REST Web Service – SOA Enablement für das Web

Stefan Kühnlein
IBM Deutschland Enterprise Solution Applications GmbH
München

Schlüsselworte

REST, SOA, ROA, Web Service, HTTP

Einleitung

Die architekturellen Design Patterns von Service orientierten Architekturen (SOA) und Ressource orientierten Architekturen (ROA) und die zugehörigen verteilten Programmierparadigmen stellen für die Entwicklung von verteilten Architekturen konzeptionelle Methoden und Entwicklungs-tools bereit. Das wesentliche Kennzeichen einer verteilten Architektur ist die Bereitstellung von Komponenten, auf die Service Consumer oder weitere Komponenten über das Netzwerk und über eine fest definierte Schnittstelle zugreifen können. In einer SOA- bzw. ROA-Architektur werden diese verteilten Komponenten als Services bzw. Resources bezeichnet.

Services in einer Service orientierten Architektur basieren auf einer funktionellen Zerlegung der Enterprise Business Architektur in zwei Top-Level Abstraktionen dar: Enterprise Business Process und Enterprise Business Services.

REST stellt auf der anderen Seite eine Reihe von architekturellen Richtlinien für Ressource orientierte Architektur dar. ROA basiert auf dem Konzept von Ressourcen, die jeweils eine direkt zugänglich verteilte Komponente, deren Zugriff über eine standardisierte Schnittstelle erfolgt.

Service Orientierte Architektur

Ein Service in einer SOA-Architektur stellt für bestimmte Geschäftsanwendungen bestimmte Funktionalität zur Verfügung und in ISO 19119 ist dies definiert als ein bestimmter Teil einer Funktionalität, die von einem Unternehmen über eine Schnittstelle zur Verfügung gestellt wird. Diese Services sind in der Regel Teile einer Anwendung, die ohne eine menschliche Interaktion auskommen, auf Servern ausgeführt werden und über eine fest definierte Schnittstelle konsumiert werden können.

Im Rahmen des Designs einer SOA-Architektur muss die Granularität der verschiedenen Service-Typen definiert werden. Normalerweise wird diese Phase durch die Katalogisierung der Funktionalität und an Hand der zugrundeliegenden Infrastruktur getrieben.

Die richtige Entwicklung einer SOA-Architektur hängt von der spezifischen Plattform, der Kommunikationsinfrastruktur (in der Regel http) und der Implementierung der Schnittstelle ab.

Ressource Orientierte Architektur

Ressource orientierte Architekturen basieren auf dem Konzept von Ressourcen. Ressourcen folgen einem relativen abstrakten und generischen Konzept. Jede Ressource ist eindeutig identifizierbar und ihr Status kann über eine standardisierte Schnittstelle geändert werden. Eine Ressource zeichnet sich durch folgende wesentliche Eigenschaften aus.

- Ressource
  Die Ressource selbst, die extern über eine eindeutige URL erreichbar ist
- Name Ressource
  Eindeutige Identifizierung der Ressource
- Darstellung der Ressource
  Information über den aktuellen Zustand einer Ressoruce

- Link zur Ressource
  Link zu einer anderen Darstellung der gleichen Ressource oder einer anderen Ressoruce
- Schnittstelle der Ressource
  Einheitliche Schnittstelle für den Zugriff auf die Ressource und Manipulation des Zustandes

Die Semantik der Schnittstelle für den Zugriff auf Ressourcen basiert auf den vom HTTP-Protokoll zur Verfügung gestellten Operationen. Die folgende Tabelle stellt die Methoden dar, mit denen auf Basis des HTTP-Protokolls auf Ressourcen zugegriffen und deren Status geändert werden können.

