React.js meets Loopback

ReactとLoopbackことはじめ

tyhsc

PROFILE

樋口 剛 TSUYOSHI HIGUCHI @tyshgc

Service Designer & UI Developer

国内UIデザインファームGoodpatch、ニュースサービスGunosyを経てフリーランス中

スタートアップや新しいプロダクトの
サービスデザインと設計・実装まで行っています。

AGENDA

1 簡単にLoopbackを紹介

2 React.jsとLoopbackを繋ぐ

3 まとめ

1

簡単にLoopbackを紹介

Loopback

StrongLoop社（IBM）のNode.js Web Framework

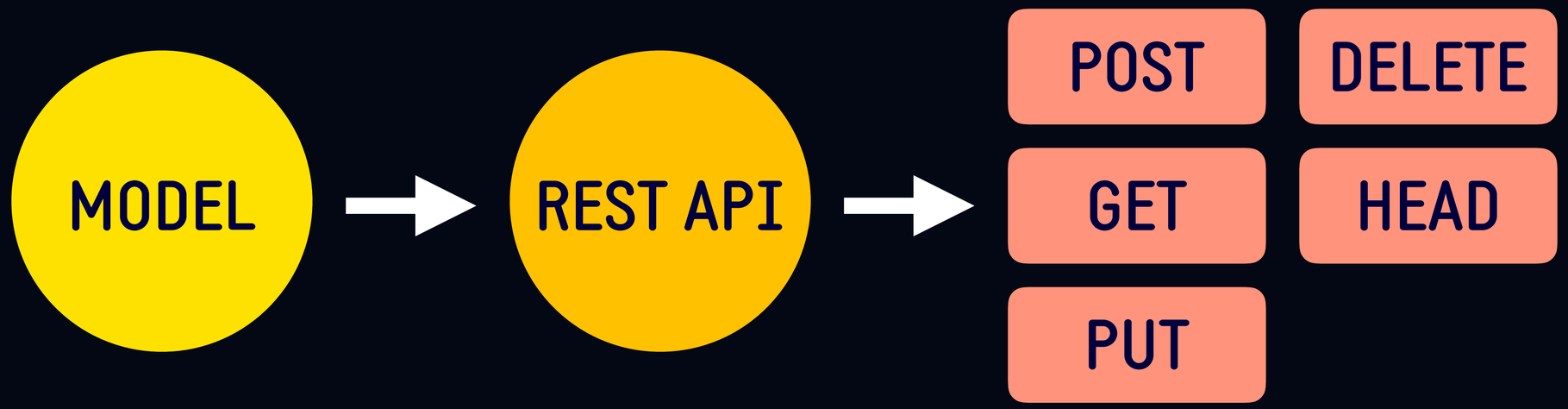
loopback.io

こちらから入手

```
$ npm i -g strongloop
```

特徴

1.抽象化モデルからREST APIを自動で生成



loopback-explorerが組み込まれているのでREST APIのAPIドキュメントの生成もやってくれる。

loopback-explorer

User

Show/Hide | List Operations | Expand Operations

GET	/Users	Find all instances of the model matched by filter from the data source.
PUT	/Users	Update an existing model instance or insert a new one into the data source.
POST	/Users	Create a new instance of the model and persist it into the data source.
GET	/Users/{id}	Find a model instance by id from the data source.
HEAD	/Users/{id}	Check whether a model instance exists in the data source.
PUT	/Users/{id}	Update attributes for a model instance and persist it into the data source.
DELETE	/Users/{id}	Delete a model instance by id from the data source.
GET	/Users/{id}/accessTokens	Queries accessTokens of User.
POST	/Users/{id}/accessTokens	Creates a new instance in accessTokens of this model.
DELETE	/Users/{id}/accessTokens	Deletes all accessTokens of this model.

loopback-explorer

User

Show/Hide | List Operations | Expand Operations

GET /Users — Find all instances of the model matched by filter from the data source.

PUT /Users — Update an existing model instance or insert a new one into the data source.

Response Class (Status 200)

Model | Model Schema


```
{
  "realm": "string",
  "username": "string",
  "credentials": {},
  "challenges": {},
  "email": "string",
  "emailVerified": true,
  "status": "string",
  "created": "2016-02-23",
  "lastUpdated": "2016-02-23",
  "id": 0
```


Response Content Type application/json

loopback-explorer

```
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
data	{ "realm": "ほげー", "username": "捕鯨", "credentials": {}, "challenges": {}, "email": "hoge@hogey.com", "emailVerified": true,	Model instance data	body	Model | Model Schema

Parameter content type: application/json

