

Maple 祭り その1

株式会社アイサイト 熊倉 洋介

(kumakura@isite.co.jp)

Maple に至るまでの
お話

2003年10月

「何も使ってない」時代

- Smarty
- 自前 Controller
 - if, else
 - switch, case

2004年1月

「HTML_QuickForm」時代

Form の Validate, 再入力処理が
楽できそうだった。

- PEAR HTML_QuickForm
- PEAR HTML_QuickForm_Controller
- Smarty

2004年中頃～ 「Mojavi」時代

それでも Mojavi 内で QuickForm を使ったり使わなかったり。

- Mojavi
- PEAR HTML_QuickForm
- Smarty

2005年1月

「Maple」利用開始

- Bye QuickForm !!!!
 - 目的のフォームを作るためのコーディングで悩むのはもう勘弁。
- Bye Mojavi ?
 - ケースバイケース。使う場面では使う。
 - さよならしたつもりはない。

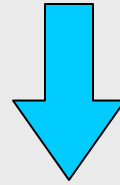
「Maple」の理由

- ・コーディング量の減少
 - 「良くある処理」が準備されている
 - Configuration Base
- ・薄い（色・層）
 - 必要以上にやってくれない
 - やってくれなくていい

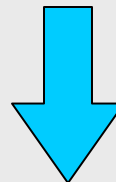
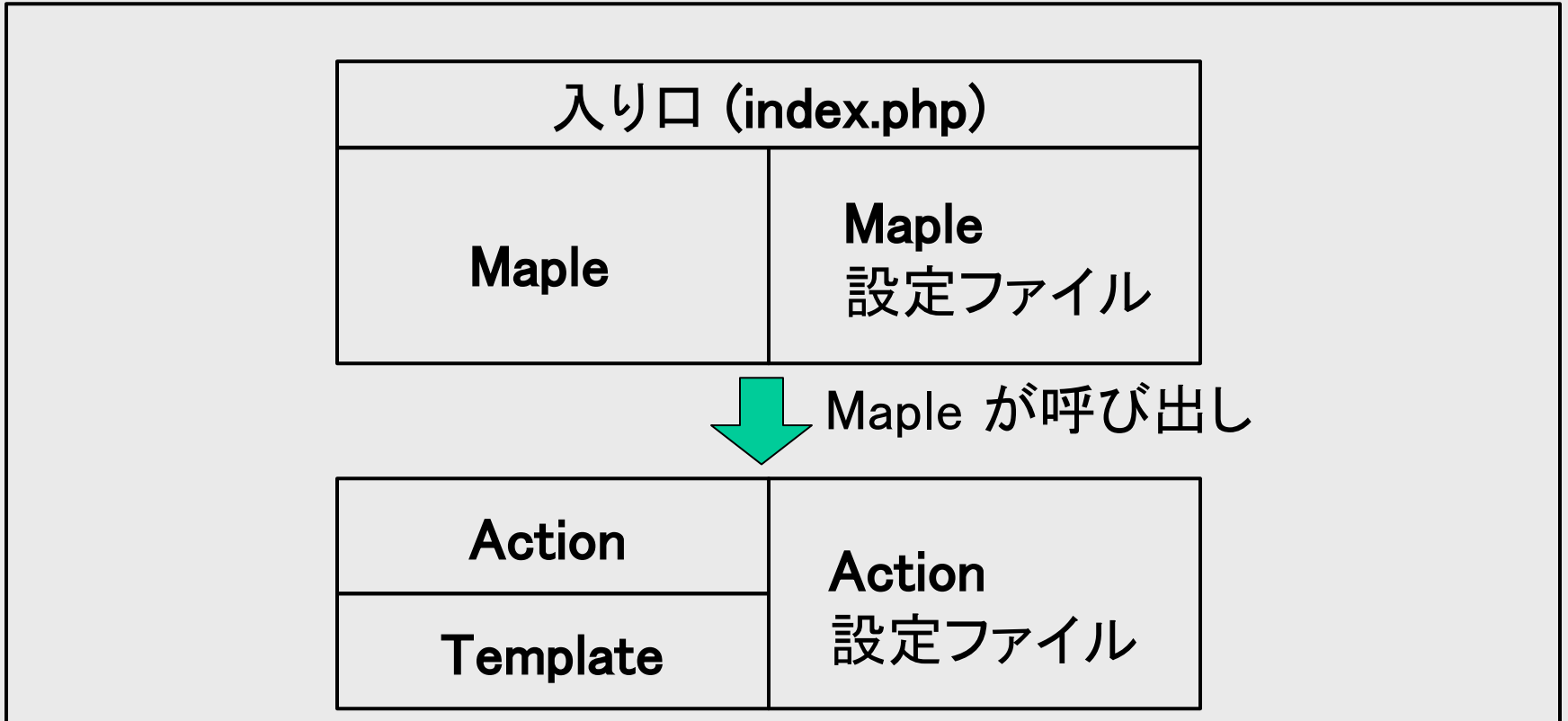
Maple 導入部の お話