Ressource Methode	Beschreibung	HTTP-Operation
Create()	Erstellen einer neuen Ressource	PUT
GetRepresentation()	Ermitteln einer Ressource	GET
Delete()	Löschen ein Ressource	DELETE; POST sofern es sich um das Löschen von verlinkten Ressourcen handelt)
Modify()	Ändern einer Ressource	POST
GetMetaInformation()	Ermitteln der Metadaten einer Ressource	HEAD

Gegenüberstellung von SOA und ROA

Basierend auf den getroffenen Definitionen einer SOA- und ROA-Architektur sind die Unterschiede zwischen den beiden Architekturkonzepten deutlich.

Services stellen in der Regel einen Kontroller dar, der eine bestimmte Aufgabe, unabhängig der zu Grunde liegenden Ressource, ausführt. So kann z.B. ein Bankenservice auf ein Konto einen bestimmten Betrag abbuchen.

Ressourcen hingegen stellen einen Mechanismus für den Datenzugriff auf eine bestimmte Instanz eines bestimmten Datenobjektes. So muss für die Abbuchung eines Betrages von einem Konto dieses als Ressource ermittelt werden und dann die Abbuchung vornehmen zu können.

Verwendung von RESTful Web Services

Für die Entscheidung, ob eine RESTful-Architektur eingesetzt werden kann, benötigen Architekten und Entwickler Kriterien, an Hand derer die Entscheidung getroffen werden kann. Die Verwendung einer ROA-Architektur kann in folgenden Fällen verwendet werden:

- Zustandslose RESTful Web Services
  RESTful Web Services sind vollständig zustandslos und idempotent. D.h. ein RESTful Web Service muss im Falle eines Kommunikationsabbruches oder eines Fehlerfalls auf dem Server mehrfach aufgerufen werden können. In einer ROA-Architektur wird vorgeschrieben, dass der Zustand entweder von Client gehalten oder vom Server in einen Ressourcenstatus umgewandelt wird. In einer ROA-Architektur ist es nicht gewollt, dass der Status serverseitig transient und clientspezifisch über die Dauer eines Request hinweg abgelegt wird.

- Ressourcen mit eindeutiger Identifikation und unterschiedlichster Repräsentation
  Wie bereits erwähnt, sind Ressourcen das zentrale Konzept einer REST-Architektur. Eine Ressource zeichnet sich durch zwei wesentliche Eigenschaften aus – Identifikation und Repräsentation. In einer REST-Architektur kann somit die ein und dieselbe Ressource mehrere Repräsentationen haben. Somit kann zur Steigerung der Performance kann eine Caching-Infrastuktur verwendet werden. Dies ist der Fall, wenn die von der http-Methode Get angeforderten Daten nicht dynamisch generiert und zwischengespeichert werden.

- Begrenzung der Bandbreite
  Wird eine Begrenzung der Bandbreite benötigt, so ist eine ROA-Architektur zu bevorzugen. Ein ROA basierter Architekturansatz ist vor allem bei der Verwendung von begrenzten Profilen wie PDAs und SmartPhones effektiv, da hier auf den Overhead der Schichten in einem SOAP-Protokoll verzichtet werden kann.

Im Gegenzug sollte eine SOA-Architektur eingesetzt werden, wenn folgende Punkte erfüllt werden:

- Es wird eine formale Beschreibung der Schnittstelle, die durch den Web Service bereitgestellt wird, benötigt. Die Web Service Desciption Language (WSDL) beschreibt die Details wie Operationen, Binding und Ort des Dienstes.
- Die Architektur adressiert komplexe nicht funktionale Anforderungen. Beispiele sind hierfür Transaktionen, Security, Adressierung usw. Die meisten funktionalen Anforderungen von komplexen Systemen gehen über die einfachen Create-, Read-, Update- und Delete-Operationen und verlangen kontextbezogene Informationen. Mit einer ROA-Architektur müssten diese Anforderungen in der Anwendungsschicht erneut implementieren.
- Die Architektur muss eine asynchrone Verarbeitung von Aufrufen behandeln. Für diese Anforderung stellt die Infrastruktur bereits die notwendigen Standards wie Web Services Relaiable Messaging (WSRM) und die notwendigen APIs wie JAX-WS bereit.

