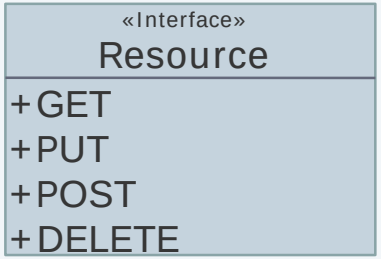
- **Uniforme bzw. einheitliche Schnittstelle**
  - Markanteste Charakteristik ist das REST-Prinzip der uniformen Schnittstelle.
  - Im Gegensatz zu anderen *remote-procedure-calls*-Ansätzen gibt es nur eine Schnittstellenvereinbarung.
    - Alle Ressourcen müssen prinzipiell die Operationen dieser Schnittstelle unterstützen (HTTP-Operationen).
    - Eine Analogie ist die deklarative Sprache SQL mit `INSERT`, `DELETE`, `UPDATE` und `SELECT`.
  - Vorteile
    - Universalität
    - Siehe Web-Browser

Grundprinzipien der REST-Architektur

- **Hypermedia**
  - *»Hypermedia as the Engine of Application state«*
  - Server überträgt nicht nur Daten zum Client, sondern auch die durch den Client initiierbaren Zustandsübergänge in Form von sogenannten Links.
  - Wieder Analogie zum Web-Browser
    - Hyperlinks

