

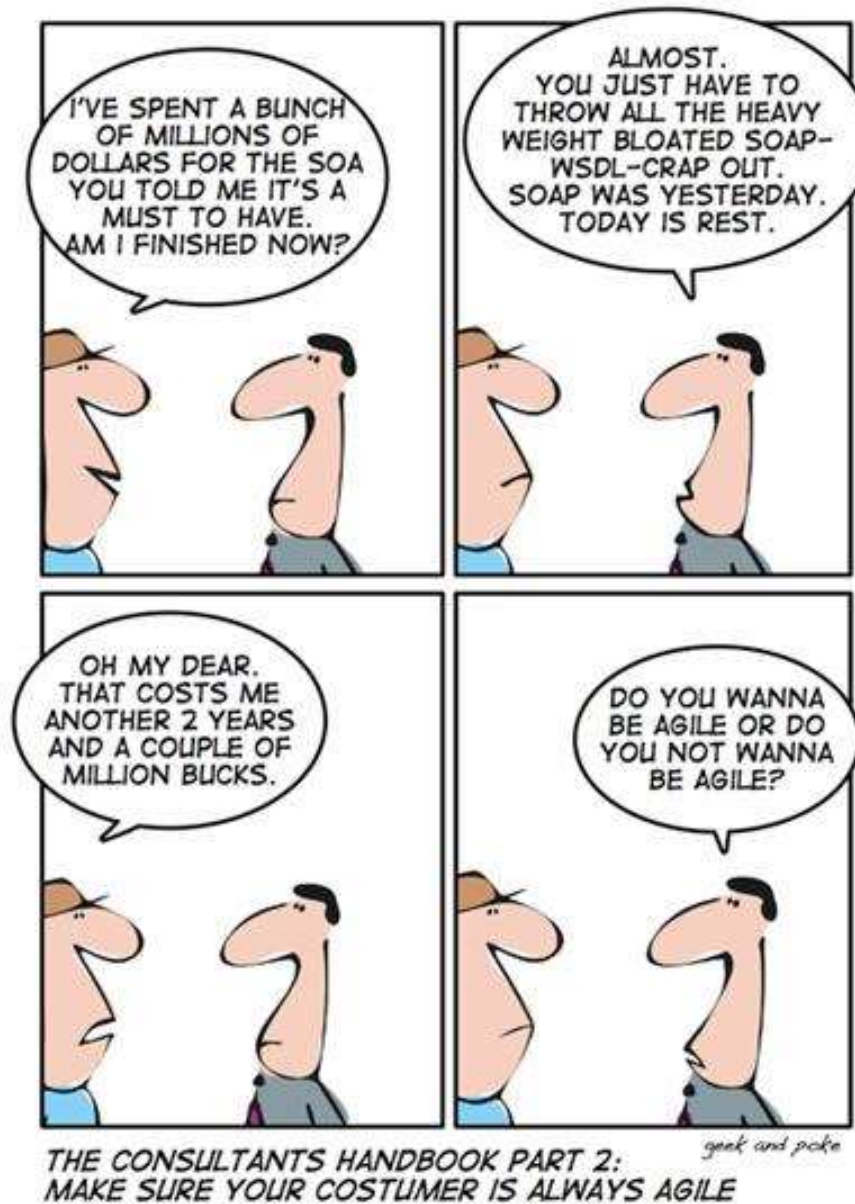
Prof. DI Dr. Erich Gams

Web Services 2

RESTful Services

dezentrale systeme - htl-wels


Soap und REST



Übersicht ➡ Was lernen wir?

- Was ist REST?
- Eigenschaften
- Beispiel

Web Services mit REST

- Neben SOAP gibt es eine weitere Alternative für die Realisierung von Web Services
 *REpresentational State Transfer Architektur (REST)*
- Dissertation von Thomas Roy Fielding *REpresentational State Transfer Architektur (REST)*
(<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>)
"Representation 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 an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use."
- REST ist kein Protokoll wie SOAP, sondern ein Architekturstil!

Was ist REST?