リクエスト

- ・Maple アクションを指定
- ・アプリケーションに関連する値



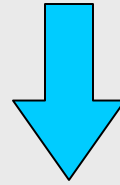
アプリケーション



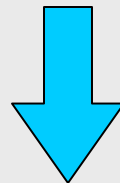
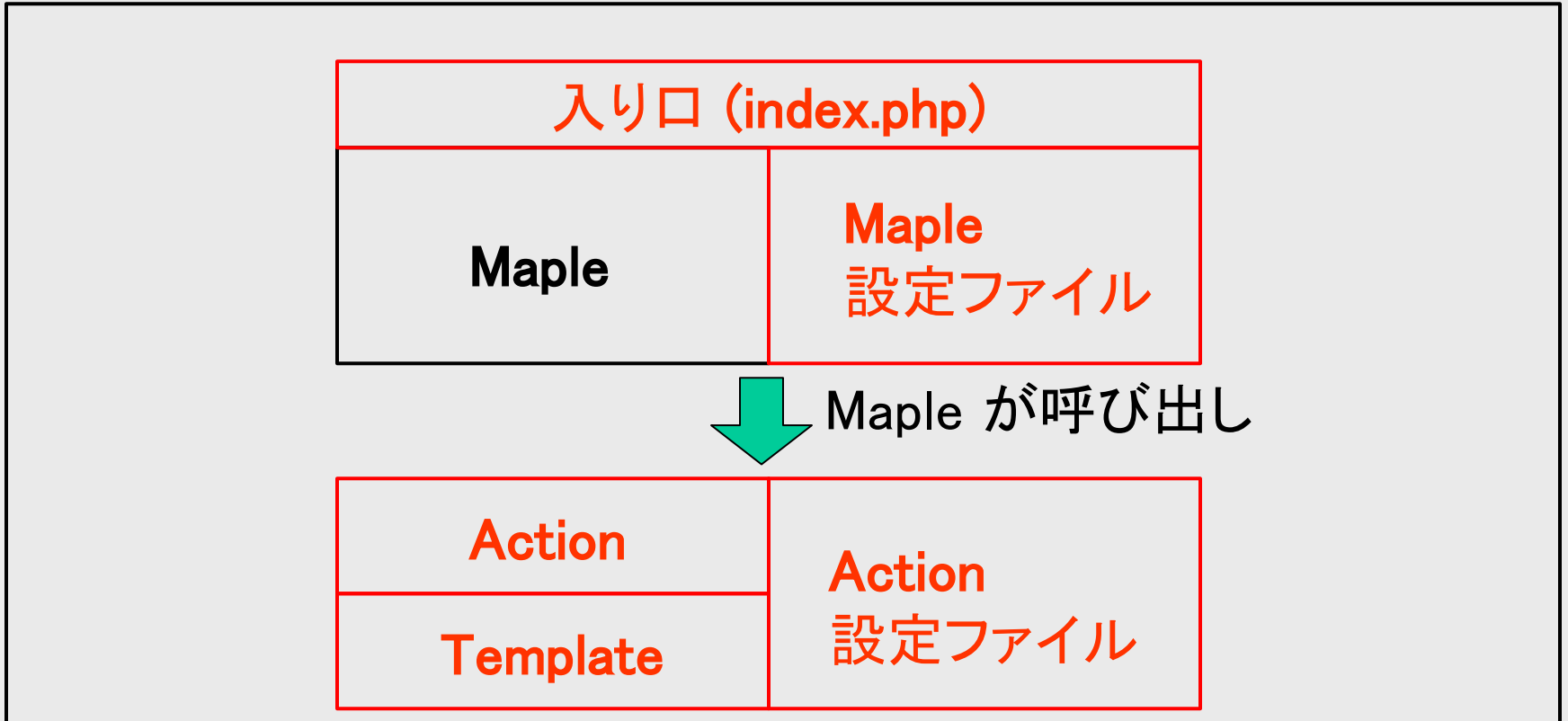
レスポンス

