Thomas Bayer und Dirk M. Sohn

REST Web Services

Eine Einführung

Web Services werden meist mit SOAP in Verbindung gebracht. Mit REpresentational State Transfer oder kurz REST, einem Architekturstil, können aber ebenfalls Web Services realisiert werden. Dieser Artikel beschreibt REST anhand eines Beispiels und erläutert die wichtigsten Konzepte.

Neben SOAP gibt es eine weitere Alternative für die Realisierung von Web Services. Thomas Roy Fielding beschreibt in seiner Dissertation einen Architekturstil, den er „REpresentational State Transfer Architektur“ oder kurz REST nennt [1]. REST basiert auf Prinzipien, die in der größten verteilten Anwendung eingesetzt werden – dem World Wide Web. Das WWW stellt selbst eine gigantische REST-Anwendung dar. Viele Suchmaschinen, Shops oder Buchungssysteme sind ohne Absicht bereits als REST-basierter Web Services verfügbar. REST ist kein Produkt oder Standard. REST beschreibt, wie Webstandards in einer webgerechten Weise eingesetzt werden können.

Beispiel einer REST-Anwendung

Ein Onlineshop soll als Beispiel für eine RESTful-Anwendung dienen. In der Anwendung können Kunden Artikel einem Warenkorb zuordnen. Jedes einzelne Objekt der Anwendung wie Artikel oder Kunde stellt eine Ressource dar, die extern über eine URL erreichbar ist. Mit dem folgenden Aufruf ist der Warenkorb mit der Nummer 5873 erreichbar:

Aufruf

```
GET /warenkorb/5873
```

Listing 1

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. Die Verwendung von XML macht es leicht, die Repräsentation sowohl für Menschen als auch für Maschinen verständlich zu gestalten. Das Ergebnis der Warenkorbabfrage könnte wie folgt aussehen:

Repräsentation eines Warenkorbs

```
HTTP/1.1 200 OK
Content-Type: text/xml
<?xml version="1.0"?>
<warenkorb xmlns:xlink="http://www.w3.org/1999/xlink">
  <kunde xlink:href="http://shop.oio.de/kunde/5873">5873</kunde>
  <position nr="1" menge="5">
    <artikel xlink:href="http://shop.oio.de/artikel/4501" nr="4501">
      <beschreibung>Dauerlutscher</beschreibung>
    </artikel>
  </position>
  <position nr="2" menge="2">
    <artikel xlink:href="http://shop.oio.de/artikel/5860" nr="5860">
      <beschreibung>Earl Grey Tea</beschreibung>
    </artikel>
  </position>
</warenkorb>
```

Listing 2

Die Antwort des Servers ist ein XML-Dokument, das mit zahlreichen XML-Standards kompatibel ist und weiterverarbeitet werden kann. Der Warenkorb enthält zwei Positionen und den zugehörigen Kunden. Die Positionen verweisen mittels XLink auf weitere Ressourcen, die Artikel. Der Client kann einen Link verfolgen und die Repräsentation eines Artikels anfordern. Er wechselt auf diese Weise von einem Status in einen anderen.

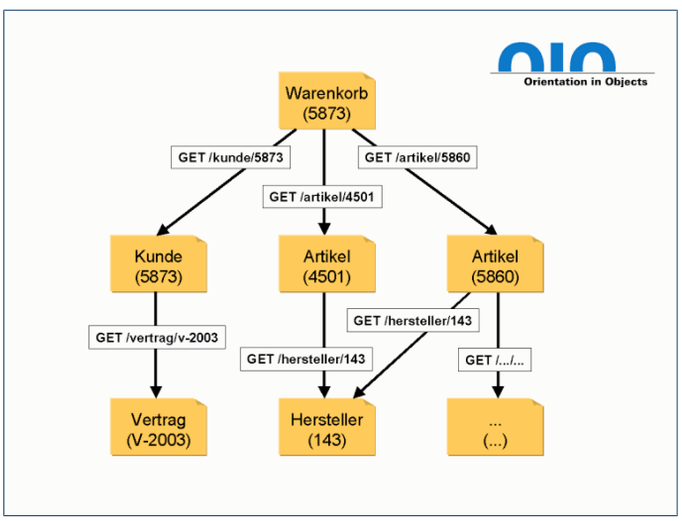
Aufruf

```
GET /artikel/5860
```

Listing 3

Artikelwechsel mittels XLink

Mit jedem Dokument kann der Client tiefer in die Anwendung einsteigen. Als Einstiegspunkt reicht eine einzige URL aus. Die Idee der aus dem Web bekannten Hypertext-Dokumente wird so auf Web Services übertragen. Das Navigieren von Ressource zu Ressource über den Webbrowser können Sie mit dem auf sqlREST basierenden Demo REST-Service [2] online ausprobieren.