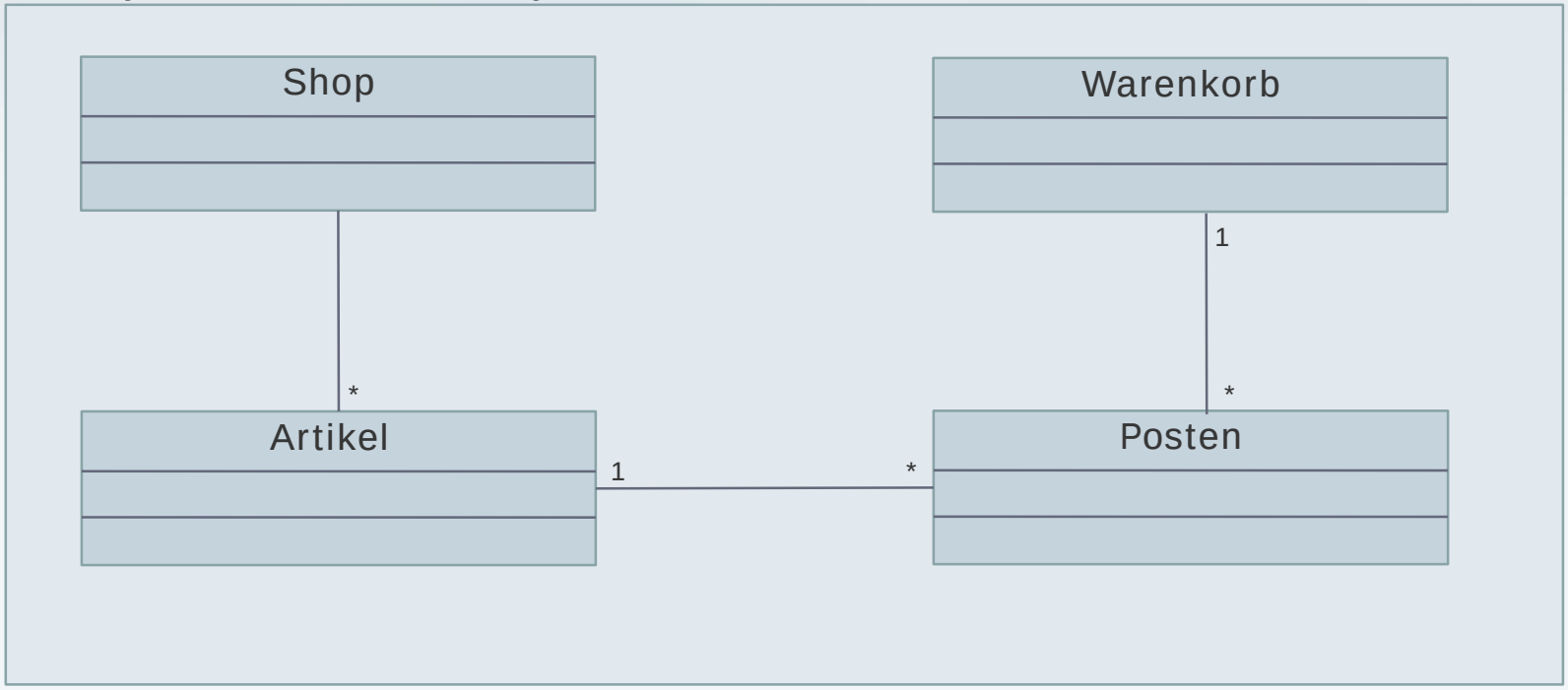
Grundprinzipien der REST-Architektur

- **Ressourcen-orientierte Analyse**
  - Jedes einzelne Objekt der Anwendung stellt eine Ressource dar.
  - Jede Ressource kann über eine URL erreicht werden.
  - Auf jeder Ressource lassen sich HTTP-Methoden aufrufen: GET, PUT, POST und DELETE.
    - GET
      Fordert eine Repräsentation einer Ressource an.
    - PUT
      Erfasst eine Ressource oder manipuliert eine bestehende.
    - POST
      Übermittelt Daten zwecks Verarbeitung (funktionale Ressourcen)
    - DELETE
      Löscht eine Ressource.