リクエスト

- ・Maple アクションを指定
- ・アプリケーションに関連する値



アプリケーション



レスポンス

基本コーディング内容

- 入り口 (index.php)
- Maple設定ファイル
- Action
- Template
- Action設定ファイル

入り口 (index.php)

- ・ Maple を起動する

Maple 設定ファイル

- Maple を動かすための値を設定
- 基本触る個所はほとんど無し
(自分で好きなディレクトリ構成にしたい時のみ)
- 一番触る可能性があるのは
文字コードの指定

Maple 設定ファイル

- ・文字コードの指定

PC 用サイト → 全部 EUC-JP

Mobile 用サイト →

INPUT_CODE, OUTPUT_CODEは
Shift_JIS, 残り EUC-JP

- ・コーディングはすべて EUC-JP

Action

- Step by Step !

1. リクエストがあれば受け取る
2. 処理があれば処理する
3. 描写する内容を指定する

Action

```
<?php
class Example {
    // 1. リクエストがあれば受け取る
    var $foo;

    function execute() {
        // 2. 処理があれば処理する

        // 3. 描写する内容を指定する
        return "success";
    }
}
?>
```

Template

- Smarty で書く
- Template 変数とかなかったら HTML
- 次バージョンでは
Smarty 以外の選択肢ができるはず

Action 設定ファイル

- Maple の機能を使う宣言
 - Session
 - Token
 - Request Validate
 - Request Convert
 - ex...
- テンプレートファイルの指定

Action 設定ファイル (ini)

例: Example アクションの設定ファイル

[View]

success = example.html

First Step

Sample を動かす

- ・Maple の基本プレゼンテーション層機能
 - Request Validate
 - Request Convert
 - Session
 - Token

が全部動いている。

Second Step

Sample をいじる

- Form を増やしてみる
- アクション内でリクエスト値を覗いてみる
- 別のページを挿入してみる

Web アプリケーション開発の お話

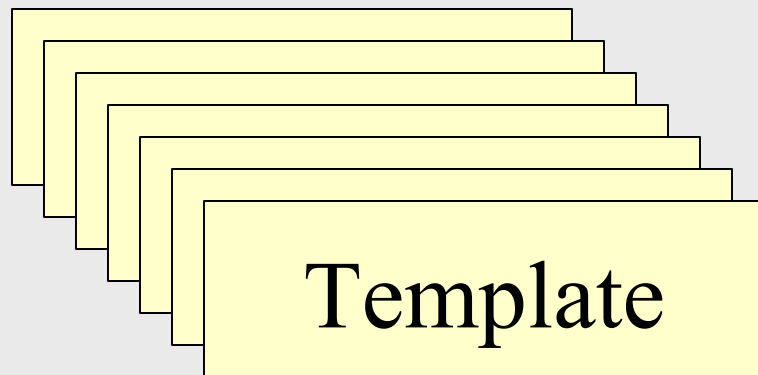
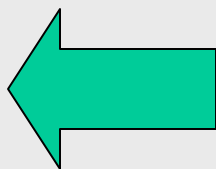
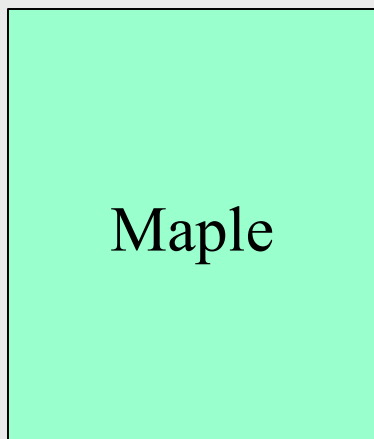
Maple Action

```
<?php
class PageN {
    function execute() {
        return "success";
    }
}
?>
```

