trivadis
Part of Accenture

REST Web Services
Eine Einführung

) Schulung)

) Beratung)

) Entwicklung)

) Artikel)

AUTOR

Thomas Bayer
Orientation in Objects GmbH

Veröffentlicht am: 27.11.2002

ABSTRACT

Web Services werden meist mit SOAP oder XML-RPC in Verbindung gebracht. Mit REpresentational State Transfer oder kurz REST, einem Architekturstil, können ebenfalls Web Services realisiert werden. Dieser Artikel beschreibt REST anhand eines Beispiels und erläutert Aspekte wie Sicherheit und Skalierbarkeit. Unterschiede zu einer RPC Middleware werden mit einem Vergleich zwischen REST und SOAP aufgezeigt.

Trivadis Germany GmbH

Weinheimer Str. 68
D-68309 Mannheim

Tel. +49 (0) 6 21 - 7 18 39 - 0
Fax +49 (0) 6 21 - 7 18 39 - 50

www.oio.dekontakt@trivadis.com

EINLEITUNG

Neben SOAP und XML-RPC 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.

REST basiert auf Prinzipien, die in der größten verteilten Anwendung eingesetzt werden - dem World Wide Web. Das World Wide Web stellt selbst eine gigantische REST Anwendung dar. Viele Suchmaschinen, Shops oder Buchungssysteme sind ohne Absicht bereits als REST basierter Web Services verfügbar.

Die *REpresentational State Transfer Architektur* ist ein Architektur Modell, welches beschreibt, wie das Web funktionieren sollte. Das Modell soll als Anleitung und als Referenz für zukünftiger Erweiterungen dienen.

REST ist kein Produkt oder Standard. REST beschreibt, wie Web Standards in einer Web gerechten Weise einsetzt werden können.

1 BEISPIEL EINER REST ANWENDUNG

Ein Onlineshop soll als Beispiel für eine RESTful Anwendung dienen. 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 1: Aufruf Warenkorb #5873

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 Repräsenation sowohl für Menschen als auch für Maschinen verständlich zu gestalten. Das Ergebnis der Warenkorbabfrage könnte wie folgt aussehen:

```
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>
```

Beispiel 2: Repräsentation eines Warenkorbes

Die Antwort des Servers enthält wie aus Listing 1. ersichtlich ein XML Dokument, welches von Natur aus mit zahlreichen XML Standards kompatibel ist und weiterverarbeitet werden kann. Die Anwort kann mittels einer XSLT Transformation beispielsweise in HTML, SVG oder PDF umgewandelt werden. Das Dokument kann auf weitere Resourcen mit XLink und XPointer verweisen. Mit XPath oder XQuery können Abfragen an das Dokument formuliert werden.

Der Warenkorb enthält zwei Positionen und den zugehörigen Kunden. Die Positionen verweisen mittels XLink auf weitere Resourcen, die Artikel. Der Client kann einen Link verfolgen und die Repräsentation eines Artikels anfordern. Er welchselt auf diese Weise von einem Status in einen anderen.

```
GET /artikel/5860
```

Beispiel 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 auf Web Services übertragen. Abbildund 1. zeigt einen Graphen, der sich durch die Verzeigerung aufspannt. Der Client kann sich über Verweise von Resource zu Resource durchhangeln.

