REST on Rails

Phillip Ghadir, pg@innoq.com
http://www.innoq.com/blog/pg

Stefan Tilkov, st@innoq.com
http://www.innoq.com/blog/st

rails-konferenz.de 2007

www.innoq.com

Agenda

- REST-Einführung
- REST jenseits von CRUD
- Rails & REST & die Zukunft

REST: Architektur des WWW

- REpresentational State Transfer
- Beschrieben von Roy Fielding in seiner Dissertation
- Einer von mehreren “Architekturstilen”
- Architekturprinzinzien des HTTP-Protokolls, definiert a posteriori

http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Disclaimer

- REST als abstraktes Konzept könnte mit beliebigen Technologien umgesetzt werden
- Die bekannteste Implementierung ist HTTP
- Hier: REST = RESTful HTTP

http://savas.parastatidis.name/2005/03/12/505b74f7-d5d3-4b94-95d4-65129ce2bf2b.aspx

REST Kernprinzipien

- Identifizierbare Ressourcen
- Uniforme Schnittstelle
- Statuslose Kommunikation
- Ressource-Repräsentationen
- Hypermedia

Identifizierbare Ressourcen

- Eine Ressource repräsentiert eine wirkliche oder virtuelle Entität
- Im Web werden Ressourcen durch URIs identifiziert
- Jede URI steigert den “Wert” des Internets
- Was wäre Amazom.com ohne URIs?

Uniforme Schnittstelle

- Ist die URI bekannt, kann man mit einer Ressource über *genau eine* Schnittstelle interagieren
- Begrenzte Menge an Operationen (*Verben*) in HTTP: GET, PUT, POST, DELETE (+ ein paar mehr)
- Vordefinierte Semantik erlaubt Optimierungen (z.B. Caching)

Ressource-Repräsentationen

- Der Zugriff auf Ressourcen erfolgt immer über Repräsentationen
- Es kann mehrere Repräsentationen geben, z.B. HTML, PDF, XML, JSON
- HTTP unterstützt *content types* und *content negotiation*
- Idealerweise: Standardformate

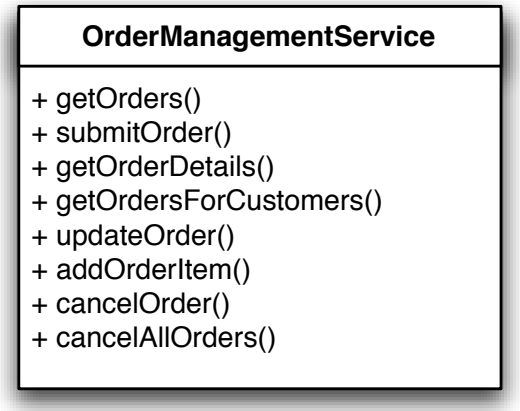
Statuslose Kommunikation

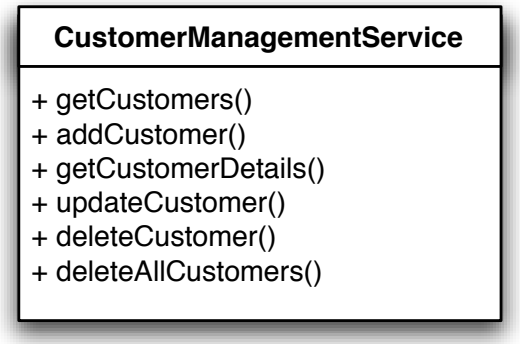
- Der Server muss keinen kommunikationsbezogenen Status für jeden Client halten
- Massiver Skalierbarkeitsvorteil
- Unterstützt lose Kopplung (keine Bindung an einen spezifischen Server)

Hypermedia

- Statusübergänge über *Links*
- Ermöglicht das Verfolgen von beliebigen Assoziationen
- Links sollten vom Server, nicht vom Client erzeugt werden
- Unterstützt Evolution von Schnittstellen
- Abstrahiert von konkreten Lokationen

Schnittstellendesign ohne REST





- Eine neue Schnittstelle (Façade) für jeden Service
- Bekannt aus CORBA, DCOM, RMI/EJB
- Typisch verwendet für SOA (“CORBA mit spitzen Klammern”)
- Applications-spezifisches Protokoll

/customers/
/customers/:id
/customers/:id/orders
/orders/:id

URIs

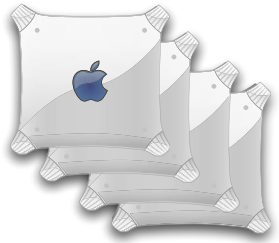
Design einer REST-Applikation

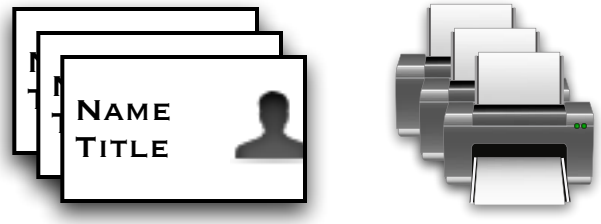
http://bitworking.org/news/How_to_create_a_REST_Protocol