```
{
  "realm": "string",
  "username": "string",
  "credentials": {},
  "challenges": {},
  "email": "string",
  "emailVerified": true,
  "status": "string",
  "created": "2016-02-23",
  "lastUpdated": "2016-02-23",
  "id": 0
}
```

Click to set as parameter value

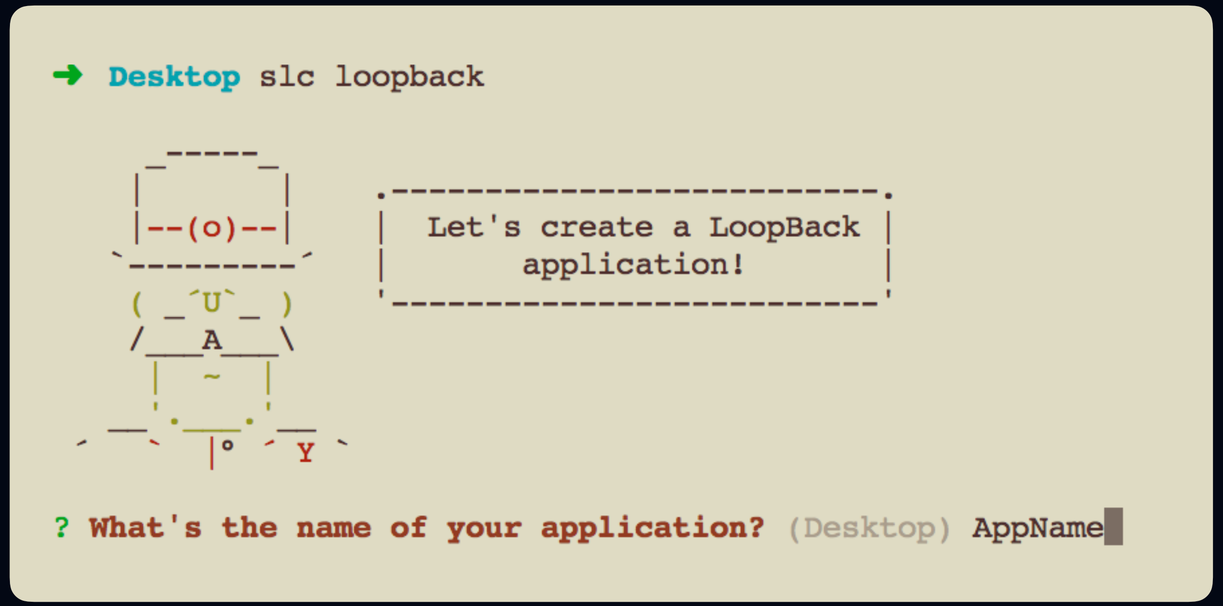
Try it out!

モックデータくらいなら
非エンジニアでも登録できる

2.CLIで作っていけるので割と簡単

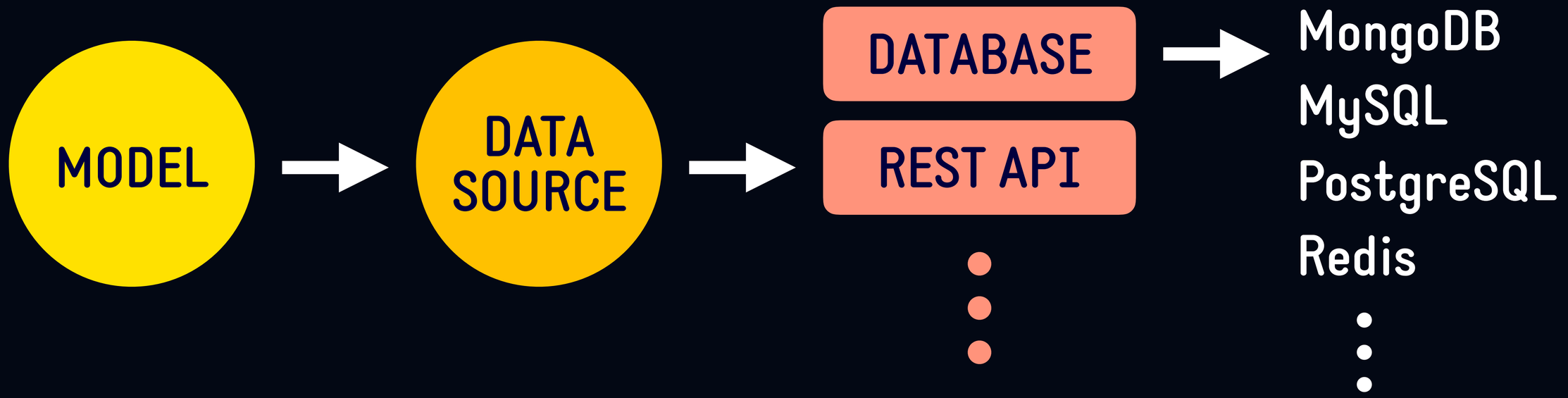
アプリケーションをつくるコマンド

