

REST - En webservice - lavet med HTTP og XML

Martin Boel, formand, javagruppen

REST - En webservice

representational state transfer & REST introduced in 2000 by Roy Fielding

HTTP	CRUD
POST	Create
GET	Read
PUT	Update, Create
DELETE	Delete

Kilde: OIORest - principper

OIORest - principper

1. Giv alle ting (ressourcer) en URL
2. Link tingene (ressourcerne) til hinanden
3. Alle ressourcer tilgås via samme interface
4. Tillad forskellige repræsentationer af samme ressource
5. Kommuniker tilstandsløs
6. Fejlhåndtering

Kilde:

<http://oiorest.dk/Documentation/restprincipper.aspx>

1. Giv alle ting (ressourcer) en URL

Navngiv alt hvad du er interesseret i.

Eksempler:

Københavns kommune: <http://oiorest.dk/danmark/kommuner/101>

Sorgenfri slot: <http://oiorest.dk/danmark/kommuner/173/lokaliteter/Sorgenfri> Slot

Dronningens adresse: <http://oiorest.dk/danmark/adresser/Amalienborg> Slotsplads,2,1257

Region Hovedstaden: <http://oiorest.dk/danmark/regioner/1085>

2. Link tingene (ressourcerne) til hinanden

☒ Hypermedia Driving Application State

☒

☒ Eksempel:

☒

☒

☒

☒ <?xml version="1.0" encoding="utf-8" ?>

☒ <kommune ref=<http://oiorest.dk/danmark/kommuner/101>

☒ xmlns="<http://itst.dk/schemas/danmarkservice>">

☒ <nr>101</nr>

☒ <navn>København</navn>

☒ <veje ref="<http://oiorest.dk/danmark/kommuner/101/veje>" />

☒ <adresser ref="<http://oiorest.dk/danmark/kommuner/101/adresser>" />

☒ </kommune>

☒

☒

☒

Metode	Beskrivelse	Egenskaber
GET	Modtag (muligvis cached)	safe, idempotent, cacheable
PUT	Opdater eller opret med kendt id.	idempotent
DELETE	Slet	idempotent
POST	Opret	

3. Alle ressourcer tilgås via samme interface (http)

4. Tillad forskellige repræsentationer af samme resource

- Self-Describing Messages
-
- Eksempler:
-
- XML: <http://oiorest.dk/danmark/kommuner/173/lokaliteter/Sorgenfri/Slot/adresser.xml>
-
- JSON: <http://oiorest.dk/danmark/kommuner/173/lokaliteter/Sorgenfri/Slot/adresser.json>
-
- Google Earth format: <http://oiorest.dk/danmark/kommuner/173/lokaliteter/Sorgenfri/Slot/adresser.kml>

5. Kommuniker tilstandsløs

- Hvert request er står alene. Den REST baserede web service husker ikke foregående kald fra samme klientapplikation.
-
- Første request kan håndteres af en server; andet request fra samme klient kan håndteres af en anden server. Tilstandsløsheden gør det bl.a. lettere at skalere sin løsning samt at debugge den.

6. Fejlhåndtering

- Option 1: Stick to HTTP Error Codes (and status text)
- http://www.example.com/xml/book?id=1234&dev_token=ABCD1234
- HTTP 404 Not Found Error Code. - "Id 1234 is not found in db_prod".
-
- Option 2: Return an Empty Set
-
- Option 3: Use Extended HTTP Headers
- HTTP/1.1 200 OK
- X-DAS-Status: 403
- Content-Type: text/plain
-
- Option 4: Return an XML Error Document
- <http://xoomle.dentedreality.com.au/search/?hl=en&ie=ISO-8859-1&key=&q=oreilly+php>
- <xoomleError>
- <errorString>Invalid Google API key supplied</errorString>
- </xoomleError>
-
- http://www.oreillynet.com/onlamp/blog/2003/12/restful_error_handling.html
-

What about: Transactions

- Keyword: Idempotent
- 1: put idkey
- 2: post data with idkey
- 3: get idkey, to verify status

ser

What about: Events

- Subscribe (with heartbeat):
- <http://example.com?notify=me.dk/notifications>
-
- Send event
- <http://me.dk/notifications/updated?idkey=1234>
-
- → get details
-
- Use a per-to-per world view
- Remember: 2-way