

Ruby on Rails 講義

第18回

一番ちいさなRailsアプリつくり

五十嵐邦明

twitter : igaiga555

<http://www.facebook.com/kuniaki.igarashi>



2013.10.17 at 一橋大学

ニフティ株式会社寄附講義
社会科学における情報技術と
コンテンツ作成IV

五十嵐邦明

講師

株式会社 spice life

twitter: igaiga555

<https://github.com/igaiga/>

<http://www.facebook.com/kuniaki.igarashi>



濱崎 健吾

Teaching Assistant
fluxflex, inc(米国法人)

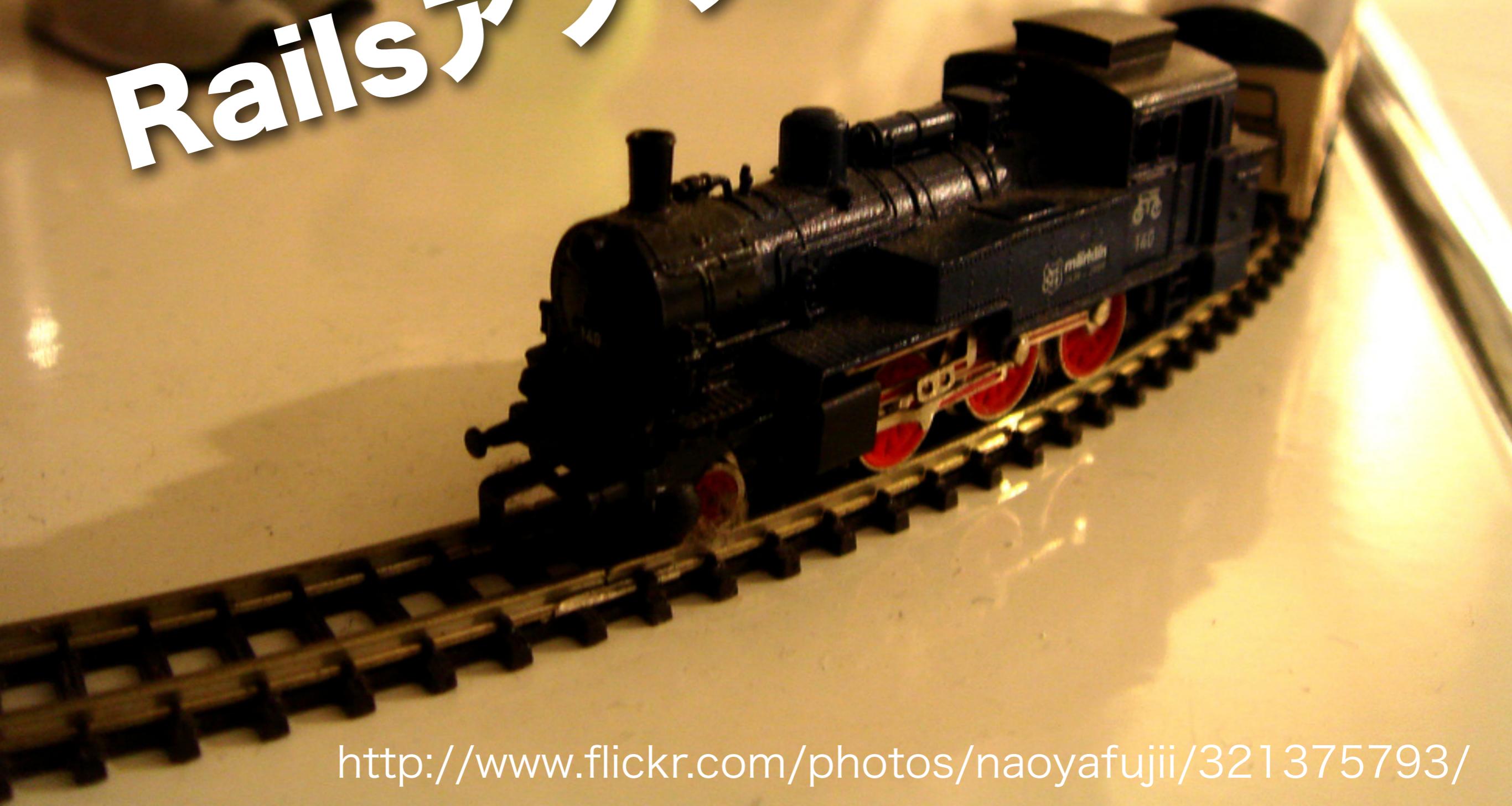


twitter: hmsk

<https://github.com/hmsk/>

<http://www.facebook.com/hamachang>

今日やること
一番ちいさな
Railsアプリづくり



一番小さなRailsアプリつくり

ブラウザにhello world を表示するアプリを作ります。

1. Railsアプリをつくる

```
$ rails new helloworld
```

2. RailsRootへ移動してページのひながたをつくる

```
$ cd helloworld
```

```
$ bundle exec rails g controller hello index
```

g は generate の略です。コマンドの意味は後半で説明します。
打ち間違えて削除したい場合は、g の代わりに d で削除できます。

3. Webサーバを起動する

```
$ bundle exec rails s
```

※終了はCtrl+c (終了しない場合、Ctrl+PAUSEを試してください。)