Über Verweise kann sich der Client von Ressource zu Ressource durchhangeln.

Bestellen – oder Operationen auf dem Server auslösen

In einem guten Schwarztee darf Kandiszucker nicht fehlen. Für das Hinzufügen einer untergeordneten Ressource oder den Aufruf von Logik, die auf dem Server eine Statusänderung hervorruft, wird die HTTP-Methode POST verwendet. Die folgende POST-Anfrage fügt dem Warenkorb den Artikel Kandiszucker mit der Artikelnummer 961 hinzu:

Aufruf

```
POST /warenkorb/5873
artikelnummer=961
```

Listing 4

POST-Anfragen können mit einem kleinen Skript, beispielsweise in Perl, Ruby oder Groovy, abgeschickt werden. POST- oder PUT-Anfragen können auch ganz einfach mit RESTGate [3], einem Web-REST-Gateway, abgesetzt werden.

Anlegen von neuen Ressourcen

Für das Anlegen neuer Ressourcen, die nicht anderen Ressourcen zugeordnet sind, kann die HTTP-Methode PUT verwendet werden.

PUT Anfrage

```
PUT /artikel
<artikel>
 <beschreibung-kurz>Rooibusch Tee</beschreibung-kurz>
 <beschreibung>
 Feiner namibischer Rooibusch Tee
 </beschreibung>
 <preis>2,80</preis>
 <einheit>100g</einheit>
</artikel>
```

Listing 5

Listing 5 zeigt, wie ein neuer Artikel angelegt werden kann.

Antwort auf einen PUT

```
HTTP/1.1 201 OK
Content-Type: text/xml;
Content-Length: 30
Location: http://shop.oio.de/artikel/6005
```

Listing 6

Der HTTP-Status-Code 201 in der Antwort des Servers bedeutet „created", das heißt eine neue Ressource wurde angelegt. Im Location-Header steht ein Verweis auf die neu erzeugte Ressource, den Rooibusch-Tee.

Löschen von Ressourcen

Die neu angelegte Ressource kann mit der folgenden HTTP-DELETE-Anfrage wieder gelöscht werden.

Anfrage

```
DELETE /artikel/6005
```

Listing 7

Das Beispiel hat gezeigt, wie mit Ressourcen, HTTP-Methoden und URLs gearbeitet werden kann. Die folgenden Abschnitte vertiefen die verwendeten Konzepte.

Ressourcen

Webseiten, Bilder und CGI-Skripte bzw. Servlets stellen Ressourcen dar, die über URLs adressiert und angesprochen werden können. Eine Webanwendung ist also nichts anderes als eine Ansammlung von Ressourcen. Mit HTTP können Nachrichten an die Ressourcen gesendet werden, beispielsweise durch den Aufruf einer Seite im Browser. Eine direkte Manipulation einer Ressource ist nicht vorgesehen. Jeder Zugriff erfolgt indirekt über die der Ressource zugeordnete URL.

Die Semantik des HTTP-Protokolls wurde in REST übernommen. Eine zentrale Rolle bei REST spielen die HTTP-Methoden GET, PUT, POST und DELETE. Sie stellen die „Verben" dar, die auf „Hauptwörter" bzw. Ressourcen angewendet werden können.

Repräsentationen

Die Repräsentation einer Ressource kann auf weitere Ressourcen verweisen. Folgt ein Client einem Link in einer Repräsentation, so gelangt er von einem Zustand in einen anderen.

Weshalb die Bezeichnung „REpresentational State Transfer" verwendet wird, macht das folgende Szenario deutlich: Ein Webbrowser fordert eine Seite oder allgemeiner eine Ressource über eine URL an. Ein HTML-Dokument, das eine Repräsentation der Ressource darstellt, wird vom Server zum Client übertragen. Dieses Dokument kann Links enthalten, die auf weitere Ressourcen im Web verweisen. Navigiert der Client zu einer neuen Seite, verändert er seinen Zustand, er wechselt oder macht einen Transfer zu einem neuen Zustand durch. Über Repräsentationen wird ein Transfer von einem Status in einen anderen Status durchgeführt.

Die Methoden

Das Interface von REST ist generisch. Es müssen keine Protokoll-Konventionen bekannt sein, damit Client und Server sich verständigen können. Die folgende Tabelle beschreibt die Bedeutung der HTTP-Methoden, wie sie von REST verwendet werden.

