RESTful APIの みつけかた

2015-08-19

@moro

諸橋恭介 (@moro)

Kyosuke MOROHASHI

はじめる！ Cucumber

諸橋恭介

日本語で書かれた受け入れテストをRubyで実行できるようにする。

グランドルール

RESTfulとは DB のテーブルをそのままhttpで公開することを指すのではない

- パスワードは見せない、とかそういう話でもない
- モノの場合、同じようになることも多い

- Web アプリケーションが提供する機能を
- ユーザに役立つ粒度(リソース)で
  抽象化し
- そのリソースへのシンプルな操作(CRUD)
  を行うことで
- ユーザの用を満たせるようにするアーキテクチャパターン

- Web アプリケーションが提供する機能を
- ユーザに役立つ粒度(リソース)で抽象化

RESTful APIの みつけかた

2015-08-19

@moro

(DBの)
リソースエンティティ起点

- モノのCRUD
- POST /articles
- 一番?わかりやすい
- よくサンプルに出てくる

(DBの)
イベントエンティティ起点

- 関係性のCRUD
- `POST /groups/42/memberships`
- 2番め?にわかりやすい
  個人的に好き

- ～する/したこと
- DBテーブルがあるのでわかりやすい
- has_many :through で指定するやつ

サブリソースの操作

- add_fooしたくなったら考える
- POSTのことが多い
- リソースのURI決まるならPUT もあり

```
# よくやっちゃう。こうしたくなったら
POST /articles/42/add_comment

# サブリソースへ落とせる
POST /articles/42/comments
```

```
# config/routes
resources :articles do
  resources :comments
end
```

- verb+object なアクション
  はだいたいサブリソースへ
- もちろん参照ならGET

サブコレクションの操作

- 特定のテーマでまとめたリソースの集合
- indexで条件分岐を頑張り過ぎたら

```
# あと4つくらいパラメータが増える
GET /articles?filter=favorited&limit=3
```

```
# これは典型的なまとまりに名前をつけるとよい
GET /articles/hot
```

- アクションが分かれるので
すっきりする

うんちく: routing はセレクタブルディスパッチなわけで、ようするにそれ自体分岐の一種。

それを一つ処にまとめた上でもう一回自分で分岐を書き直す、みたいなことをするのは不毛。

メンバリソース

の操作

- 特定の(比較的大事な)属性の操作。
- set_foo みたいなルーティングを書きたくなったらコレ。
- POST や PUT のことが多い。

```
# 記事を下書きから公開にする例

# set_foo ではなく
POST /articles/42/set_public

# public -> publication と名詞にした上で
# それを「作成」する
PUT  /articles/42/publication


# 非公開にするには「公開していること」を消す
POST /articles/42/set_private
DELETE /articles/42/publication
```

トランザクション

リソース

- 「大きめの処理すること」を表すリソース
- 特に、複数のデータをまとめて扱うような場合に使う

- 作るだけでなく、実行や結果確認まで含む、処理全体をリソースとして抽象化する
- 実装的には、処理の実体は非同期Job に流すこともある

```
=begin
実際の例:

とある条件で、たくさんのデータをエクスポート
する必要あり。
まずはエクスポート条件を指定する画面
(ユーザナビゲーション用のリソースが必要)
=end
```

```
# ここでは条件指定フォームを表示
GET /foo_exports/new

# 指定した条件でエクスポート処理リソース作成。
# 処理の本体はActiveJobとかに
POST /foo_exports
```

```
# ジョブを作ったら詳細にリダイレクト
# ジョブの進捗を出せる範囲で出しておく
GET /foo_exports/42

# しばらく待ってリロードすると(※)終わる。
# 結果やログへのナビゲーションをつけておく
GET /foo_exports/42/data.csv
GET /foo_exports/42/log.txt



// ※ もちろんもっといまふうなWebSocketとかでもOK
```

サブリソースの
インライン化

- 関連しているリソースはJSONなどでネストする。
- ネストしない場合の参照はURLで。
- id だけで組み立ては避ける。

```
{article:  {
    id: 42,
    tiltle: 'RESTful API のみつけかた',
    url:    ‘http://speaekrdeck.com/moro/...',
    comments: [
      {id: 1, body: 'すごい'},
      {id: 2, body: '感動した'},
    ]
}}
```

AResは人類に
早すぎる話

は、今日はしません

デザインの

ヒント

- Railsの`resources`が作るすべてのアクションを持つ必要はない。
- あのサブセットとして、リソースを操作しつつ、リンクやフォームを使ってアプリケーションを作ることが大事。

よくある

ナビゲーション例

- 一覧画面の下部に追加フォームがある
  (`new` がない)

- 消せないので delete がない

- 修正できないので edit / update がない。

- 詳細画面の中で、はじめからテキストフィールドに値が書いてある
  - or JSでフォームを作ったりするのですぐに更新できる(edit がない)
- 親リソースの詳細画面に子リソースが表示される。(index も show もない)

独自研究

複合リソース/

多種ポータル

- Q: RESTfulAPIで「ポータルページ」をどう作るか
  - いろんなものを都度取りに行くとリクエスト数増えすぎてつらい

- 新着エントリ10件
- 人気エントリ10件
- お知らせ 3件
- ニュース 5件 …

- A: ポータルというリソースを定義すれば良いのでは?

おさらい

RESTfulとは DB のテーブルをそのままhttpで公開することを指すのではない

- Web アプリケーションが提供する機能を
- ユーザに役立つ粒度(リソース)で
  抽象化し
- そのリソースへのシンプルな操作(CRUD)
  を行うことで
- ユーザの用を満たせるようにするアーキテクチャパターン

それでけっこう

アプリが出来るのだ