- Operationenaufrufe und Argumente **werden nicht in einem XML Dokument** übergeben
- **Ressource** wird in einer **URL kodiert** und nur wenige Operationen sind möglich
- Im **Mittelpunkt steht eine Ressource**, die eindeutig adressierbar ist

Was ist REST?

- Die *REpresentational State Transfer Architektur* ist ein Architektur Modell, welches beschreibt, wie das Web funktionieren sollte.
- Das World Wide Web stellt selbst eine gigantische REST Anwendung dar.
- REST ist kein Produkt oder Standard.
- REST beschreibt, wie Web Standards in einer „Web-gerechten“ Weise eingesetzt werden können.

REST Eigenschaften

- REST Web Services sind immer synchron und folgen dem Kommunikationsmuster Request/Response.
- ➡ Statuslose Client/Server Kommunikation
- Verwendet HTTP Operation GET,PUT,POST,DELETE
- URIs um alle Ressourcen anzusprechen
- Navigation durch Selektion von URIs
- Zugriff auf einen Web Services durch die *unmittelbare* Verwendung des HTTP-Protokolls

SOAP vs REST

- SOAP is a protocol whereas REST is an architectural style.
- SOAP server and client applications are tightly coupled and bind with the WSDL contract whereas there is no contract in REST web services and client.
- Learning curve is easy for REST when compared to SOAP web services.
- REST web services request and response types can be XML, JSON, text etc. whereas SOAP works with XML only.
- JAX-RS is the Java API for REST web services whereas JAX-WS is the Java API for SOAP web services.

■ Ressource

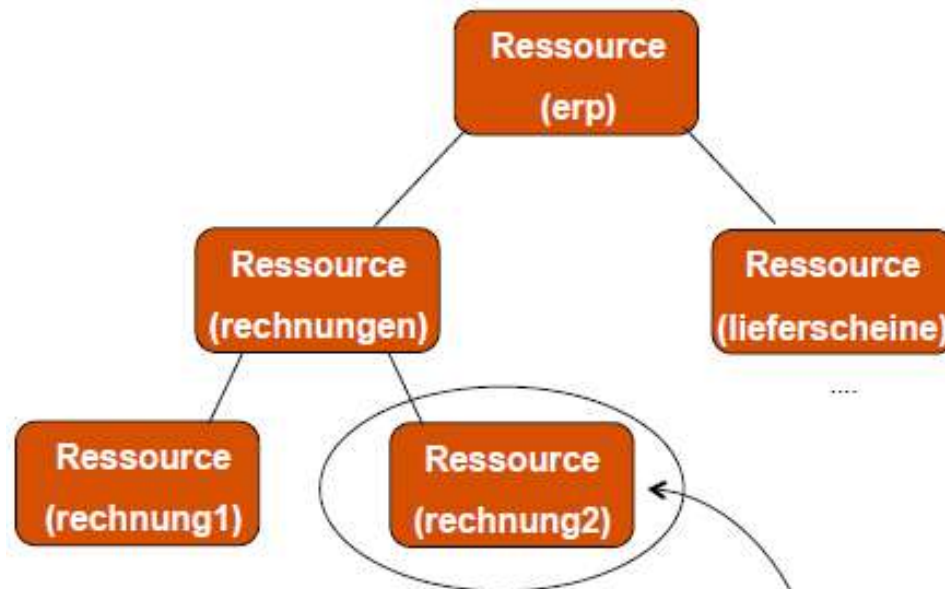
- Eindeutig identifizierbares Objekt (bspw. eine Person)
- In ressourcenorientierter Architektur immer adressierbar

■ Ressourcenidentifikation

- Im Web einheitlicher Standard: **Uniform Resource Identifier**
- Eine URI adressiert eine Ressource.