s は server の略。

4. ブラウザから以下のURLにアクセスする

```
http://localhost:3000/hello/index
```

4. ブラウザから以下のURLにアクセスする

http://localhost:3000/hello/index

Hello#index

Find me in app/views/hello/index.html.erb

こんな画面が出ればOKです。

この表示を Hello world! に変えてみましょう。

5. Hello world! を表示させる

app/views/hello/index.html.erb

```
<h1>Hello#index</h1>
```

```
<p>Find me in app/views/hello/index.html.erb</p>
```

となっているところを以下のように修正

```
<h1>Hello world!</h1>
```



さきほど同様にブラウザで以下へアクセス

<http://localhost:3000/hello/index>

こんな画面が出ればOKです。

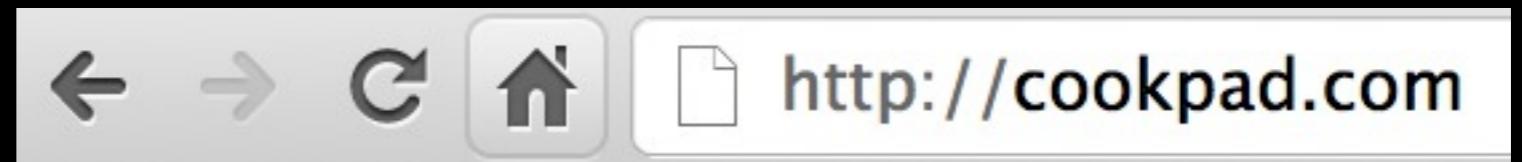
Hello world!

解説

Webアプリの基本動作 (一部復習)

Webアプリの基本動作

ブラウザにURLを
入力してEnter



リクエスト

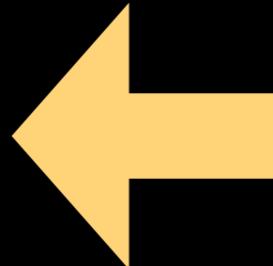
URL : http://cookpad.com/

Web Server

Web App

Browser

http://cookpad.com/



リクエストが飛びます。

Webアプリの基本動作

レスポンスとしてHTMLが返ってくる

リクエスト

URL : <http://cookpad.com/>

Web Server

Web App

レスポンス
HTML

Browser

<http://cookpad.com/>



HTML

HyperText Markup Language

Webページを記述するための言語

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2
3 <html>
4 <head>
5 <script type="text/javascript">var __rendering_time = (new Date).getTime();</script>
6 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
7
8 <meta name="robots" content="noarchive" />
9
10 <link rel="alternate" media="handheld" type="text/html" href="http://m.cookpad.com/" />
11 <link rel="apple-touch-icon" href="/images/device/apple-touch-icon.png" />
12
13 <title>レシピ検索No.1／料理レシピ載せるなら クックパッド</title><meta content=",レシピ,簡単,料理,COOKPAD,くっくぱっど,recipe" name="keywords" />
content="日本最大の料理レシピサイト。125万品を超えるレシピ、作り方を検索できる。家庭の主婦が実際につくった簡単実用レシピが多い。利用者は
を公開することもできる。" name="description" />
14
15
16
17
18
19 <script src="/javascripts/jpack.js?1340964911" type="text/javascript"></script>
20
21
22 <script type="text/javascript">AsyncView.load({"specify":["kondate_ext_kondate_unit"],"assigns":"{\\\"kondate_ext_kondate_unit\\\":{}}
,"controller_name":"top","action_name":"top"});</script>
```

ブラウザでHTML表示

右クリックからソースを表示



ブラウザの主な機能

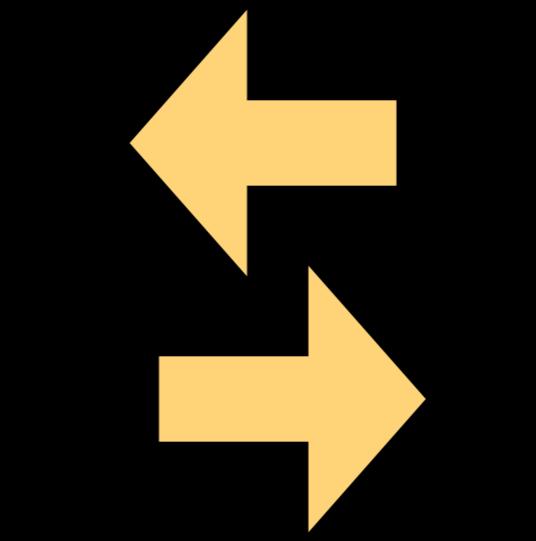
1. リクエストを飛ばす
2. レスポンスでもらったHTMLを人が見る形式で表示する

リクエスト

URL : <http://cookpad.com/>

Web Server

Web App



レスポンス
HTML

Browser

<http://cookpad.com/>



インターネットへのアクセス

Internet

Web Server

Web App

Browser

リクエスト

レスポンス
HTML

WebServerはネットの向こうにある

開発時の構成

自分のPC

Web Server

Web App

URL
リクエスト

Browser

レスポンス
HTML

1台のマシンの中で開発、アクセス可能です。

Rails アプリ

自分のPC

Web Server

Web App
Rails App

URL
リクエスト

Browser

レスポンス
HTML

ブラウザで動作
確認しながら進
めます。

作っているのはこの部分

今回つくったRailsアプリの動作

リクエスト

▶ URL : <http://localhost:3000/hello/index>

Web Server

Rails App