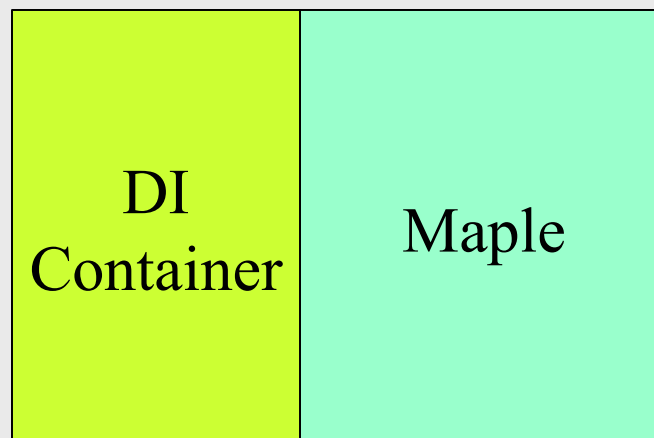
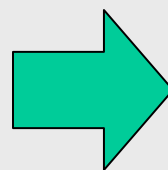
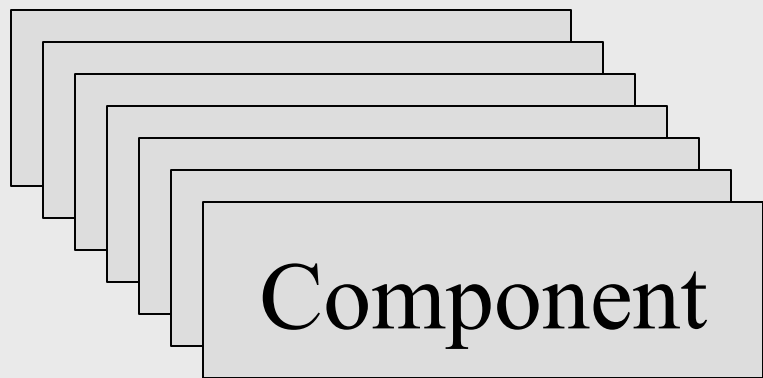
Action 設定ファイル

[View]

success = PageN.html



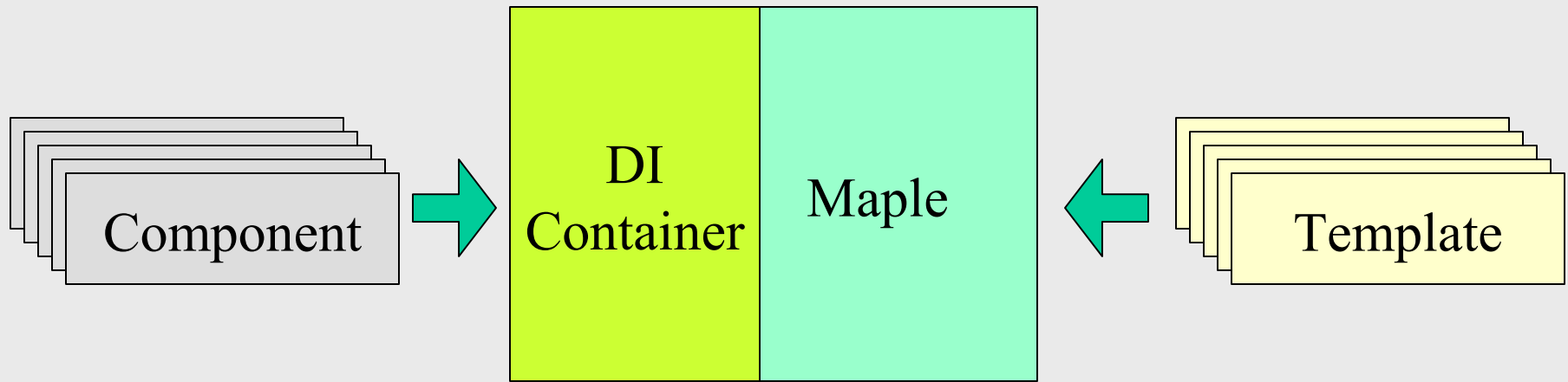
HTML or Smarty
で作成



Maple 一切関係なし
(厳密には関係あり)