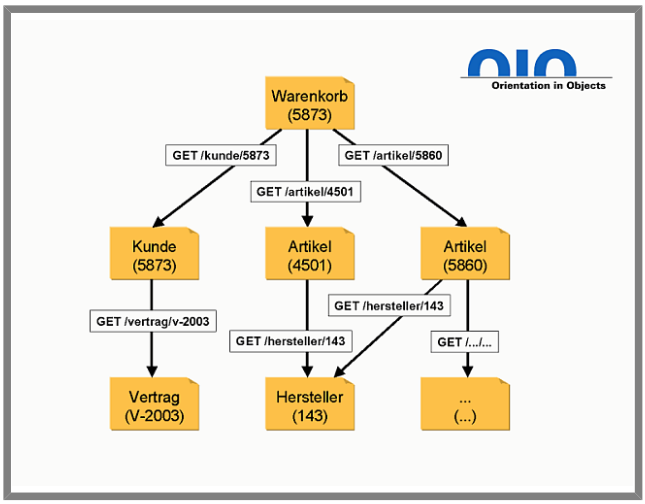

Abbildung 1: Navigation über Links u. HTTP GET

1.1 BESTELLEN - ODER OPERATION AUF DEM SERVER AUSLÖSEN

In einem guten Schwarztee darf Kandis Zucker nicht fehlen. Die Bestellung muss um diesen wichtigen Artikel erweitert werden. Angenommen Kandis Zucker hätte die Artikelnummer 961. Für das Hinzufügen einer untergeordneten Resource oder den Aufruf von Logik, die auf dem Server eine Statusänderung hervorruft wird die Methode POST verwendet. Die folgende POST Anfrage fügt dem Warekorb den Artikel Kandis Zucker mit der Artikelnummer 961 hinzu.

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

Beispiel 4: POST Anfrage

1.2 ANLEGEN VON NEUEN RESOURCEN

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

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

Beispiel 5: PUT Anfrage

Das Ergebnis des PUT Aufrufes liefert über HTTP eine URL zum neu angelegten Artikel. Mit dem Anlegen wurde ein neues Objekt erzeugt, dass von weiteren Clients verwendet werden kann.

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

Beispiel 6: Antwort auf einen PUT

Listing 3. zeigt die Antwort des Servers. Der *HTTP Status Code* 201 bedeutet "created", d.h. eine neue Resource wurde angelegt. Im Body der Nachricht steht ein Verweis auf die neu erzeugte Resource, den Rooibusch Tee.

1.3 LÖSCHEN VON RESOURCEN

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

```
DELETE /artikel/6005
```

Beispiel 7: Löschen von Resourcen

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

2 DIE WELT VON REST

2.1 RESOURCEN

Web Seiten, Bilder und CGI Skripte bzw. Servlets stellen Resourcen dar, die über URLs adressiert und angesprochen werden können. Eine Web Anwendung stellt eine Ansammlung von Resourcen dar. Mit HTTP können Nachrichten an die Resourcen gesendet werden, beispielsweise durch den Aufruf einer Seite im Browser.

Eine direkte Manipulation einer Resource ist nicht vorgesehen. Jeder Zugriff erfolgt indirekt über die der Resouce zugeordnete URI.

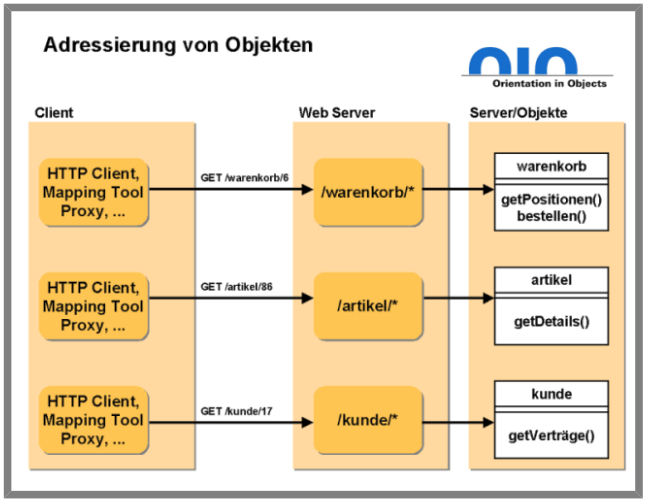

Abbildung 2: Routing von REST Nachrichten

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. Resourcen, angewandet werden können. Mit diesem begrenzten Vorrat von vier Methoden müßten alle Anwendungsfälle generisch abgedeckt werden können.

2.2 REPRÄSENTATIONEN

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

Weshalb die Bezeichnung *REpresentational State Transfer* verwendet wird, wird aus dem folgendem Szenario deutlich. Ein Web Browser fordert eine Seite, oder allgemeiner eine Resource über eine URL an. Ein HTML Dokument, welches eine Repräsentation der Resource darstellt, wird vom Server zum Client übertragen. Das HTML Dokument kann Links enthalten, die auf weitere Resourcen im Web verweisen. Navigiert der Client zu einer neuen Seite, so verändert er seinen Zustand, er welchselt oder macht einen *Transfer* zu einem neuen Zustand durch. Über Repräsentationen wird ein Tranfer von einem Status in einen anderen Status durchgeführt.