Adressierung von Ressourcen

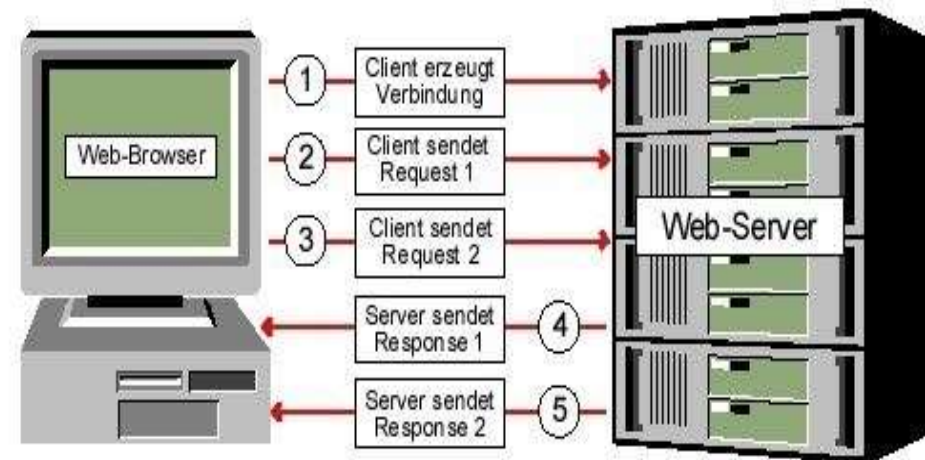
- Ressourcen können hierarchisch angeordnet sein



In Rest adressiert unter:
<http://myserver/erp/rechnungen/rechnung2>

HTTP-Verbindung

- Die Kommunikation zwischen Client und Webserver erfolgt durch den Austausch von HTTP-Nachrichten. Diese Nachrichten übertragen die Anfragen und Antworten zwischen Client und Server.



tecChannel

HTTP-Methoden in REST



GET

- Abfrage einer Repräsentation einer Ressource



POST

- Hinzufügen einer Ressource in einer gegebenen Hierarchie



DELETE

- Löschen einer Ressource

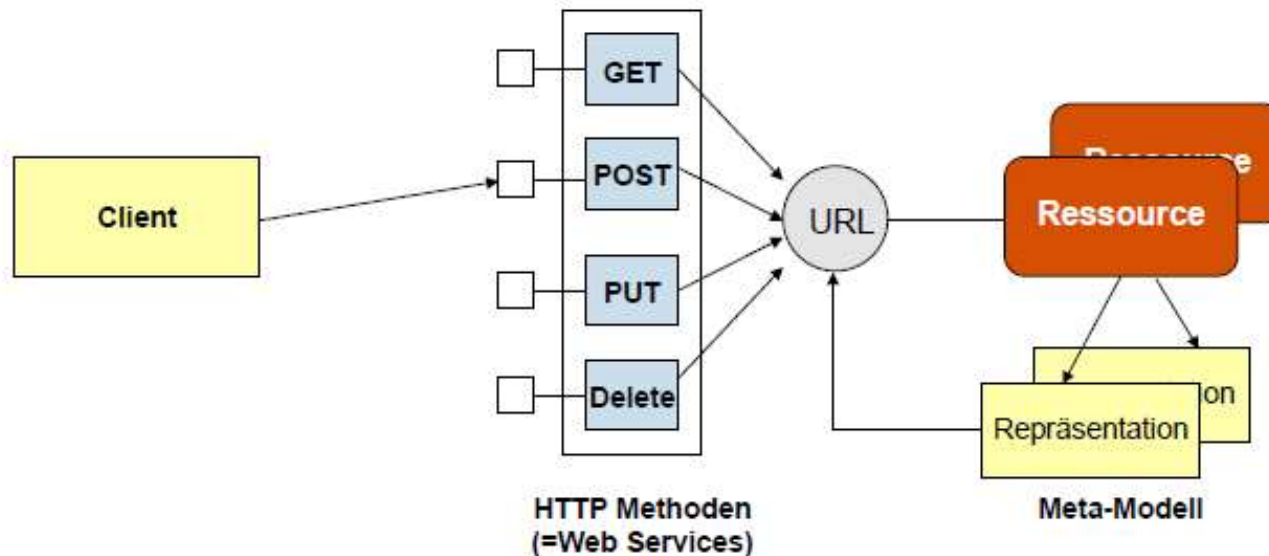


PUT

- Aktualisierung Daten einer vorhandener Ressource (Statusänderung)
- Hinzufügen einer Ressource, die nicht in einer gegebenen Hierarchie anderen Ressourcen untergeordnet ist