要件の
実装を続ける

デザイナーさん
頑張る



要件とする機能が
正しく動くことが重要

Test First

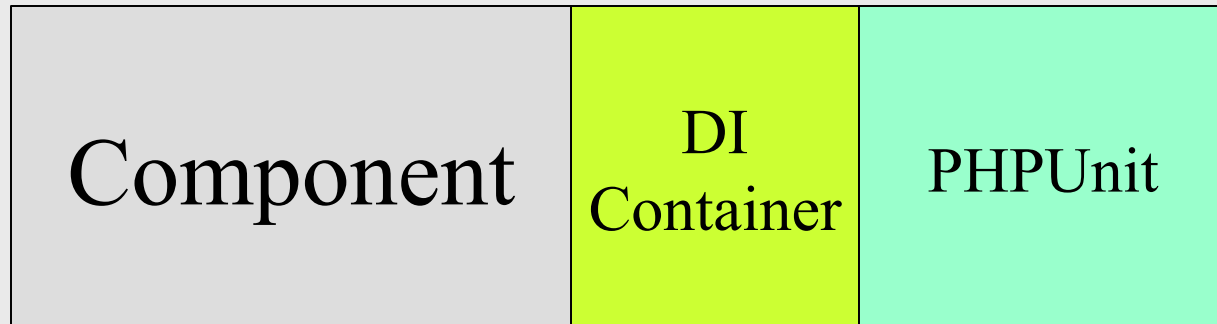
- 期待される機能（要件）を実装する
⇒ **Component**
- **Component が出来たら Test**
 - **success**
 - **fail**

Test First

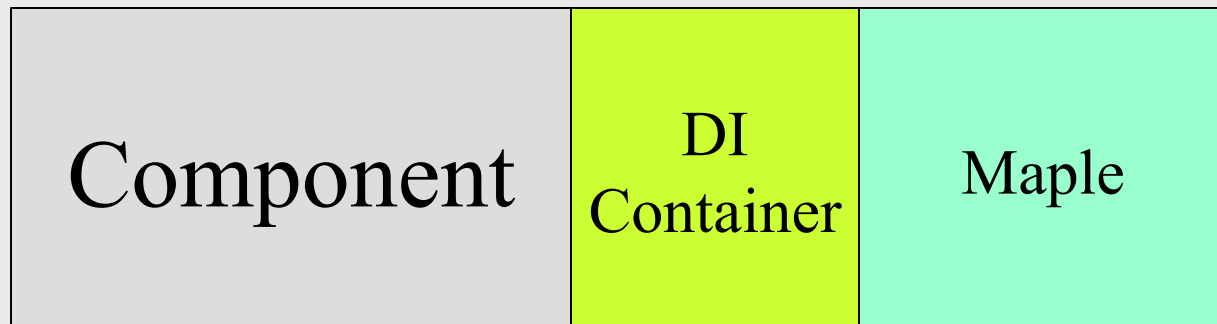
- ・ Test に必要なもの
 - Component
 - Controller (PHPUnit)
 - Component と Controller を繋ぐ為のもの (DIContainer)
 - CLI PHP

Test First

これで動けば...

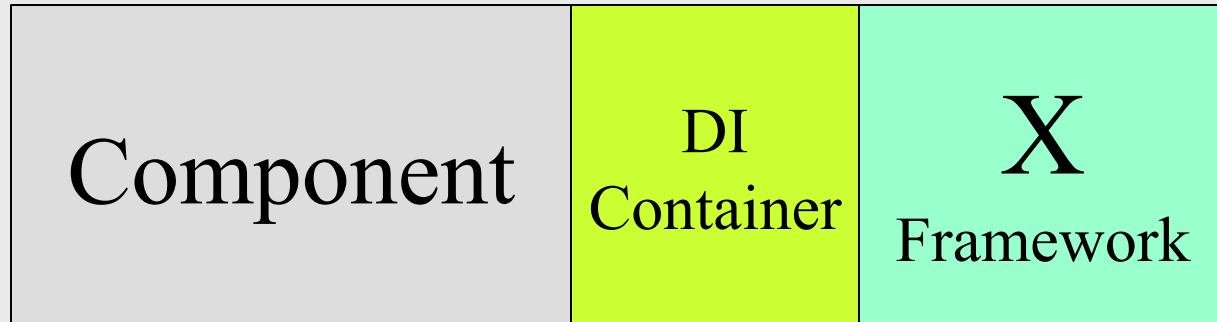


これでも動く



Test First

200X年、これでも動くはず



Test First

コマンドラインで DB アクセスまでテスト
(もちろん DB など存在しない)