```
$ slc loopback
```



他のコマンドはコチラから➡ **https://docs.strongloop.com/x/eYk6**

3.データソースとモデルの連携をいい感じに



DBや外部APIなどのデータソースに関するコネクターアドオンがある

外部APIなどをデータソースとして勿論扱えるので…

フロントエンドサーバ的な扱いができる

4.そのほか、色々ある様子

- Android SDKやiOS SDKが用意されてある
- Push通知やサードパーティログインなども用意されている
  API Explorer, OAuth2, Push Notifications, Storage, Third-party login (Passport)
- StrongLoop ArcというGUIツールもある
  `https://strongloop.com/node-js/arc/`
  使わなくても全く問題ない。

プロトタイプ開発にはライトに導入できて良い印象

2

React.jsとLoopbackを繋ぐ

APIがLoopbackでサクっと用意できるので

あとはReact.jsと連携するだけ

ただのAPIなのでFetchしてReduxのせればええやん…

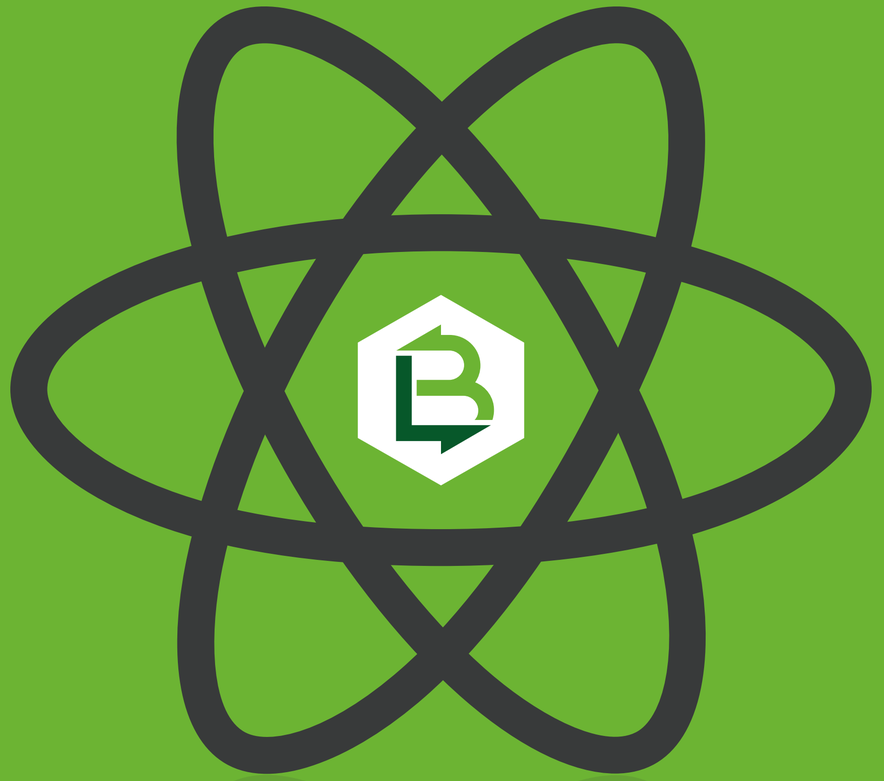
UIデザイナーの自分からすると、

正直、Reduxとか
プロトタイピングには重い…

勿論、ちゃんと開発するってなったらやるYo!

APIをサクっと生やして
実装により近いインタラクションを
プロトタイピングで実現したい


いいのモノがあった



React-Loopback

github.com/diogodoreto/react-loopback

こちらから入手

```
$ npm i react-loopback
$ npm i es6-promise whatwg-fetch
```

React-Loopbackは…

A small library to connect React components to data in a LoopBack API without the need for Flux, Redux, etc.

ReduxなどのFluxアーキテクチャを必要としない
Loopbackの生やすREST APIに繋ぐライブラリ

React-Loopbackは…

Inspired by Facebook's Relay.

Relayにインスパイヤされている

サンプルのTodo.jsxを見てみる

todo.jsx