GET	GET fragt die Repräsentation einer Ressource ab. Requests sollten frei von Seiteneffekten sein. GET-Requests können beliebig oft abgeschickt werden. Man kann einen Client für seine Auswirkungen nicht in die Verantwortung ziehen, d.h. ein GET kann bedenkenlos abgeschickt werden.
POST	Mit POST kann einer Ressource etwas hinzugefügt werden – beispielsweise eine Ware zu einem Warenkorb. POST ist nicht frei von Seiteneffekten. Beispielsweise können durch einen POST-Aufruf Felder in einer Datenbank verändert oder Prozesse auf dem Server gestartet werden.
PUT	Neue Ressourcen können mit PUT erzeugt oder der Inhalt bestehender Ressourcen kann mit PUT ersetzt werden.
DELETE	Ressourcen können mit DELETE gelöscht werden.

Jede REST-Ressource besitzt über die HTTP-Methoden GET, POST, PUT und DELETE eine generische Schnittstelle. Mit diesen vier Methoden können die meisten Anwendungsfälle abgedeckt werden.

Nachrichten

Sämtliche Dokument-Typen können in REST-Anwendungen übertragen werden. Beispielsweise werden im Web unter anderem HTML-, GIF- und PDF-Dateien verwendet. Für die Übertragung von strukturierten Daten eignet sich XML. Wer eine Anwendung mit REST realisieren möchte, muss kein neues Format erlernen, sondern kann bereits bekannte Formate verwenden.

Nachrichten sind in REST selbstbeschreibend. In einer Nachricht muss alles Notwendige zur Interpretation enthalten sein, es ist kein Wissen über vorherige oder spätere Nachrichten notwendig. Der Status einer Anwendung wird durch den Inhalt einer oder mehrerer Hypertext-Dokumente repräsentiert.

Status und Session

Der Server kennt seinen Status. Für den Client-Status oder Sessions zu seinen Clients interessiert er sich nicht. Der Client verwaltet seinen Status selbst. Er entscheidet auch über die Reihenfolge, in der er verschiedene Methoden auf dem Server aufruft.

In REST-Anwendungen wird meist keine spezielle Funktionalität für das Login benötigt. Alle Ressourcen lassen sich mit verfügbaren Webtechnologien, wie zum Beispiel HTTP und HTTPS, authentifizieren und autorisieren.

Die folgenden Merkmale kennzeichnen den REST-Stil:

- Die Kommunikation erfolgt auf Abruf. Der Client ist aktiv und fordert vom passiven Server eine Repräsentation an bzw. modifiziert eine Ressource.
- Ressourcen, die „Objekte" der Anwendung, besitzen eine ihnen zugeordnete URL, mit der sie adressiert werden können.
- Die Repräsentation einer Ressource kann als Dokument vom Client angefordert werden.
- Repräsentationen können auf weitere Ressourcen verweisen, die ihrerseits wieder Repräsentationen liefern, die wiederum auf Ressourcen verweisen können.
- Der Server verfolgt keinen Client-Status. Jede Anfrage an den Server muss alle Informationen beinhalten, die zum Interpretieren der Anfrage notwendig sind.
- Caches werden unterstützt. Der Server kann seine Antwort als Cache-fähig oder nicht Cache-fähig kennzeichnen.

Vorteile von REST

Neben der einfachen Architektur und der Möglichkeit, bestehende Technologien zu verwenden bietet REST folgende Vorteile:

Skalierbarkeit
Das enorme Wachstum des World Wide Web hat gezeigt, in welchem Ausmaß Webtechnologien skalieren. Millionen Benutzer verwenden Ressourcen, die von vielen tausend Servern angeboten werden. Proxys und Caches erhöhen die Performance. Komponenten wie Server, Proxys und Webanwendungen können einzeln installiert und gewartet werden.

Es gibt eine Fülle von Formaten, von HTML und SVG bis AVI. Selbst neue Formate können über MIME-Types leicht hinzugefügt werden. Die Größe der übertragenen Dokumente schwankt von wenigen Bytes bis zu vielen Megabytes.