Fake - DB mock	imoDI	Cloth - TestRunner
Component ×機能分		PHPUnit

Web アプリケーション開発

- ・要件分の Component を作る
 - ビジネスロジック
 - データベースアクセス
 - アプリケーション・ユーティリティ

この際「Web から使うには…」という意識をしない。
あくまで要件に沿った機能をコーディング。
加えていきなり完成形を作ろうと考えない。

- ・Test

Web アプリケーション開発

- Maple の中で Component を使う
- with DIContainer
 - 単純明快な Step
 1. Component を詰め込む
 2. 使う場面で引っ張ってくる
 3. 使う
- 問題が出れば Component 見直し, Test
- 必要なものが出れば Component 作成, Test

Maple Action

- Step by Step !

1. リクエストがあれば受け取る
2. 処理があれば処理する
3. 描写する内容を指定する

Maple Action

- Step by Step !

1. リクエストがあれば受け取る

2. 処理があれば処理する

= Component の実行

3. 描写する内容を指定する

Maple Action

```
function execute()  
{  
    // 2. 処理があれば処理する  
    $this->logic->store($this->foo);  
  
    // 3. 描写する内容を指定する  
    return "success";  
}
```

Web アプリケーション開発

- ・リクエストを受け付けてみる - Maple
- ・画面遷移を確認してみる - ブラウザアクセス
- ・意地悪な操作を試してみる - ブラウザアクセス

48時間の

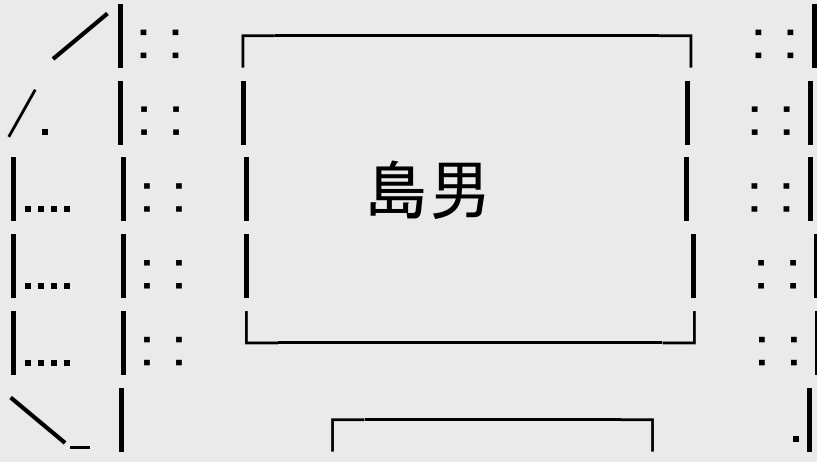
お話

「恋におちたら」

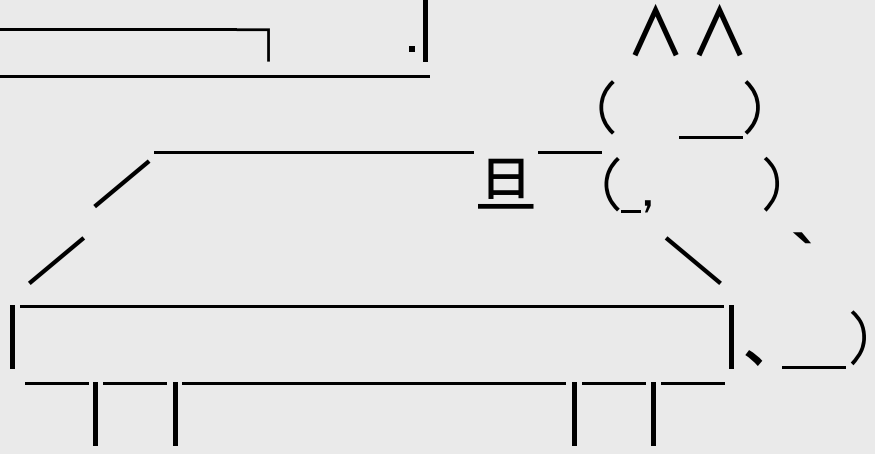
鈴木島男