Design von ROA basierten Architekturen

Einer der wichtigsten Aspekte beim Design einer Ressource orientierten Architektur ist der Entwurf der Ressource selbst und welche Anforderung an die Ressource müssen erfüllt werden.

Dies ist einer der Hauptunterschiede zwischen einer ROA und SOA Architektur. Wie bereits erwähnt werden in einer SOA komplexe Funktionalitäten einer Geschäftsanwendung über mehrere Komponenten verteilt, die wiederum mit Ressourcen arbeiten. Diese Ressourcen können beim Design einer ROA basierten Architektur als Kandidaten für verteilte Ressourcen verwendet werden.

Nach dem die Granularität einer Ressource für die ROA Architektur definiert wurde, ist es notwendig für jede Ressource den Inhalt der Nachricht für die Aufrufe der zu verwendenden Methoden als auch die Antwort auf den Request zu definieren. Dies bedeutet im Detail, dass neben der Definition der Ressourcetypen (inkl. den Subtypen) ein Adressierungsschema zum Zugriff auf die Instanzressourcen, ein Antwortschema und ein Mapping der logischen Funktionen auf die entsprechenden HTTP-Methoden erforderlich ist.

Ein wesentlicher Vorteil einer ROA basierten Architektur ist die gemeinsame Schnittstelle – HTTP. An dieser Stelle ist jedoch zu erwähnen, dass durch die Verwendung einer gemeinsamen Schnittstelle nicht notwendiger weise bedeute, dass alle notwendigen Informationen und somit die Interoperabilität und Zusammenarbeit zwischen den Ressourcen.

Beschreibung von RESTful Endpoints

Im Gegensatz zu SOAP-basierten Web Services, deren Endpoints über eine WSDL beschrieben werden, verfügen RESTful Web Service über eine derartige Grammatik. Damit ein Service-Consumer den Kontext und den Inhalt der gesendeten Daten verstehen kann, müssen sowohl der Service-Consumer als auch der Service-Provider eine Vereinbarung treffen. Dies geschieht mit Hilfe der Web Application Description Language WADL.

WADL ist eine Beschreibungssprache, die sowohl zu REST als auch HTTP passt. Mit WADL soll erfolgt eine Beschreibung von Ressourcen, von den ausgetauschten Formaten und den unterstützen Methoden auf Basis von XML.

Im Gegensatz zu WSDL und anderen Schnittstellenbeschreibungssprachen ist WADL spezielle auf die Verwendung von RESTful Web Services zugeschnitten. Das folgende Bespiel soll den Aufbau einer WADL verdeutlichen.

URI-Template	Verb	Request	Response	Beschreibung
/res	GET	-	XML	Ermitteln einer Liste von Ressourcen
	POST	XML	XML	Anlegen einer Ressource
/res/[id]	GET	-	XML	Ermitteln einer bestimmten Ressource
	PUT	XML	XML	Aktualisieren einer Ressource
	DELETE	-	-	Löschen einer Ressource

Zusammenfassung

Der wesentliche Unterschied zwischen einer SOA- und ROA- basierten Architektur besteht in der Semantik der zugrundeliegenden Web Services. SOAP-basierte Web Services basieren auf einer formalen Beschreibungssprache und können komplexe Funktionalitäten adressieren. ROA-basierte Web Services adressieren hingegen fest definierte Ressourcen über HTTP, deren Repräsentation auf unterschiedlichster Art und Weise erfolgen kann.

Kontaktadresse:

Stefan Kühnlein
Firma IBM Deutschland Enterprise Application Solutions GmbH
Hollerithstraße 1
D-81829 München

Telefon: +49 (0) 160-88 48 611
E-Mail Stefan.Kuehnlein@de.ibm.com
Internet: www.ibm.com/de