DELETE
GET
POST
PUT

200 - OK
201 - CREATED
202 - ACCEPTED
302 - MOVED TEMPORARILY
404 - NOT FOUND

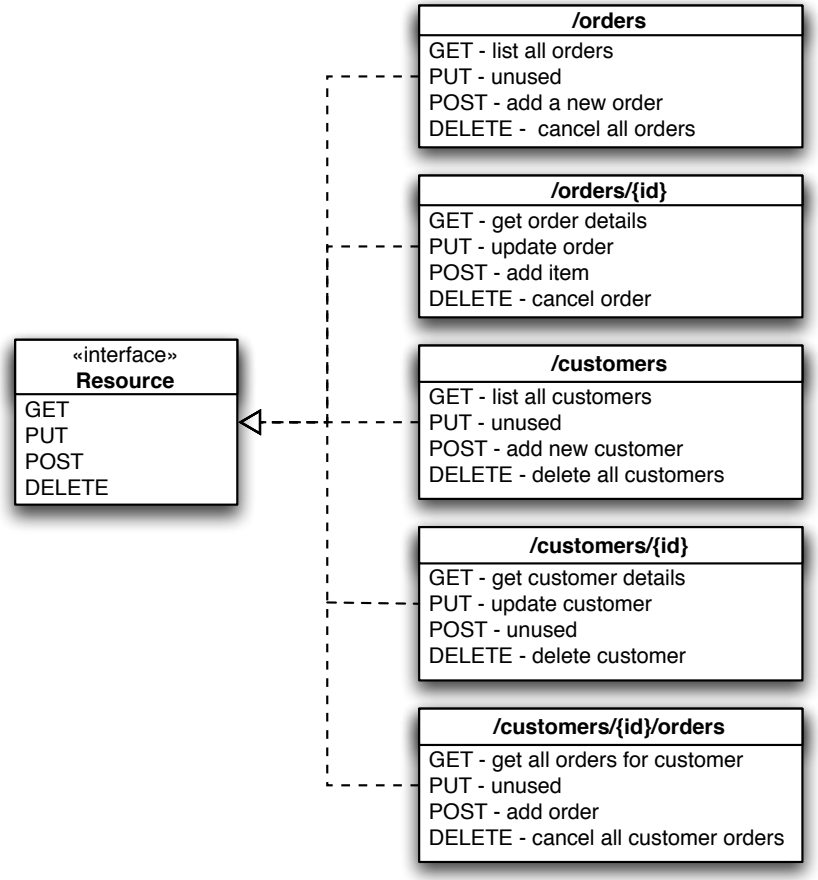
Operationen und Rückgabewerte





Ressourcen

REST-Beispiel



- Einheitliches Interface für alle Ressourcen
- Abbildung generischer Verben auf konkrete Applikationssemantik
- Standard-Applikationsprotokoll

Vorteile des REST/HTTP-Ansatzes

- *Wirklich* universelle Unterstützung (Programmiersprachen, Betriebssysteme, Server, …)
- Vorhandene, hochoptimierte Infrastruktur
- Bewiesene Skalierbarkeit
- Unterstützung für Redirects, Caching, Ressourcen-Identifikation
- Web-Integration für Maschinen-zu-Maschinen-Kommunikation
- Unterstützung für XML, aber auch andere Formate

REST & Rails

7 Statements

Rails hat kein Komponentenmodell

- C, C++ haben CORBA
- Java hat EJB
- Ziel der Vereinheitlichung über Web-Services
- Das Komponentenmodell von Rails ...
  ... ist die Architektur des WWW

Komponenten = Rails-Anwendungen

- Anwendungen als Einheit des Deployments
- RESTful HTTP-Schnittstellen zwischen Anwendungen
- Trennung von Schnittstelle und Implementierung
- Interoperabilität
- Plattformunabhängigkeit

Aggregation von Anwendungen

- Die Enterprise-Antwort: per Portal-Server
- In Rails: HTML/JavaScript, HTTP
- Universelle Benutzerschnittstelle: das Web

REST ist die Zukunft, nicht WS

- REST ist kurz vor dem Mega-Hype - siehe Gartner, IBM, Burton, ...
- Wir glauben das.
- DHH - bzw. die Roadmap von Rails - auch. :-)
- Edge Rails (und 2.0) fokussiert noch stärker auf REST als Default

REST hilft Rails

- REST gewinnt an Momentum
- Damit “passt” das Rails-Konzept
- Minimierung des (tatsächlichen oder angenommenen) Risikos
- Geringere Hürde wg. Austauschbarkeit der Implementierung
- Das Unternehmen setzt auf REST, nicht auf Rails

Rails hilft REST

- Rails-Anwendungen sind per Default “RESTful”
- Freies Design des URI-Raums problemlos
- “Best practices” werden “ab Werk” unterstützt (z.B. “Collection resources”)
- Rails ist eine besonders geeignete Technologie für dessen Umsetzung

Rails & REST & Web 2.0

- REST liefert Grundlagen für Web 2.0-Anwendungen (Mashups & Co.)
- Einheitliches Interface fördert Netzwerkeffekt
- Content Types erlauben Evolution
- Hypermedia unterstützt Verknüpfung

Fazit

- REST: Architektur des erfolgreichsten verteilten Systems der Welt
- *Wirklich* große Systeme: Integration über Web-Technologien
- Rails ist auf dem “REST-Trip”
- REST hilft Rails – Rails hilft REST
- Macht REST!

Vielen Dank!

weitere Informationen:

- www.innoq.com/resources/REST
- REST-Tutorial auf b-simple.de

Noch Fragen?

Phillip Ghadir
http://www.innoq.com/blog/pg/
phillip.ghadir@innoq.com

Stefan Tilkov
http://www.innoq.com/blog/st/
stefan.tilkov@innoq.com

innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Phone +49 21 02 77 162-100

innoQ Schweiz GmbH
Gewerbestrasse 11
CH-6330 Cham
Phone +41 41 743 01 11

info@innoq.com · www.innoq.com