48時間で
オークションサイト
作成

しかもリッチクライアント仕様



ま、所詮
TV だしな。



その半年後。

とあるサービスの立ち上げ計画がスタート。

しかし一向にその仕様が固まらない。

スタート予定7日前の会議…

^^
(; ▽)
/U U
し—J

もうすぐ一週間切りますよ…

^_^
(`▽´) どうしょっか

|| \ . | | |
|| \ ^_^ (^ ▽ ^)
|| (^ ▽ ^) ~ \ ^_^
/ \ こうしょっか \ | (`▽´)
| \ \ (= ^) \ | | |
| \ \ ^_^ (^ ▽ ^) /

内容は決まったものの、2日後(48時間)の
スタート予定は変わらず。つまり48時間以内に
(小規模ながらも)Web アプリケーションを
作ることになってしまったのです。

((^ ^))

ファビョー ||| ||| ーン!

(^ ;... ^ _ ^

(^ .c , \ # ` D `) <なんでやねん!

(^)人 \ \、从

从ノ:(,,フ .ノ>コ

人从;...レ'ノ;...从人

1. 設計

- ホワイトボードを前にスタンドアップミーティング
- スピーディに、簡潔に
- 一人でやらず、二人以上でやる

| Database Table

foo.

Logic

bar..

hoge

baz...

^^

^_^

(; D°)

(`v`)

| つつ

()

~. |

| | |

し`J

ITEMAN ()(/

< くまっちも大変だなあ。
| (他人事)

2. Component 作成

- 決めたルールに沿って順に行う

(※この時はまだ前述の手法にまでは至っていなかった)

- 光の速さでタイピング

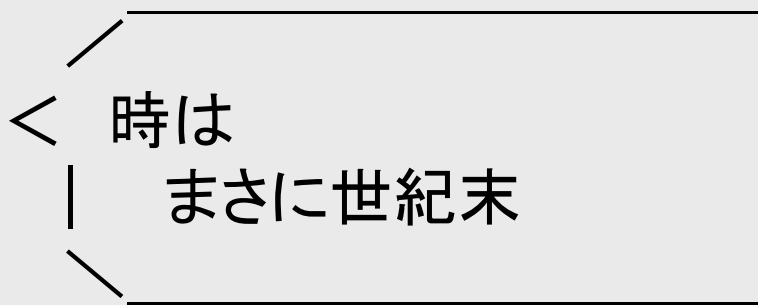
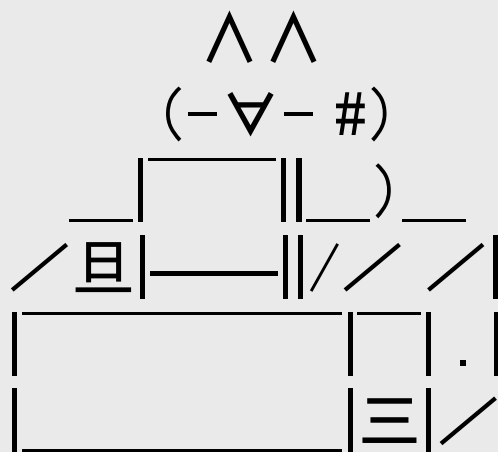
 r ^ ^ \
 (^ ^) ホッポー
カタカタ | | /
 ^ ^
 (. v . #)
 | | | |)
 / 旦 | | | / / /
 | | | | | . |
 | | | | | 三 | /