SAMPLE: Common operations on news item object	SOAP approach	RESTful approach
Create news item	CreateNewItem(string id, string title)	/news.svc/{id} HTTP METHOD: PUT
Update news item	UpdateNewItem(string id)	/news.svc/{id} HTTP METHOD: PUT
Get news item	GetNewItem(string id, string title)	/news.svc/{id} HTTP METHOD: GET
Get news items	GetNewsItems()	/news.svc/ HTTP METHOD: GET
Delete news item	DeleteNewItem(string id)	/news.svc/{id} HTTP METHOD: DELETE

Aufbau einer REST-basierten Web Services Architektur



- Ressource = Web Seiten, Funktionen, Bilder, Dokumente etc.
- **Eine Ressource kann verschiedene Repräsentationen haben (z.B. XML, JSON)**
- Über eine URL eindeutig identifizierbar und lokalisierbar
- Zugriff über vier HTTP-Methoden (ggf. weitere Methoden) ☐ Web Services

Navigation in einer REST-Architektur

- Eine **Repräsentation** kann **Links** enthalten, die auf weitere Ressourcen verweisen
- **Durch Navigation** zu einer anderen Ressource (bzw. zu einer anderen Repräsentation) **wechselt Client seinen Zustand**
- Status-Transfer über Repräsentationen
- **REST = REpresentational State Transfer**

HTTP GET

- Bei Zugriff über HTTP GET erfolgt der Zugriff über eine eindeutige URL.
- **Web Service-Operation** besitzt:
 - URL
 - Operationsnamen
 - Parameterwerte.
- Dies ist auch der Grund, weshalb bei HTTP GET nur simple Datentypen wie *String* oder *int* verwendet werden können
- komplexe Datentypen lassen sich nur schlecht in Form einer URL ausdrücken.

HTTP Post

- Bei Verwendung von **HTTP POST** können sowohl einfache als **auch komplexe Datentypen** verwendet werden.
- Bei der Kommunikation entfallen SOAP-Envelope, SOAP Body und SOAP Header.
- Die Nutzdaten werden direkt übertragen.
- REST verzichtet also auf SOAP.

Beispiel

- Abfragen eines Kontostands:

GET http://meinebank.de/engagement/1234450

Als Ergebnis eines solchen *GETs* wird ein Objekt (z.B. eine XML-Darstellung des Hauptkontos) zurückgeliefert.

- Wie das Ergebnis einer Anfrage repräsentiert wird, ist bei REST nicht spezifiziert.
- Zwischen Client und Server muss ein gemeinsames Verständnis über die Bedeutung der Repräsentation vorhanden sein.

Beispiel

■ Mögliche Antwort: XML-Datei

```
<konto>
  <inhaber>Dagobert Duck</inhaber>
  <unterkonten>
    <unterkonto xlink:href="
      http://meinebank.de/engagement/1234450/30">
      Giro</unterkonto>
    <unterkonto xlink:href="
      http://meinebank.de/engagement/1234450/40">
      Spar</unterkonto>
  </unterkonten>
</konto>
```

Beispiel

Wählt der Benutzer als nächstes das Girokonto, so werden diese Informationen durch den Aufruf der folgenden URI geliefert:

GET <http://meinebank.de/engagement/1234450/30>

Beispiel

Das Ergebnis des Aufrufs ist die aktuelle Darstellung des spezifischen Kontos

```
<kontoinfo typ="giro">  
  <saldo>4000 H</saldo>  
  <funktion xlink:href="  
    http://meinebank.de/engagement/1234450/30/bewegungen>Bewegungen  
  </funktion>  
</kontoinfo>
```

REpresentational State Transfer - Ablauf

- Die Repräsentation einer Resource kann auf weitere Ressourcen verweisen.
 - Folgt ein Client einem Link in einer Repräsentation, so gelangt er von einem Zustand in einen anderen.
1. Ein Web Browser fordert eine Seite, oder allgemeiner eine Resource über eine URL an.
 2. Ein HTML Dokument, welches eine Repräsentation der Resource darstellt, wird vom Server zum Client übertragen.
 3. Das HTML Dokument kann Links enthalten, die auf weitere Ressourcen im Web verweisen.
 4. Navigiert der Client zu einer neuen Seite, so verändert er seinen Zustand, er wechselt oder macht einen *Transfer* zu einem neuen Zustand durch.
 5. Über Repräsentationen wird ein Transfer von einem Status in einen anderen Status durchgeführt.

