Mythos oder WahrheIT: RESTFUL WEBSERVICES

Sogenannte RESTful Webservices erfreuen sich seit Jahren zunehmender Verbreitung. Auch in der öffentlichen Verwaltung begegnet man diesem Ansatz immer wieder. Wird REST die klassischen SOAP Webservices mittelfristig verdrängen?

| von ANDREAS RAQUET

Der Begriff „REST“ basiert auf der Dissertation von Roy Thomas Fielding, einem der Autoren der http-Spezifikation. Darin hat Fielding die Prinzipien des WWW auf die Domäne der Softwarearchitektur angewandt. Das verwundert zunächst, versteht man Softwarearchitektur doch ansonsten in einem sehr viel kleineren Kontext. Fielding sah die Zukunft des Internets in einer weltweit verteilen Anwendungslandschaft, die organisch aus unzähligen Einzelanwendungen zusammenwächst – ähnlich wie das heutige Internet durch Verlinkung aus unzähligen Inhaltsangeboten organisch zusammengewachsen ist. Um das zu erreichen, so postulierte er, müssen die Anwendungen unabhängig voneinander entwickelt und organisationsübergreifend integriert werden sowie nahezu beliebig skalierbar sein. Heute anzutreffende RESTful Webservices orientieren sich an einigen der von Fielding aufgestellten Prinzipien, andere wiederum ignorieren sie vollständig. Wahr geworden ist Fieldings Vision der weltweiten Anwendungsarchitektur bisher nicht. Die meisten Unternehmen und Behörden zeigen wenig Interesse daran, ihre Anwendungen öffentlich zugänglich zu machen und „organisch“ untereinander zu vernetzen. Wenn das so ist, worin liegt dann die enorme Verbreitung von RESTful Webservices begründet?

Zunächst sind RESTful Webservices sehr einfach (siehe Infokasten „RESTful Webservices“). Sie verzichten auf statische Typisierung und formale Schemavalidierung und werden so viel zugänglicher als die sperrigen Protokolle rund um SOAP (siehe Infokasten „Die SOAP-Protokollfamilie“). Allerdings hat diese Einfachheit auch ihren Preis: Statische Typisierung und Schemavalidierung

tragen erheblich zur Robustheit von Softwaresystemen bei. Auch bei den derzeit bevorzugt genutzten Programmiersprachen und Datenbanken legt man mit gutem Grund Wert auf diese Eigenschaften. Darüber hinaus bieten die sogenannten WS-*Spezifikationen zusätzliche nützliche Vereinbarungen für SOAP-Webservices, wie Autorisierung, Ende-zu-Ende-Verschlüsselung, verteilte Transaktionen oder Kompensationsdienste, zu denen es bei REST keine Entsprechung gibt. Auf all das zu verzichten, nur um ein einfacheres Kommunikationsprotokoll zu ermöglichen, scheint ein sehr unausgewogener Kompromiss zu sein. Das gilt umso mehr, als die SOAP-Technologien mittlerweile gut durch Werkzeuge unterstützt sind und professionelle Softwareentwickler diese sehr gut beherrschen. Ist die Verbreitung von RESTful Webservices also nur das Ergebnis einer großen Fehleinschätzung?

Das Bild ändert sich, wenn man zwei weitere Trends in die Betrachtung einbezieht, die sich etwa zeitgleich mit RESTful Webservices entwickelt haben: Mobile Computing und Cloud Computing. Beide Trends haben unterschiedliche Plattformen von mehreren Herstellern hervorgebracht, die in intensiver Konkurrenz zueinander stehen. Der Erfolg dieser Plattformen hängt in hohem Maße von der Breite des Angebots und damit auch davon ab, wie viele Entwickler für die jeweilige Plattform tätig sind. Mit dem Erfolg der Plattform steigt die Nutzerzahl und damit die Anforderung an die Skalierbarkeit. Vor diesem Hintergrund kippt der Kompromiss zwischen SOAP und REST vollkommen: Wer große Scharen von Entwicklern an sich binden möchte, kann keine komplexe Technologie auswählen, die nur von professionellen Softwaretechnikern beherrscht wird und insbesondere in ihren Anfangszeiten für mittelmäßige Performance berüchtigt war. Viel geeigneter ist eine Technologie, die auch für Hobbyentwickler, insbesondere Schüler und Studenten, und damit viel größeren Teilen der Gesellschaft leicht zugänglich ist und zu guter Letzt auch noch sehr gut skaliert: REST. Tatsächlich werden beispielsweise viele mobile Apps von Start-ups und Einzelpersonen entwickelt. Für die Anbieter von Cloud- und mobilen Plattformen sind RESTful Webservices also durchaus eine vernünftige Wahl. Aber wie sieht das mit der IT für die öffentliche Verwaltung aus?

Behördliche Informationssysteme, wie Register und Vorgangsbearbeitungssysteme, verfolgen nach wie vor primär die Ziele Robustheit, Sicherheit und Wirtschaftlichkeit. Die Zugänglichkeit für semiprofessionelle Entwickler spielt genauso wenig eine Rolle wie die Skalierbarkeit auf Millionen von Nutzern. Daher rücken die Stärken der SOAP Webservices wieder in den Vordergrund: Typsicherheit, Schemavalidierung, Authentifizierung, Verschlüsselung und andere werden durch SOAP Webservices unmittelbar unterstützt und als Teil des Schnittstellenvertrags explizit gemacht. Für die Anwendungsintegration innerhalb einer Behörde oder zwischen Behörden, insbesondere für komplexe und sicherheitskritische Verfahren, ist SOAP also nach wie vor die richtige Wahl – und wird es aufgrund der unterschiedlichen Philosophie von SOAP und REST auch bleiben. Das spiegelt sich beispielsweise auch in der Abstützung des im XÖV-Kontext bedeutsamen OSCI-Netzwerkprotokolls auf SOAP-Webservices wider.