< オイラを怒らせると
大変なことになるお！

3. Template 作成

- リクエスト非受付から開始
- 画面遷移が完成すればリクエスト受付
- 光の速さでタイピング

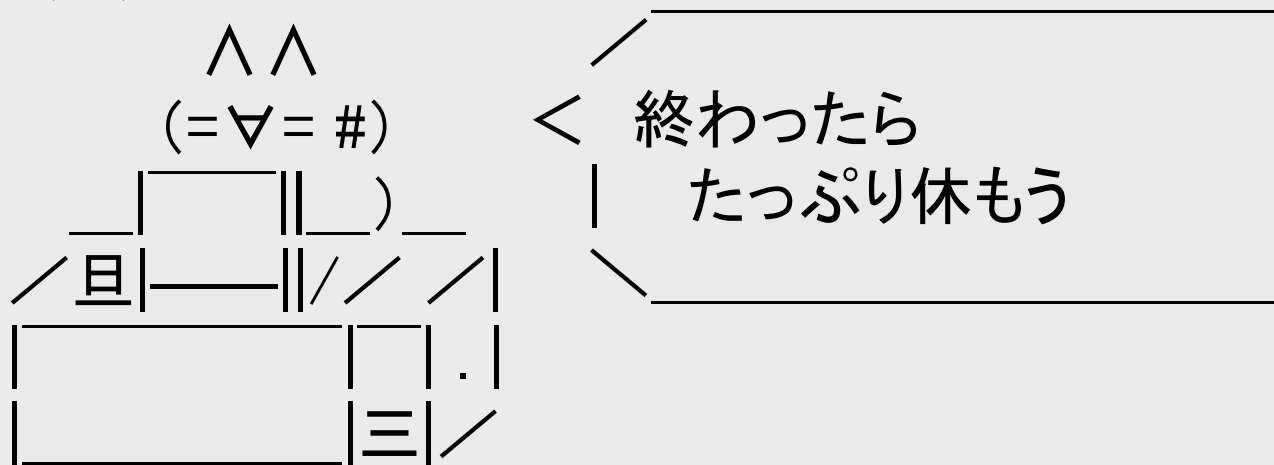
カタカタ



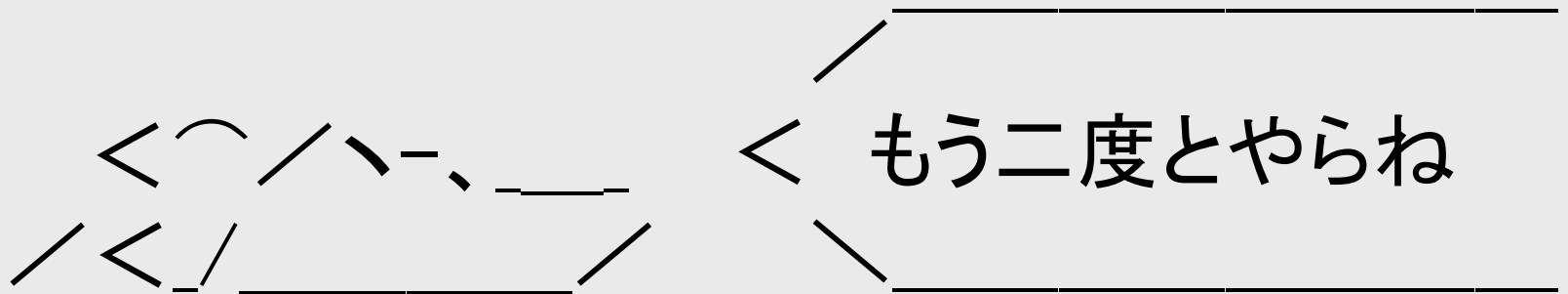
4. Component 繋ぎこみ

- 正しくリクエストが受け付けれる Action から Component を適応していく
- ブラウザ越しにひたすら Test
- 驚くべき勢いでタイピング

カタカタ



怒涛のコーディングを行い、結果
13,000行ものコードを書き上げ、
無事完成させたのでした。



決め手

- **Maple に備わる機能**
(特に **Validate, Convert**)
- **Component** 作成時自らに
コーディングルールを定める。
無駄な思考を一切排除。
順に追ってコーディングしていただくだけ。

後日

ページ内容を大幅に変えよう
と思うんだけど(仕様変更)

Λ_Λ
(^ ▽ ^)
()
| |
() ()

< ^ / \ - - -
/ < / _____ /

時間をかけてきっちりやりました
(使える Component は当然そのまま)

<http://hatotech.org/kumatch/>