NetBeans Ruby Pack / Rails 2.0

Kazuhiro Yoshida <moriq@moriq.com>

2008-03-15

第24回 Ruby/Rails勉強会@関西

Rails 2.0

- **evolution** rather than **revolution**. by DHH
  - 革命的(revolution)というよりは漸進的(evolution)な進化
- RESTful

REST

REST

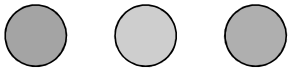
- Representational State Transfer (REST)
  - As architectural style
  - As application interface

ARCHITECTURAL STYLE

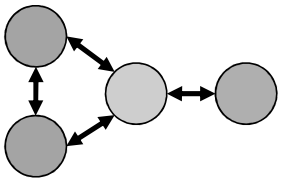
Network software 望まれる特性

- Scalability of component interactions
- Generality of interfaces
- Independent deployment of components
- Intermediary components to reduce interaction latency
- Enforce security
- Encapsulate legacy system

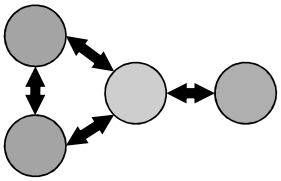
Component



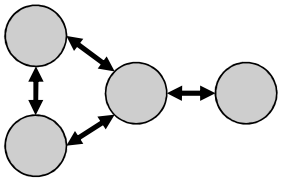
Software



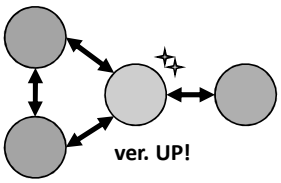
Scalability of component interactions



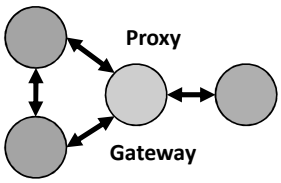
Generality of interfaces



Independent deployment of components



Intermediary components to reduce interaction latency



Network-based application architectural style

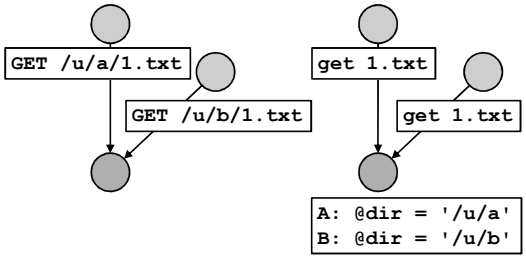
- Client-Server:
  クライアントサーバ構成
- Stateless:
  サーバはアプリケーション状態を持たない
- Cache:
  応答をキャッシュできる
- Uniform interface:
  統一インターフェイス
- Layered system:
  階層構造をもつ
- Code on Demand:
  クライアントはscriptを受信して実行できる

REST is …

- Uniform Layered Code-on-Demand Client Cache Stateless Server
- Architectural style

STATE

Stateless / Stateful

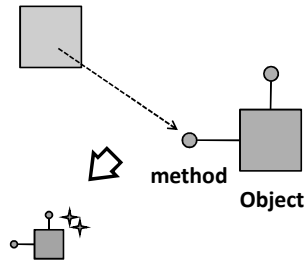


State

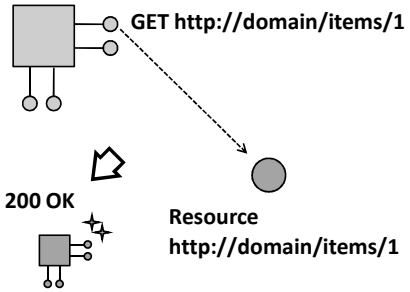
- Application state:
  クライアントの状態
- Resource state:
  サーバの状態
- Session:
  アプリケーション状態をサーバに持たせる仕組み
- Hypermedia as application state:
  アプリケーション状態としてのlink/form

INTERFACE

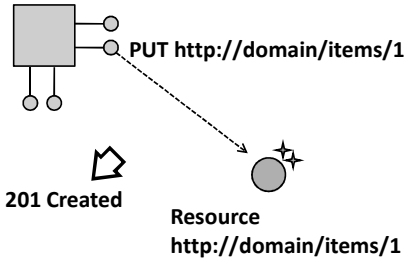
Object interface



Hypermedia interface



Hypermedia interface



WWW

World Wide Web

- www = REST.new
  - URI (Universal Resource Identifiers)
  - HTTP (Hypertext Transfer Protocol)
  - HTML (Hypertext Markup Language)

HTTP

HTTP method

- GET HEAD POST PUT DELETE OPTIONS TRACE CONNECT
  - HTTP/1.0
- PUT DELETE
  - ブラウザに実装されない理由
- http://www.studyinghttp.net/method

Safe and idempotent

- Safe method:
  GET HEAD
- Idempotent method:
  GET HEAD PUT DELETE

EXAMPLE

AWDwR 2nd Ed. Depot app.

- GET /store/index　カタログリスト
- POST /store/add_to_cart　カートに入れる
- POST /store/empty_cart　カートを空にする
- POST /store/checkout　チェックアウト
- POST /store/save_order　注文する

Uniform interfaceを適用

- GET /products　商品リストを得る
- POST /cart/items　カートに商品を入れる
  product_id=1
- DELETE /cart/items　カートを空にする
- GET /order　チェックアウト
- PUT /order　注文する
  committed=true

Session and singleton resource

- User home page
  - /users/1
  - /home
- Shopping cart
  - /carts/1
  - /cart
- Shopping order
  - /users/1/orders/1
  - /order

- sessionに関連付けたresourceはroutes上 singleton resourceとして表現できる

Cart as application state

- application状態としてのcart
  - cartをsessionに置く
  - sessionはapplication状態
- HTMLとしてのcart
  - hypermedia as application state
  - sessionなしでもできるけど…

Cart as resource state

- resource状態としてのcart
  - cartをDBに置く
  - AR modelはresource状態

Order as transaction resource

- cart resourceは不要
  - cartに入っている商品は注文確定前の注文商品
  - 注文はtransaction resourceとみなせる
  - するとcart resourceは要らなくなる

Order resource

- /order　注文　as transaction resource
  - GET　〇注文情報を得る
  - DELETE　〇注文を取り消す
  - PUT　〇注文を実行する　committed=true
  - POST　×注文を開始する

Order-items resource

- /order/items　注文商品リスト
  - GET　〇商品リスト情報を得る
  - DELETE　〇注文から商品を全て削除する
  - PUT　×
  - POST　〇注文に商品を追加する
product_id=1&quantity=1

Order-item resource

- /order/items/1　注文商品(id=1)
  - GET　〇商品情報を得る
  - DELETE　〇注文から商品を削除する
  - PUT　〇注文商品を更新する
product_id=1&quantity=1
  - POST　×