Im Web werden mit Cookies und URL-Rewriting auf das statuslose HTTP künstlich Sessions aufgesetzt, die stateful sind. In diesem Punkt weicht REST vom Web ab. Interaktionen sind in REST stateless – jede Operation steht für sich. Alle notwendigen Informationen sind in den Repräsentationen der Ressourcen enthalten. Mit HTTP gibt es keine Grenzen zwischen den Anwendungen. Durch die Verfolgung eines Links kann man, ohne es zu beabsichtigen, zu einer völlig anderen Anwendung gelangen. Da alle Interaktionen statuslos sind, muss bei einem Wechsel von einem Server zu einem anderen kein Status zwischen den Servern propagiert werden. Dies hat positive Auswirkungen auf die Skalierbarkeit einer Anwendung.

Anbindung von Fremdsystemen
In keiner anderen Anwendung sind so viele Legacy-Systeme wie im Web integriert. Über verschiedene Gateways kann auf eine Fülle von Systemen zugegriffen werden. Die Details von Fremdsystemen werden hinter Schnittstellen versteckt.

Unabhängig installierbare Komponenten
Für Komponenten kann in einer REST-Anwendung unabhängig voneinander das Deployment durchgeführt werden, ähnlich wie im Web, wo auch der Inhalt einzelner Seiten ausgetauscht werden kann, ohne weitere Seiten anzupassen. Für sehr große Systeme, wie z. B. das World Wide Web oder den E-Mail-Dienst im Internet, ist ein unabhängiges Deployment eine Grundvoraussetzung.

Komposition von Diensten
Einzelne REST-Services können leicht zusammen genutzt werden. Genau genommen gibt es gar keine REST-Services. Es gibt nur Ressourcen, die angeboten werden. Über den globalen und universellen Adressraum der URLs können die Grenzen einer Anwendung leicht überschritten werden. Ein Dokument verweist einfach auf eine Ressource, die sich in einer anderen Organisation befindet.

Wie erstellt man eine REST-Anwendung?

Verteilte Anwendungen basieren auf Funktionen oder Methoden, die auf Objekten arbeiten. Der Client kann über entfernte Methoden auf die Objekte des Servers zugreifen. Eine REST-Anwendung macht Ihre Objekte über URLs nach außen sichtbar. Geschäftsobjekte müssen über eine URL erreichbar sein und mit einem Dokument, vorzugsweise in XML, repräsentiert werden können.

Der Kern eines REST-Servers unterscheidet sich nicht von einer RPC-Anwendung. Der Unterschied liegt in der Schnittstelle, die sich bei REST grundlegend von einer RPC-Anwendung unterscheidet. REST verwendet die HTTP-Methoden, die auf über URL adressierbaren Ressourcen arbeiten, während RPC ein komponentenbasiertes Interface mit spezialisierten Methoden oder Funktionen zur Verfügung stellt.

Toolunterstützung

REST basiert auf eingeführten Standards, für die es bereits eine Unzahl von Tools gibt. REST-fähig sind Webserver wie Apache oder IIS, Webcontainer wie Tomcat und Proxy-Server wie Squid. Spezialisierte REST-Tools sind nicht notwendig.

Es gibt bereits eine Reihe von öffentlichen Web Services, die REST verwenden. Zu den bekannteren gehören Web Services von Amazon, Google oder FlickR. Einige Dienste im Web, wie beispielsweise Wikis, sind von Haus aus zumindest teilweise REST-konform.

Fazit

REST verwendet ausschließlich reife und standardisierte Standards wie HTTP oder URLs, die bereits seit einigen Jahren verfügbar sind. REST macht sich die Ideen des Webs wie Hypertext und einen unbegrenzten, globalen Adressraum zunutze. Damit sind REST-basierte Web Services wirkliche Web Services – im Gegensatz zu RPC-basierter Middleware.

Links und Literatur

Softlink 1577

[1] Thomas Roy Fielding, Architectural Styles and the Design of Network-based Software Architectures: http://www.ics.uci.edu/%7Efielding/pubs/dissertation/top.htm

[2] REST Service: http://www.thomas-bayer.com/rest-demo.htm

[3] RESTGate: http://thomas-bayer.com/restgate/

DER AUTOR

Thomas Bayer (www.thomas-bayer.com) ist Berater und Trainer für serviceorientierte Architekturen und Web 2.0. Er bietet Schulung, Entwicklung und Beratung für Softwarearchitekten, Entwickler und Projektleiter, die ihr Wissen auf den neuesten Stand bringen möchten. Sein Zweitberuf ist Heilpraktiker.

DER AUTOR

Dirk M. Sohn ist Geschäftsführer der Orientation in Objects GmbH (http://oio.de). Er verfügt über 15 Jahre Erfahrung in der objektorientierten Softwareentwicklung. Sein Schwerpunkt liegt auf dem Management von Java-Enterprise-Projekten sowie auf Business Intelligence.