Nachrichten

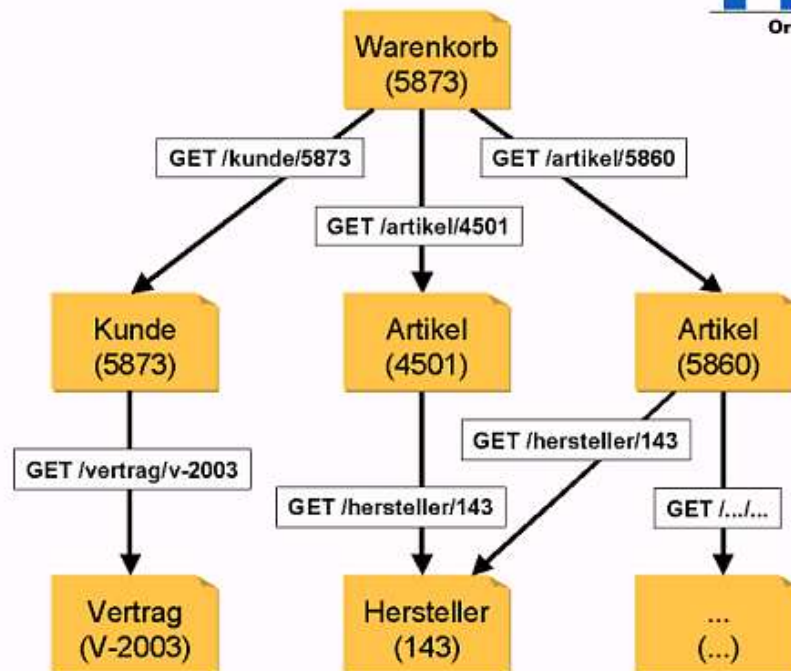
- Sämtliche Dokumenttypen können in REST Anwendungen übertragen werden.
- Für die Übertragung von strukturierten Daten eignet sich XML. XML Dokumente können XLink für Verweise benutzen.
- Nachrichten sind in REST selbstbeschreibend.
 - In einer Nachricht muss alles enthalten sein, um die Nachricht zu interpretieren.
 - Für die Interpretation einer Nachricht ist kein Wissen über vorherige oder spätere Nachrichten notwendig.
 - Der Status einer Anwendung wird durch den Inhalt einer oder mehrerer Hypertext Dokumente repräsentiert.

Onlineshop Beispiel einer REST-Anwendung

- In der Anwendung gibt es Kunden, die Artikel in Warenkörbe aufnehmen können.
- Jedes einzelne Objekt der Anwendung wie Artikel oder Kunde stellt eine Resource dar, die extern über eine URL erreichbar ist.
- Mit dem folgenden Aufruf ist in der Beispielanwendung der Warenkorb mit der Nummer 5873 erreichbar.

