REST

Software Engineering II (IB)

Prof. Dr. Oliver Braun

Letzte Änderung: 07.04.2019 14:34

Not RESTful

Quelle: http://geek-and-poke.com/geekandpoke/2013/6/14/insulting-made-easy

Grundlage

Roy T. Fielding, Richard N. Taylor: **Principled design of the modern Web architecture**.

Was ist REST

- **RE**presentional **S**tate **T**ransfer
- Ressourcenbasiert
- Repräsentationen
- 6 Constraints
  - Uniform Interface
  - Stateless
  - Cacheable
  - Client-Server
  - Layered System
  - Code on Demand (optional)

Ressourcenbasiert

- Dinge, keine Actions
- Nomen, keine Verben
- keine RPCs
- Identifiziert durch URIs
    - mehrere URIs für eine Ressource möglich
- getrennt von der Repräsentation

Repräsentationen

- Teil des Zustands einer Ressource
    - wird zwischen Client und Server übertragen
- typischerweise JSON oder XML
- Beispiel
    - Ressource: person (Hugo)
    - Service: contact information (GET)
    - Repräsentation
        - Name, Adresse, Telefonnummer
        - JSON oder XML Format

Uniform Interface

- definiert die Schnittstelle zwischen Client und Server
- vereinfacht und entkoppelt die Architektur
- fundamental für das RESTful Design
- in HTTP
    - HTTP Verben (GET, PUT, POST, DELETE)
    - URIs (Namen von Ressourcen)
    - HTTP Response (Status, Body)

Stateless

- Server hat keinerlei Client-Zustand
- jeder Request muss genug Informationen enthalten um die Message zu behandeln
    - self-descriptive messages
- jeglicher Zustand wird auf dem Client vorgehalten

Cacheable

- Server Responses (Repräsentationen) sind cacheable
    - implizit
    - explizit
    - verhandelbar

Client-Server & Layered System

Code on Demand (optional)

- z.B. über JavaScript oder Java Applets

RESTful oder nicht?

- wenn einer der Constraints (außer Code on demand) verletzt ist, ist es nicht mehr RESTful
- wenn alles eingehalten wird, bietet ein RESTful Service
    - Skalierbarkeit
    - Einfachheit
    - Anpassbarkeit
    - Portabilität
    - Zuverlässigkeit

REST APIs must be hypertext-driven

Blog-Post von Roy T. Fielding, 20.10.2008.

- A REST API should not be dependent on any single communication protocol...
- A REST API should not contain any changes to the communication protocols aside from...
- A REST API should spend almost all of its descriptive effort in defining the media type(s) used for representing resources and driving application state...
- A REST API must not define fixed resource names or hierarchies (an obvious coupling of client and server).
- A REST API should never have “typed” resources that are significant to the client.

HYPERTEXT!!!

- REST soll wie das World Wide Web funktionieren
- Es muss nur die Startseite bekannt sein!
- diese antwortet unter anderem mit Links wo der Rest zu finden ist
- Smart Client Message (kein REST)

```
{
  "account": {
    "name": "Rest",
    "accountnumber": 12345,
    "balance": 6000.00
  }
}
```

Hypermedia As The Engine Of Application State(HATEOAS) (1/2)

- HATEOAS Message

```
{
  "account": {
    "name": "Rest",
    "accountnumber": 12345,
    "balance": 6000.00,
    "link": [
      {
        "rel": "self",
        "href": "/account/9963",
        "method": "get"
      },
      ...
```

- HATEOAS Message

```
        ...
        {
          "rel": "deposit",
          "href": "/account/9963/deposit",
          "method": "post"
        },
        {
          "rel": "withdraw",
          "href": "/account/9963/withdraw",
          "method": "post"
        }
    ]
  }
}
```

Beispiel: Paypal REST API

https://developer.paypal.com/docs/integration/direct/paypal-rest-payment-hateoas-links/