

“*java*と*javascript*って何が違うの?”

“メロンとメロンパンみたいなものさ”

“インドとインドネシアとも言える。

全く違うのさ”

“*javascript*はブラウザで動いて、*java*は
*jvm*で動くのさ、おっとだれか来たよ
うだ

悪魔合体ソリューションのご提案



2015.12.22 社内LT大会 @tomlla

突然ではございますが、
みなさま、未だに
javaでご消耗中でしょうか？

個人的な不満を
ちらっと流します

つらい of the year 2015 finalist!

- ❖ eclipse しか使えない
eclipse ラムダ書いてられない おそい。
intellij idea使いたい。
- ❖ にいさん、ぼくはあと死ぬまでに何回 `public static` って書けば良いの？
- ❖ palyのモジュールじゃなくて、gradle or maven でライブラリ作りたい。
- ❖ rack, wsgi, plack, のmiddlewareみたいなコントローラーメソッドの前の処理を束ねる概念がない。
- ❖ javaはコレクションのAPIもstreamのAPIもいけてない。
- ❖ scala, clojure, kotlin が輝いて見える
May the frege be with you.
- ❖ jadeつかいたい. あれやりたい、これやりたい、でも、Playの上でそれかなえるの？
- ❖ ほんのすこしフォームが複雑になった途端、サーバーサイドレンダリングがしんどい

まあ、



理想への道のりは険しい

つらい of the year 2015 finalist!

- ❖ eclipse しか使えない
eclipse ラムダ書いてられない おそい。
intellij idea使いたい。
- ❖ にいさん、ぼくはあと死ぬ前に何回 `public static` って書けば良いの？
- ❖ palyのモジュールじゃなくて、gradle or maven でライブラリ作りたい。
- ❖ rack, wsgi, plack, のmiddlewareみたいなコントローラーメソッドの前の処理を束ねる概念がない。
- ❖ コレクションのAPIもstreamのAPIもいけてない。
- ❖ scala, clojure, kotlin が輝いて見える
May the frege be with you.
- ❖ jadeつかいたい. あれやりたい、これやりたい、でも、Playの上でそれかなえるの？
- ❖ ほんのすこしフォームが複雑になった途端、サーバーサイドレンダリングがしんどい

つらい of the year 2015 finalist!

- ❖ eclipse しか使えない
eclipse ラムダ書いてられない おそい。
intellij idea使いたい。
- ❖ にいさん、ぼくはあと死ぬ前に何回 `public static` って書けば良いの？
- ❖ palyのモジュールじゃなくて、gradle or maven でライブラリ作りたい。
- ❖ rack, wsgi, plack, のmiddlewareみたいなコントローラーメソッドの前の処理を束ねる概念がない。
- ❖ コレクションのAPIもstreamのAPIもいけてない。
- ❖ scala, clojure, kotlin が輝いて見える
May the frege be with you.
- ❖ jadeつかいたい. あれやりたい、これやりたい、でも、Playの上でそれかなえるの？
- ❖ ほんのすこしフォームが複雑になった途端、サーバーサイドレンダリングがしんどい

「ほんのすこしフォームが複雑になった途端、
サーバーサイドレンダリングがしんどい」件について

「ほんのすこしフォームが複雑になった途端、
サーバーサイドレンダリングがしんどい」件でよくあるあれ。

>> jsonのAPIでSPAだ！

> history API, bfcache どうするの？

> フロントエンドリダイレクトの挙動結構はまるよね

> jsのエントリーポイントどうする？

アプリ全体で1つだけのガチSPA? webpack? browserify?

> jsのテストいるでしょ? power-assert, mocha 使うよね?

> テストするならmoduleシステムいるよね？

> universal というかisomorphic に書きたい

> gulp, webpack, babel, vinyl-source-stream, いれようね..

> flux実装にする？実装系は？ビュー層のlibraryは？

フロントのrouterは？

考えることが多い...