GET /warenkorb/5873

Beispiel: Ausschnitt eines OnlineShop

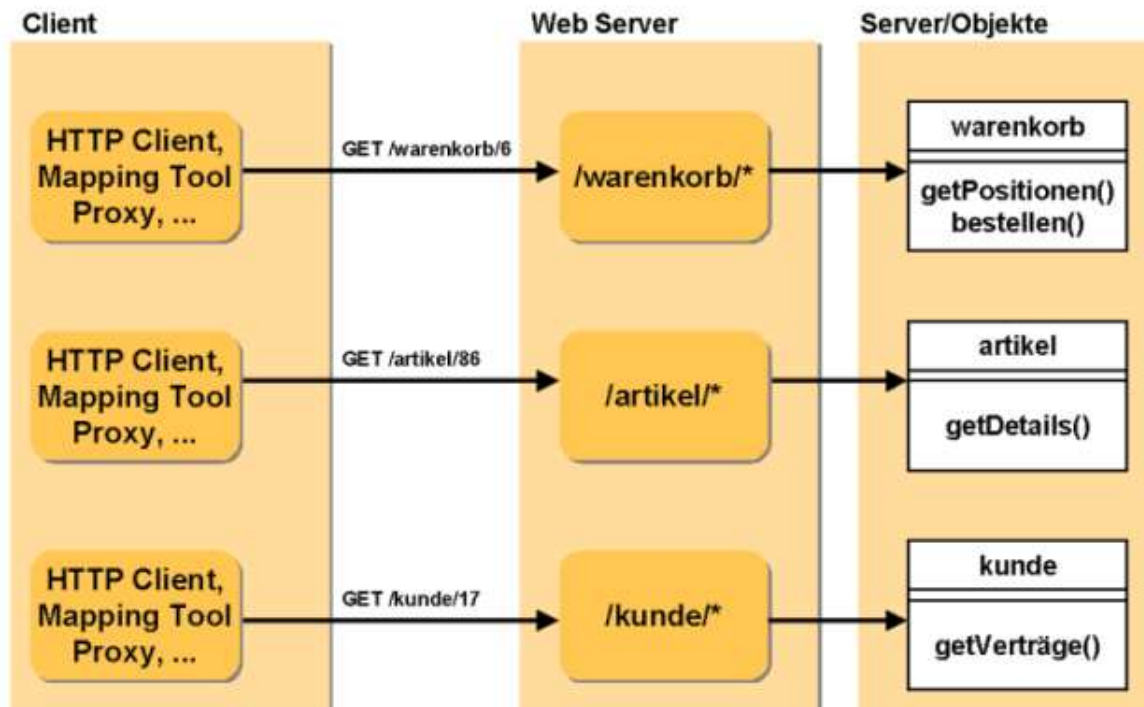


RESTful Services - POST

- Die folgende POST Anfrage fügt dem Warenkorb den Artikel mit der Artikelnummer 961 hinzu.
- `POST /warenkorb/5873 artikelnummer=961`

Ressourcenzugriff

Adressierung von Objekten



Der Standard und die Implementierungen

- In JSR-311 definiert
- In jedem Applikationsserver ab Java 6
- JAX-SR Referenzimplementierungen
- ***Jersey***
 - ermöglicht ein REST Webservice zusammen mit Tomcat oder in Java eingebauten Mini-HTTP-Server
- **RESTEasy** is the JBoss project that provides JAX-RS implementation.

Download und JAR Dateien

- Beispiel aus Java Insel
- <http://jersey.java.net/>
- jersey-bundle.jar
- JSR311-api.jar
- asm.jar (Maven)

Mein erstes REST-Service

```
package test.rest;
import javax.ws.rs.*;
@Path( "message" )
public class MessageResource
{
    @GET
    @Produces( MediaType.TEXT_PLAIN )
    public String message()
    {
        return "hallo! ";
    }
}
```

Mein erstes REST-Service

■ ■ @Path

- Die Pfadangabe, die auf den Basispfad gesetzt wird.