2.3 DIE METHODEN

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

- **GET**: Get fragt die Repräsentation einer Resource 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 Resource etwas hinzugefügt werden. Beispielsweise könnte eine Ware zu einem Warenkorb hinzugefügt werden. 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 Resourcen können mit PUT erzeugt oder der Inhalt bestehender Resourcen kann mit PUT ersetzt werden.
- **DELETE**: Resourcen können mit DELETE gelöscht werden.

Jede REST Resource besitzt über die HTTP Methoden GET, POST, PUT und DELETE eine generische Schnittstelle. Mit diesen vier Methoden können die meisten Anwendungsfälle abgedeckt werden. Viele Anwendungen, die SQL verwenden benutzen auch nur die generischen Befehle SELECT, INSERT, UPDATE und DELETE.

2.4 NACHRICHTEN

Sämtliche Dokument Typen können in REST Anwendungen übertragen werden. Beispielsweise werden im Web u.a. HTML, GIF und PDF Dateien verwendet. Für die Übertragung von strukturierten Daten eignet sich XML. XML Dokumente können XLink für Verweise benutzen. Wer eine Anwendung mit REST realisieren möchte muss kein neues Format erlernen. Man kann bereits bekannte Formate verwenden.

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.

2.5 STATUS UND SESSION

Der Server kennt seinen Status. Für den Clientstatus 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 den Login benötigt. Alle Resourcen lassen sich mit verfügbaren Web Technologien wie zum Beispiel HTTP und HTTPS authentifizieren und autorisieren.

3 REST UND SICHERHEIT

Die Vision von CORBA als galaktischer Objektbus hat sich nicht wie von seinen Verfechtern prophezeit erfüllt. Der Grund lag neben der Komplexität von CORBA daran, dass Firmen- und Organisationsnetze zunehmend durch Firewalls geschützt wurden. Das von CORBA verwendete Protokoll IIOP wird von den meisten Firewalls geblockt.

Die Tatsache, daß die meisten Firewalls HTTP für das Surfen im Web zulassen wird ausgenutzt, um beliebige Nachrichten über Firewalls zu übertragen. SOAP und XML-RPC verwenden das Anwendungsprotokoll HTTP als reines Transportprotokoll, indem Sie Anfragen und Antworten in HTTP einpacken. Mit dem *Tunneling* ist eine Objektkommunikation trotz Firewalls wieder möglich.

Für die bestehenden Firewalls und Administatoren sehen alle SOAP oder XML-RPC Nachrichten gleich aus. Es handelt sich um einen HTTP Post. Um die Bedeutung einer Nachricht zu verstehen, muß der SOAP Body betrachtet werden, der nichts mit HTTP und den Dingen zu tun hat, die ein Administrator oder eine Firewall im Normalfall kennen.

Eine REST basierte Anwendung verwendet für seine Nachrichten HTTP Methoden und URLs. Firewalls sind mit HTTP und URLs vertraut. Sie können nach Methoden und URLs filtern. Zum Beispiel läßt sich über eine Firewall der Zugriff auf eine REST Anwendung auf einen Lesezugriff beschränken, indem nur GET Anfragen von Externen erlaubt werden.

Die Bedeutung einer Nachricht geht in einer REST Anwendung aus den HTTP Anfragen hervor. Das Zugriffsprotokoll eines Web Servers verrät, welche Aktionen durchgeführt wurden. Aus dem Protokoll in Listing 4. geht hervor, dass der Warenkorb 6 und die Artikel 5 und 12 ausgelesen wurden. Mit dem folgenden POST wurde wahrscheinlich dem Warenkorb eine Ware hinzugefügt. Anschließend wird der Warenkorb bestellt und die neu erzeute Bestellung ausgelesen.

```
hermes.oio.de - - [26/Nov/2002:12:43:07 +0100]
   "GET /warenkorb/6 HTTP/1.1" 200
hermes.oio.de - - [26/Nov/2002:12:43:08 +0100]
   "GET /artikel/12 HTTP/1.1" 200
hermes.oio.de - - [26/Nov/2002:12:43:08 +0100]
   "GET /artikel/5 HTTP/1.1" 200
hermes.oio.de - - [26/Nov/2002:12:43:09 +0100]
   "POST /warenkorb/6 HTTP/1.1" 200
hermes.oio.de - - [26/Nov/2002:12:43:13 +0100]
   "POST /warenkorb/6 HTTP/1.1" 200
hermes.oio.de - - [26/Nov/2002:12:43:14 +0100]
   "GET /bestellung/3 HTTP/1.1" 200
```