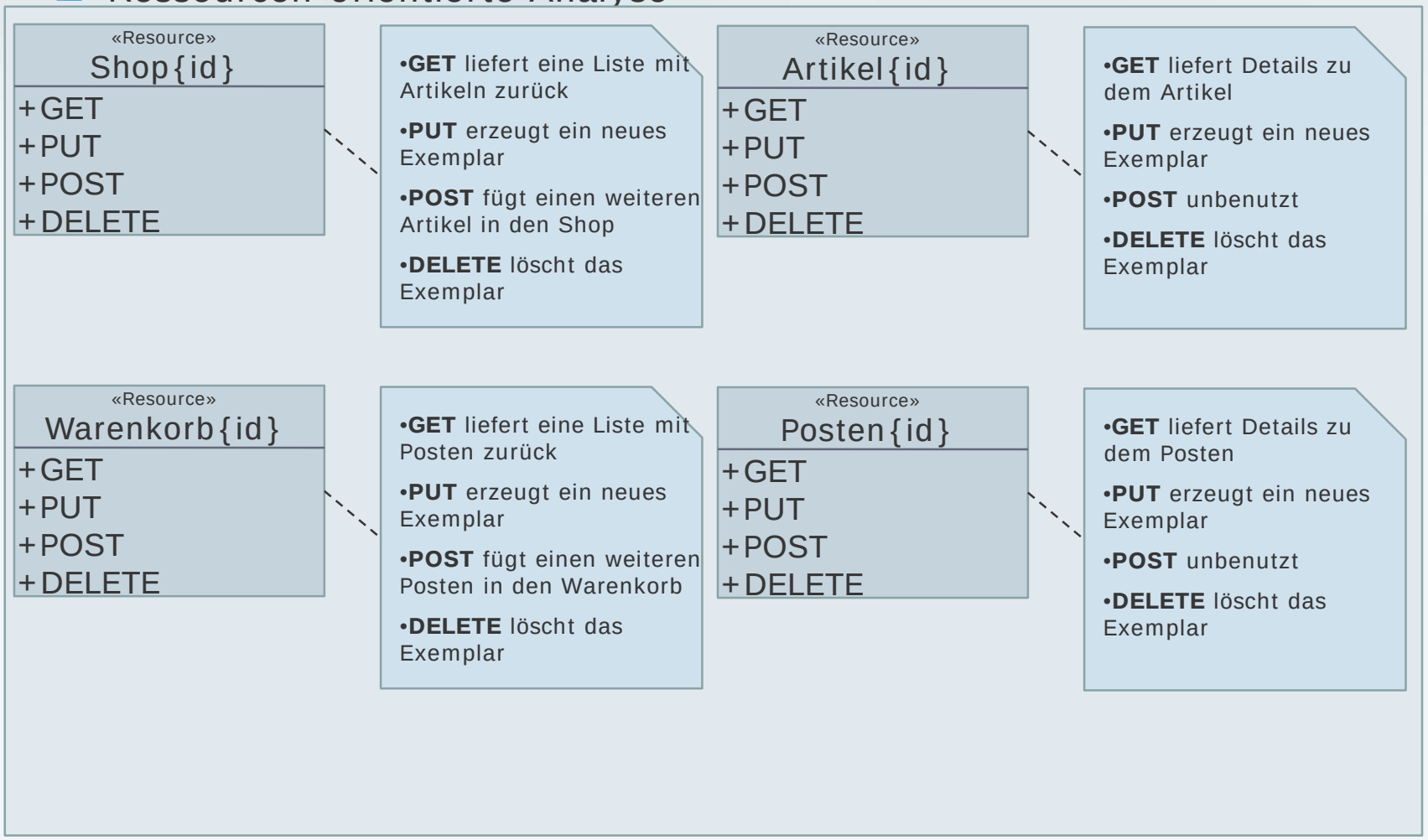
Beispiel

- **Online-Shop**
  - Objektorientierte Analyse



Beispiel

- **Online-Shop**
  - Ressourcen-orientierte Analyse



Beispiel

Zugriff auf die Artikelliste in einem Shop

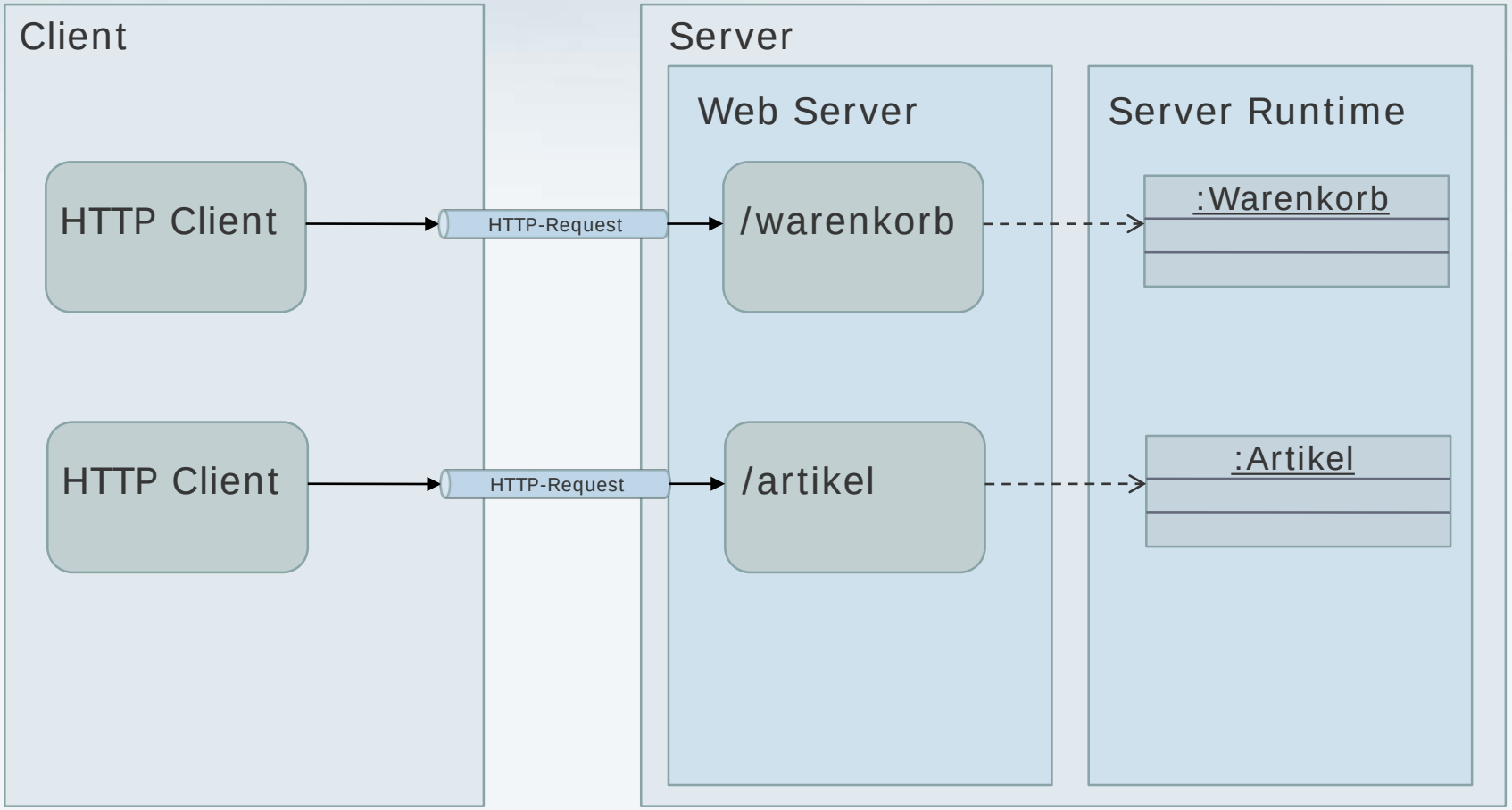
```
//Hier wird ein HTTP-GET auf die entsprechende URL abgesetzt
curl http://localhost/Shop/4711
```