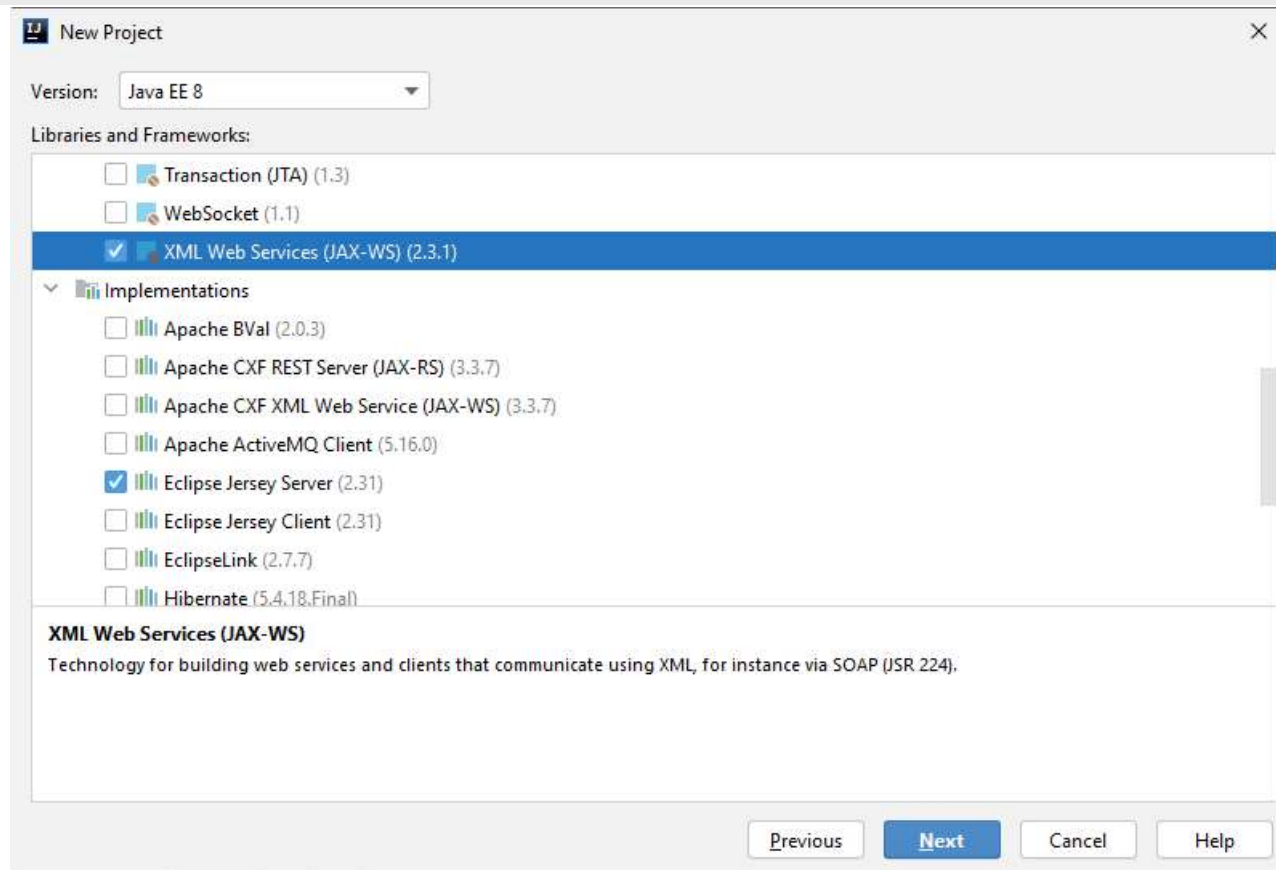
■ ■ @GET

- Die HTTP-Methode. Hier gibt es auch die Annotationen für die anderen HTTP-Methoden POST, PUT, ...

■ ■ @Produces

- MIME-Typ setzen, z.B.: MediaType.TEXT_XML oder MediaType.TEXT_HTML, und auch Strings wie »application/pdf« können direkt gesetzt werden.
- kann auch entfallen

IntelliJ + Tomcat



REST mit Tomcat und Maven

```
package testrest;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

@ApplicationPath("/")
public class RESTApplication extends Application {

}
```

REST mit Tomcat und Maven

```
<dependency>
  <groupId>org.glassfish.jersey.containers</groupId>
  <artifactId>jersey-container-servlet</artifactId>
  <version>2.29.1</version>
</dependency>
<dependency>
  <groupId>org.glassfish.jersey.inject</groupId>
  <artifactId>jersey-hk2</artifactId>
  <version>2.29.1</version>
</dependency>
```

REST mit Tomcat und Maven

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

<servlet-mapping>
  <servlet-name>Jersey REST Service</servlet-name>
  <url-pattern>/*</url-pattern>
</servlet-mapping>
```