Beispiel 8: Protokoll einer REST Anwendung

In SOAP sind alle wesentlichen Parameter in der Nachricht kodiert. Ein Firewall Administrator hat wenig Einfluß und bleibt meist außen vor. Es besteht die Gefahr, dass SOAP zukünftig in einigen Firewallinstallationen ausgefiltert werden wird. Mit REST basierten Anwendungen können Administratoren leichter eingebunden werden. Sie bekommen die Möglichkeit mit Ihren Mitteln und Tools zu Protokollieren und zu Filtern. Die Bereitschaft REST nicht zu blockieren ist damit wesentlich größer. Ein generelles Blockieren von REST ist auch nicht möglich, es sei denn, der Zugriff auf das Web wird komplett gesperrt.

4 MERKMALE EINER REST ANWENDUNG

Die folgenden Merkmale kennzeichen den REST Stil.

- Die Kommunikation erfolgt auf Abruf. Der Client ist aktiv und fordert vom passiven Server eine Repräsentation an, bzw. modifiziert eine Resource.
- Resourcen, die Objekte der Anwendung, besitzen eine ihnen zugeordnete URI, mit der sie adressiert werden können.
- Die Representation einer Resource kann als Dokument vom Client angefordert werden.
- Repräsentationen können auf weitere Resourcen verweisen, die ihrerseits wieder Repräsentationen liefern, die wiederum auf Resourcen verweisen können.
- Der Server verfolgt keinen Clientstatus. 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.

5 VORTEILE VON REST

5.1 SKALIERBARKEIT

Das enorme Wachstum des World Wide Web hat gezeigt, in welchem Ausmaß Web Technologien skalieren. Millionen Benutzer verwenden Resourcen, die von vielen Tausend Servern angeboten werden. Proxys und Caches erhöhen die Performance. Komponenten wie Server, Proxys und Web Anwendungen 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 Session 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 Resourcen 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 daher 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.

5.2 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.

5.3 UNABHÄNGIG INSTALLIERBARE KOMPONENTEN

Für Komponenten kann in einer REST Anwendung unabhängig voneinander das Deployment durchgeführt werden. Im Web kann ja auch der Inhalt einzelner Seiten ausgetauscht werden, ohne weitere Seiten anzupassen. Für sehr große Systeme, wie das World Wide Web oder der eMail Dienst im Internet, ist ein unabhängiges Deployment eine Grundvoraussetzung.

5.4 KOMPOSITION VON DIENSTEN

Einzelne REST Services können leicht zusammen genutzt werden. Genau genommen gibt es keine REST Services. Es gibt nur Resourcen, die angeboten werden. Über den globalen und universellen Adressraum der URIs können die Grenzen einer Anwendung leicht überschritten werden. Eine Dokument verweist einfach auf eine Resource, die sich in einer anderen Organisation befindet.

6 VERGLEICH VON REST UND SOAP

Bei SOAP handelt es sich um eine *RPC Middleware* [1], die HTTP oder SMTP als Transportprotokoll und XML als Nachrichtenformat verwendet. Der Vollständigkeit halber muss erwähnt werden, dass mit *SOAP with Attachments* auch Dokumente übertragen werden können.

Die Zustellung von SOAP Nachrichten ist vergleichbar mit der Funktionsweise einer Hauspost. Die Briefkörbe der Mitarbeiter können nicht direkt adressiert werden. Alle Briefe werden bei der Hauspost angeliefert. Die Hauspost muss vor einer Zustellung die Briefe öffnen und entscheidet dann, aufgrund des Inhaltes, an wen die Nachricht weitergeleitet wird. In einer REST Anwendung sind die Briefkörbe direkt adressierbar. Mit der SOAP Spezifikation 1.2 nähert sich in diesem Punkt SOAP an REST an. Zukünftig können Methoden auch außerhalb des Bodys spezifiziert werden.