Das bedeutet aber nicht, dass RESTful Webservices keine Anwendungsfälle in der öffentlichen Verwaltung haben. Gerade in den Bereichen Open Government und Open Data richtet die öffentliche Verwaltung ein elektronisches Angebot direkt an den Bürger. Darunter werden zwar zumeist Webanwendungen verstanden, die mittels grafischer Benutzeroberflächen für die Bürger nützliche

RESTFUL WEBSERVICES

RESTful Webservices machen nur wenige konkrete Vorgaben an die Gestaltung der Schnittstellen. Einzig die semantisch korrekte Verwendung des http-Protokolls und der dort vorgesehenen Befehle GET, POST und DELETE wird verlangt. Die Parameter des Aufrufs werden typischerweise direkt in die URL codiert. Die Nachrichten und die übertragenen Daten werden nicht durch ein Schema beschrieben. Vorgeschlagene Spezifikationsformate wie WADL[1] und RSDL[2] sind nicht standardisiert und haben sich nicht durchgesetzt. Es gibt auch keine feste Vorgabe über das zu nutzende Datenformat für die Übertragung. Gängig sind XML und JSON.[3]

Da REST keine formale Beschreibung des Schnittstellenvertrags verwendet, ist der Entwickler auf informelle Dokumentation durch den Serviceanbieter angewiesen. Diese ist oft durch zahlreiche Beispiele geprägt, die durch Kopieren des Codes einfach in die eigene Software übernommen werden können. Einfache Abrufe können sogar direkt über die URL-Zeile des Browsers ausgelöst werden. Aus den beobachteten Ergebnissen kann der Entwickler sich dann die weitere Funktionsweise erschließen. Dieser explorative Entwicklungsansatz passt sehr gut zu den mobilen Plattformen und den Cloud-Plattformen der Internetgiganten (wie zum Beispiel Amazon oder Google), die damit einen sehr großen Entwicklerkreis ansprechen.

1 Web Application Description Language, siehe http://java.net/projects/wadl/sources/svn/content/trunk/www/wadl20090202.pdf?rev=328
2 RESTful Service Description Language, siehe http://www.balisage.net/Proceedings/vol10/html/Robie01/BalisageVol10-Robie01.html
3 JavaScript Object Notation, siehe https://tools.ietf.org/html/rfc7159

> *Consider how often we see software projects begin with adoption of the latest fad in architectural design, and only later discover whether or not the system requirements call for such an architecture. Design-by-buzzword is a common occurrence.*[5]
>
> Roy Thomas Fielding

DIE SOAP-PROTOKOLLFAMILIE

Klassische Webservices verwenden in der Regel das SOAP-Protokoll. Dabei handelt es sich um ein komplexes, XML-basiertes Protokoll, das über unterschiedliche Transportprotokolle, wie zum Beispiel HTTP oder JMS, übertragen werden kann. SOAP sieht eine explizite Formulierung der Schnittstelle unter Verwendung des WSDL-Formats[6] vor. Dieses wiederum verwendet den XSD-Standard zur Formulierung der übertragenen Datenstrukturen.

Das ursprüngliche SOAP-Protokoll kannte viele, aus den vorausgehenden Schnittstellentechnologien wie CORBA und EJB/IIOP bekannte nichtfunktionale Eigenschaften, wie Authentifizierung und Sicherheit, nicht. Mittlerweile wurden diese Lücken durch Erweiterungen des Standards, die sogenannten WS-*Spezifikationen, geschlossen. Dadurch wird die Protokollfamilie aber immer unübersichtlicher. Weiter kompliziert wird die Situation dadurch, dass nicht alle SOAP-Werkzeuge alle WS-*Spezifikationen unterstützen.

Datensammlungen aus der öffentlichen Verwaltung (wie zum Beispiel Daten über Verkehrsflüsse) bereitstellen. Gerade für diese Open Data sind aber einfache Schnittstellen sehr sinnvoll und die Zugänglichkeit für Entwickler – auch im semiprofessionellen Bereich – ist essenziell. Hier ist der Einsatz von RESTful Webservices eindeutig anzuraten.

Damit wird klar: RESTful Webservices und SOAP-Webservices adressieren völlig unterschiedliche Einsatzgebiete. Während SOAP-Webservices robuste und sichere Schnittstellen zwischen den komplexen Fachverfahren der Behörden adressieren, unterstützen RESTful Webservices einfach zugängliche öffentliche Schnittstellen. RESTful Webservices werden SOAP daher nicht verdrängen. Das wäre auch im Sinne von Roy Fielding, der sich in seiner Dissertation deutlich gegen das blinde Umsetzen von Trends ausgesprochen hat. ●

ANSPRECHPARTNER – ANDREAS RAQUET

Principal IT-Consultant

Public Sector

- +49 711 94958-693
- andreas.raquet@msg-systems.com

5 Fielding, Roy Thomas (2000): Architectural Styles and the Design of Network-based Software Architectures, https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

6 Web Services Description Language, siehe https://www.w3.org/TR/wsdl.html