Mögliche HTTP-Response

```
HTTP/1.1 200 OK Content-Type: text/xml
<?xml version="1.0"?>
<Shop ID="4711" xmlns:xlink="http://www.w3.org/1999/xlink">
   <Artikel xlink:href="http://localhost/Artikel/1" ID="1">
       <Titel>Java 6: Anwendungen programmieren</Titel>
   </Artikel>
   <Artikel xlink:href="http://localhost/Artikel/2" ID="2">
       <Titel>Basiswissen Analysis</Titel>
   </Artikel>
</Shop>
```

Beispiel

Adressierung von Objekten



Entwurf von REST-Anwendungen

- **REST = CRUD ?**
  - Einheitliche Schnittstelle ist auf die Operationen GET, PUT, POST und DELETE beschränkt.
  - Ist daher nur simple Datenverarbeitungsanwendungen realisierbar?
  - Nein!
- **REST != CRUD !**
  - Viele Operationen haben die Bearbeitung oder Neuanlage von Geschäftsobjekten zum Ziel.
    - Hier ist die Abbildung auf die einheitliche Schnittstelle einfach.
  - Bei anderen Operationen oder Geschäftsprozessen muss am umdenken.
    - Aus Verben werden Substantive!
    - Anstatt **`commitOrder`** muss **`createOrderCommitment`** verwendet werden!

Entwurf von REST-Anwendungen

- **1. Schritt**
  - Welche Ressourcen existieren und wie sind diese miteinander verknüpft.
- **2. Schritt**
  - *Collections* müssen ebenfalls als Ressourcen modelliert werden, sofern auf eine Liste von Exemplaren zugegriffen werden soll.
- **3. Schritt**
  - Über welche URI‘s sollen die Ressourcen adressiert werden?
  - Sinnvolle und leserliche URI‘s sind von Vorteil.
- **4. Schritt**
  - Welche zusätzlichen Repräsentationen soll es für bestimmte Ressourcen geben?
  - *Content-Type-Negotiation*

Entwurf von REST-Anwendungen

- **REST und Sicherheit**
  - REST-basierte Anwendungen verwendet HTTP-Operationen und URLs.
  - Firewalls können feingranular Zugriffe beschränkten.
    - Für bestimmte Adressbereiche könnte beispielsweise der Zugriff auf eine REST-Anwendung auf einen lesenden Zugriff beschränkt werden.
  - REST vs. Big Web Services
    - Durch das SOAP-Envelope sehen alle Aufrufe bei den Big Web Services gleich aus (HTTP-POST)
    - Eine Firewall kann die Semantik erst verstehen, wenn sie die den Inhalt des SOAP-Envelope parst.
    - Hier sind andere Sicherheitsmechanismen erforderlich.
    - Evtl. sind diese schwergewichtiger!

REST mit Java

- **Historie**
  - Im Februar 2007 hat Sun den **JSR 311** *(Java Specification Request)* herausgegeben.
    - JSR 311: JAX-RS: The Java API for RESTful Web Services.
    - `http://jcp.org/en/jsr/detail?id=311`
  - Seit 10. Oktober 2008 steht die finale Version 1.0 des JAX-RS zur Verfügung.
- **Ziele**
  - Das JAX-RS hat das Ziel, das Verfolgen der REST-Prinzipien innerhalb von Java-Anwendungen einfacher zu machen.

REST mit Java

Java Implementierung des Warenkorbs

```
import javax.ws.rs.*;

//Über diese Annotation lässt sich die URI dieser Ressource definieren
@Path("/warenkorb")
public class Warenkorb {
  //Diese Annotation weist auf das HTTP-GET und den Content-Type hin.
  @GET @Produces("text/plain")
  public String listeArtikel() {
     return "1,2,3";
  }

}
```

Zusammenfassung

- **Vorteile**
  - Der Zugriff über HTTP und URI‘s auf Ressourcen der Anwendung ermöglicht die Öffnung der Anwendung für externe Suchmaschinen.
- **Nachteile**
  - Die Schnittstelle einer Anwendung durch REST-Prinzipien auf Ressourcen auf Basis von HTTP und URI‘s ist nicht einfach.
  - Ein Umdenken von bekannten Konzepten ist notwendig.
- **Fazit**
  - Ressourcenorientierung, die einheitliche Schnittstelle und Hypermedia sind zentrale Konzepte von REST.
  - Wenn Architekturziele aus loser Kopplung, Interoperabilität, Plattformunabhängigkeit, Skalierbarkeit und Internetfähigkeit bestehen, dann ist RESTful HTTP eine gute Wahl.

Vielen Dank!

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