SOAP Nachrichten werden immer an einen Endpunkt adressiert, der von einem SOAP Router, oder auch Dispatcher genannt, implementiert wird. SOAP Router werden mit einem Servlet oder einem CGI Skript realisiert. Aus Abbildung 3 ist ersichtlich, dass alle Aufrufe zunächst an den Dispatcher gerichtet werden. Jede Nachricht ist ein *HTTP POST* an die selbe Adresse.

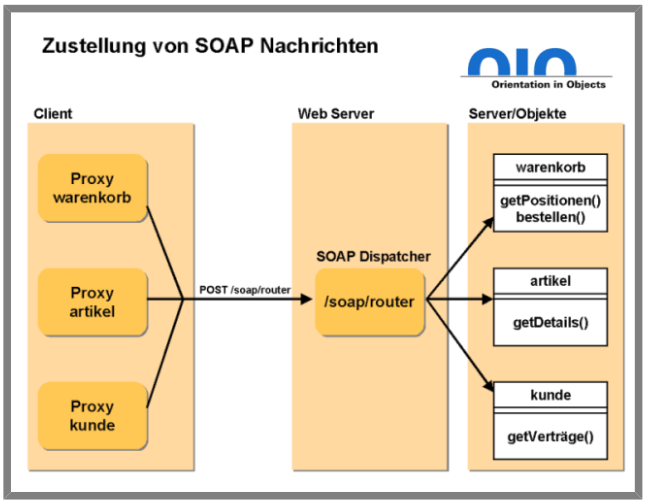

Abbildung 3: Routing von SOAP Messages

SOAP ist ein Protokollbaukasten, mit dem jeder Entwickler seine eigenen Anwendungsprotokolle entwickelt. Das Protokoll beschreibt den genauen Aufbau einer Anfrage und einer Antwort. Die Beschreibung der Anfrage und Antwort stellt einen starren Rahmen dar, aus dem nicht leicht ausgebrochen werden kann. Möchte der Server weitere Informationen als bisher liefern, so kann er das nicht über die bestehende Schnittstelle tun. Damit die bereits installierten Clients kompatibel bleiben, ist ein neuer Web Service notwendig. Mit REST kann der Server zusätzliche Informationen liefern und das Interface des Web Service kann unverändert bleiben. In SOAP fehlt die Möglichkeit zur schrittweisen Evolution bestehender Web Services.

Ein ganz entscheidender Unterschied zwischen REST und SOAP ist der Adressraum. REST bietet mit den URIs einen globalen Adressraum, über den jede Resource adressiert werden kann. Bei REST steht die Resource im Vordergrund. Einen Service im eigentlichen Sinne gibt es bei REST nicht.

6.1 PROXY SERVER

Eine RESTful Anwendung besteht aus einer Vielzahl von URLs, auf die die Clients zugreifen können. Wird über einen Proxy Server auf eine REST Anwendung zugegriffen, so kann ein normaler Proxyserver wie z.B. der beliebte Squid eine Entscheidung anhand der URL treffen, ob er einen Zugriff erlauben oder verhindern soll. Der Zugriff auf jedes Objekt kann im Proxy und im Web Server protokolliert werden. Da alle SOAP Anfragen meist über eine URL geroutet werden, hat ein Proxy oder Web Server keine Möglichkeit über die URL eine Entscheidung zu treffen. In RESTful Anwendungen kann beispielsweise der Zugriff auf alle URLs mit dem folgenden Muster verboten werden:

```
deny http://shop.oio.de/artikel/delete/*
```

Beispiel 9: Zugriff auf URL verbieten

Mit REST ist das *Cachen* von GET Aufrufen möglich. SOAP verwendet dagegen ausschließlich die POST Methode, die nicht *gecacht* wird.

6.2 GENERISCHE SCHNITTSTELLE

REST bietet mit den Methoden GET, POST, PUT und DELETE ein generisches Interface. In SOAP müssen alle Methoden für jede Anwendung selbst definiert werden. Die Entwicklung von generischen Tools und Diensten im Web wird dadurch behindert.

6.3 STANDARDS

REST empfiehlt die Verwendung von etablierten Standards. Dem Entwickler stehen beispielsweise die folgenden "Technologien" zur Verfügung:

- URIs für die Adressierung
- HTTP Methoden für den Zugriff
- XML, XHTML, HTML, PNG, ... für die Datenformate
- MIME Typen

Zentral sind die Standards für HTTP und URIs. Für die Formate der Repräsentationen können dagegen beliebige Formate eingesetzt werden. Auch zukünftige Technologien können verwendet werden.