現状すぐできるアーキテクチャ

```
ViewModel(  
  bindingVars { ... }  
  method: function() {  
    post("url")  
    .then(..) {  
    .catch(...)  
  }  
)
```

```
@get, @text/html  
def get(request.id) -> response  
  return render("xxx.html")
```

```
// @post, @application/json,  
def create(request) -> response  
  return {  
    createdId  
    validationErr: {}  
  }
```

response != 200 なら validationErrを
bindingVars.errにセット

ただし...

1. URL入れるの面倒だよね
2. DTO と ViewModelのdata部分、
ほぼおなじだよね

サーバー/フロントで共有できる
コード or データ or APIがあればいいなあ。

どうせなら

java + play1.x 以外で！

javaじゃない別の言語書いて、気持ちよくなりたい

さて、なにでやろう

いやあ、僕は最近、ほんと、
自分の方向性が定まらないのです。

浮気相手の名前	個人の意見です
ruby, python	まあ、良い。 個人的モチベーションはそんなにない。
c#, swift	プラットフォーム問題も解決されつつあるが、まだ 茨が道に落ちている
scala	でかい。やめろお！潰される！
golang	小さい、良い、でも2016年感がしないぜ
clojure	良い
js	誰もが避けては通れない

浮気相手の名前	個人の意見です
ruby, python	まあ、良い。 個人的モチベーションはそんなにない。
c#, swift	プラットフォーム問題も解決されつつあるが、まだ 茨が道に落ちている
scala	でかい。やめろお！潰される！
golang	小さい、良い、でも2016年感がしないぜ
clojure	良い
js	誰もが避けては通れない

誰もが避けては通れない！！

そこで

悪魔合体ソリューション！



サーバーサイド - フロントエンドを統合した
javascript on JVM Framework!!

feature

- ❖ APIのエントリポイントメソッド情報を共有し、urlやhttpの呼び出しミスを防ぐ
- ❖ DTOっぽい情報をserver/frontで2回書かなくてよい
- ❖ modern jsで書く! javaを書かなくて良い
- ❖ jsをnashornで動かすことでJVMの資産を使える!
- ❖ パッケージ管理は基本npm

題して

the **two** J-lang **I**ntegrated framework

JI2 (仮称)

だれかいい名前をください

ディレク トリ 構成

ごめんなさい

とりあえず起動してみる

ごめんなさい

処理の流れ

```
var handlers = [  
  {  
    url: "/xx/yy/{id}",  
    fooController: function() {  
      Java.type(...).foobar // java呼出し  
    }  
  },  
]
```

Nashorn!!!!!!!!!!!!

```
public class Bootstrap {  
  public static void main(..) {  
  }  
  public static void coreHandler() {  
    ScriptEngineManager em = new ScriptEngineManager();  
    ScriptEngine engine = em.getEngineByName("nashorn");  
    engine.eval("...");  
  }  
}
```


このフレームワークの思想

サーバーサイドFramework(rails, play,)が

npmバインディングや、フロントエンドとの
インテグレーション機構をもつのではない。

npm文化圏に、サーバーサイドFWが来い！

[参考](<http://dic.nicovideo.jp/a/〇〇が来いの一覧>)

まとめ

“高度に発達した*javascript* は
*java*と見分けがつかない”

—*Richard D. Philips*—

というか、*java/js*が混雑した
カオスアーキテクチャになってしまった...
Toy Productです。くれぐれもご注意ください

まじめな振り返り

- nashornの使い方の勉強にはなった。
- npmの勉強にもなった。
- clojureやるならclojure + clojurescript
を使う方が100倍良さそう
(jvm資産使えるし)

今後の展開

- toy project ではあるが、nashorn呼び出しするWAFでどこまでreq/secでるのか、計測 & 改良してみたい
- コントローラ関数にjsdoc情報からjson scheme生成
- ji2-middleware的なものを考えて、コントローラに処理が渡るまえのフローを用意しないと実用的ではない。(認証系,エラー時のcatchレスポンスなど)
- http 1.1 以外のプロトコルにどう対応するか考えたい(http2.0, websocket, ...)