```
import React, { PropTypes } from 'react';
import { createDataLoader } from 'react-loopback';

let Todos = React.createClass({
  propTypes: {
    todos: PropTypes.array
  },

  render() {
    const {todos} = this.props;

    return (
      <ul>
        {todos.map(todo => (
```

todo.jsx

```
  render() {
    const {todos} = this.props;

    return (
      <ul>
        {todos.map(todo => (
          <li>
            <input type="checkbox" checked={todo.done} />
            {todo.description}
          </li>
        ))}
      </ul>
    );
  }
});
```

todo.jsx

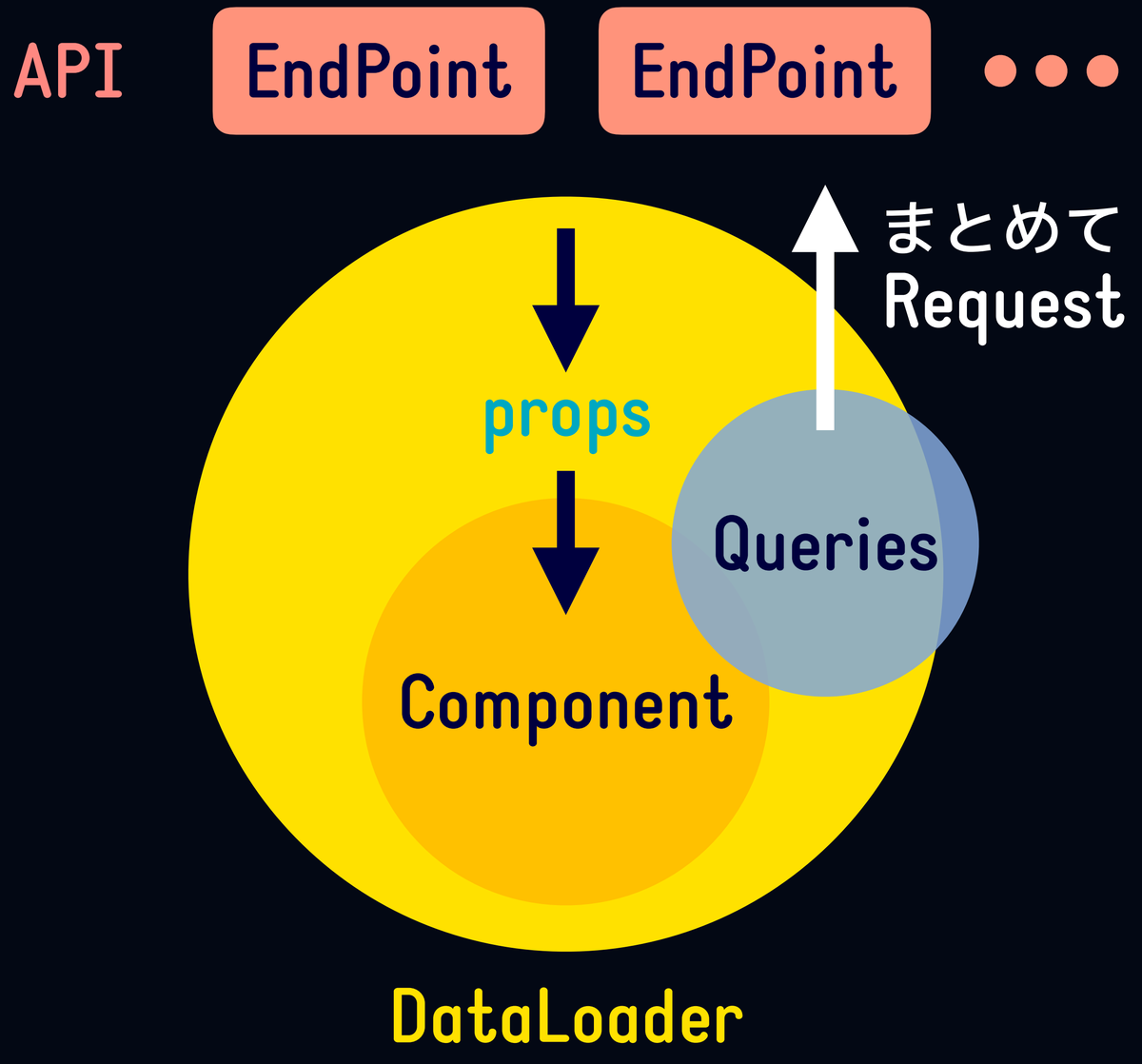
```
    );
  }
});
```

```
Todos = createDataLoader(Todos, {
  queries: [{
    endpoint: 'todos',
    filter: {
      where: {archived: false}
    }
  }]
});
```

```
export default Todos;
```

仕組み

APIのEndPointを
まとめて取りに行く
Fetch Wrapper Component



仕組み

APIのEndPointをまとめて
取りに行く
Fetch Wrapper Component

確かに

Relayぽさがある

3 まとめ

react-loopbackの人も言っている通りRelay的なアプローチ。
loopbackをすでに採用していたり今後採用予定がある場合は良いかもしれない。

もちろんReact-LoopbackでなくてもReduxやFluxとの共存も当たり前にできる。
プロトタイプにはLoopbackはかなりいいかも感。

所感

Loopbackでの開発はハッカソンやプロトタイプの開発に向いているなと思いました。

サクっと作るなら

React-Loopbackいいかも