Für SOAP wird jetzt erst eine Reihe von Standards wie WSDL, UDDI, WS-Security, WS-Transactions oder BPEL4WS neu geschaffen, die ausschließlich für SOAP gedacht sind. Bei den Standards für REST handelt es sich um "richtige" Web Standards, die für viele Zwecke, nicht nur für REST, eingesetzt werden können.

7 WIE ERSTELLT MAN EINE REST ANWENDUNG...

... oder macht eine bestehende Anwendung RESTful? 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 URIs 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 URI adressierbare Resourcen arbeiten, während RPC ein komponentenbasiertes Interface mit spezialisierten Methoden oder Funktionen zur Verfügung stellt.

8 REST UND DAS SEMANTISCHE WEB

Bei SOAP und REST muss der Client die Nachrichten des Servers zu interpretieren verstehen. Mit RDF als Nachrichtenformat und entsprechenden Interferenz-Maschinen wären Clients denkbar, die zur Laufzeit lernen können. Roger L. Costello schreibt in seiner Präsentation *Representational State Transfer* [2] :

... ‚dynamic learning of response data combined with dynamic reasoning of link traversals would yield a self-reasoning automata. This is the next step of the Web!

9 GRENZEN VON REST

REST verwendet für den Transport HTTP und transportiert ganze Repräsentationen von Objekten zum Client. Bei Internetanwendungen, die sowieso mit einer hohen Verzögerungszeit rechnen müssen ist dies in Ordnung. Zwischen spezialisierten lokalen Servern z.B. zwischen Application Server und Datenbank benötigt man effizientere Protokolle wie CORBA, RMI oder DCOM.

Das Serialisieren und Deserialisieren von und nach XML wird in REST nicht angesprochen. Der Programmierer muss sich selbst darum kümmern oder ein XML Binding Framework einsetzen.

REST ist ein Architekturstil, der für große Anwendungen mit unbekannter Anzahl von Anwendern und Objekten geeignet ist.

10 TOOLUNTERSTÜTZUNG

REST basiert auf eingeführten Standards, für die es bereits eine Unzahl von Tools gibt. REST fähig sind Web Server wie Apache oder IIS, Web Container wie Tomcat und Proxy Server wie Squid. Spezialisierte REST Tools sind nicht notwendig. Es ist durchaus denkbar, dass zukünftig Bibliotheken, Frameworks oder Tools die Entwicklung von REST Anwendungen gezielt unterstützen.

11 ÖFFENTLICHE REST SERVICES

Es gibt bereits eine Reihe von öffentlichen Web Services, die bewußt RESTful Schnittstellen verwenden. Zu den bekannteren gehören Web Services von Amazon, Google oder O'Reillys Meerkat.

Einige Dienste im Web wie beispielsweise Wiki Webs sind von Haus aus zumindest teilweise REST konform.

12 FAZIT

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

13 QUELLEN

- [1] SOAP, CORBA u. RMI für Einsteiger
  (http://www.oio.de/public/objektorientierung/rpc/rpc-corba-soap-funktion.htm)
- [2] Representational State Transfer
  Costello, Roger L.
  (http://www.xfront.com/REST.ppt)
- Architectural Styles and the Design of Network-based Software Architectures
  (http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm)
- Second Generation Web Services
  (http://www.xml.com/pub/a/2002/02/06/rest.html)
- REST and the Real World
  Prescod, Paul
  (http://www.xml.com/lpt/a/2002/02/20/rest.html)
- The Emperor's New Tags - The SOAP/REST Controversy
  Prescod, Paul
  (http://www.prescod.net/rest/soap_rest_short.ppt)
- A New Direction for Web Services
  Prescod, Paul
  (http://www.sys-con.com/xml/article.cfm?id=454)
- Building Web Services the REST Way
  Costello, Roger L.
  (http://www.xfront.com/REST-Web-Services.html)
- Hypertext Transfer Protocol -- HTTP/1.1
  , IETF - The Internet Engineering Task Force
  (http://www.ietf.org/rfc/rfc2616.txt)