```
<!DOCTYPE html>
<html>
<head>
<title>HelloWorld</title>
<link data-turbolinks-track="true" href="/assets/application.css?body=1" media="all" rel="stylesheet" />
<link data-turbolinks-track="true" href="/assets/hello.css?body=1" media="all" rel="stylesheet" />
<script data-turbolinks-track="true" src="/assets/jquery.js?body=1"></script>
<script data-turbolinks-track="true" src="/assets/jquery_ujs.js?body=1"></script>
<script data-turbolinks-track="true" src="/assets/turbolinks.js?body=1"></script>
<script data-turbolinks-track="true" src="/assets/hello.js?body=1"></script>
<script data-turbolinks-track="true" src="/assets/application.js?body=1"></script>
<meta content="authenticity_token" name="csrf-param" />
<meta content="5OxdFgeKzp2JlFeKGx9u7VJDt29KS1KSP1GmwbISpDQ=" name="csrf-token" />
</head>
<body>
<h1>Hello World!</h1>
</body>
</html>
```

HTMLを作って返す

リクエスト



レスポンス
HTML

Browser

URL入力欄に
アドレスを入力して
Enterを押すと
リクエストが飛びます

Hello World!

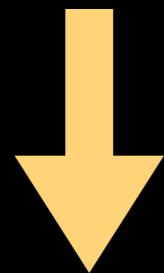
ブラウザの表示

Railsでの 開発の進め方

Railsでの開発の進め方

Railsでの基本的な開発の進め方は
以下ののようなサイクルになります。

ひな形になるファイル
(ソースコードや設定ファイル) の生成



この2つの繰り返し



そのアプリ用にソースコードを変更、追記

さきほど作ったアプリも
実はこの流れで作ってました

一番小さなRailsアプリつくり

hello world を表示するアプリを作ります。

1. Railsアプリをつくる

```
$ rails new helloworld
```

←生成

2. RailsRootへ移動してページのひながたをつくる

```
$ cd helloworld
```

```
$ bundle exec rails g controller hello index
```

←生成

g は generate の略です。コマンドの意味は後半で説明します。

打ち間違えて削除したい場合は、g の代わりに d で削除できます。

3. Webサーバを起動する

```
$ bundle exec rails s
```

※終了はCtrl+c (終了しない場合、Ctrl+PAUSEを試してください。)

s は server の略。

4. ブラウザから以下のURLにアクセスする

```
http://localhost:3000/hello/index
```

5. Hello world! を表示させる コード変更

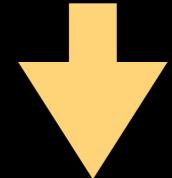
app/views/hello/index.html.erb

```
<h1>Hello#index</h1>
```

```
<p>Find me in app/views/hello/index.html.erb</p>
```

となっているところを以下のように修正

```
<h1>Hello world!</h1>
```



さきほど同様にブラウザで以下へアクセス

<http://localhost:3000/hello/index>

こんな画面が出ればOKです。

Hello world!

では、どんなものが
生成されているのでしょうか。

生成されるフォルダ・ファイル

\$ rails new helloworld

アプリを作るコマンドである

rails new [アプリ名]

を実行すると、次のページのようなファイル・
フォルダが生成されます。

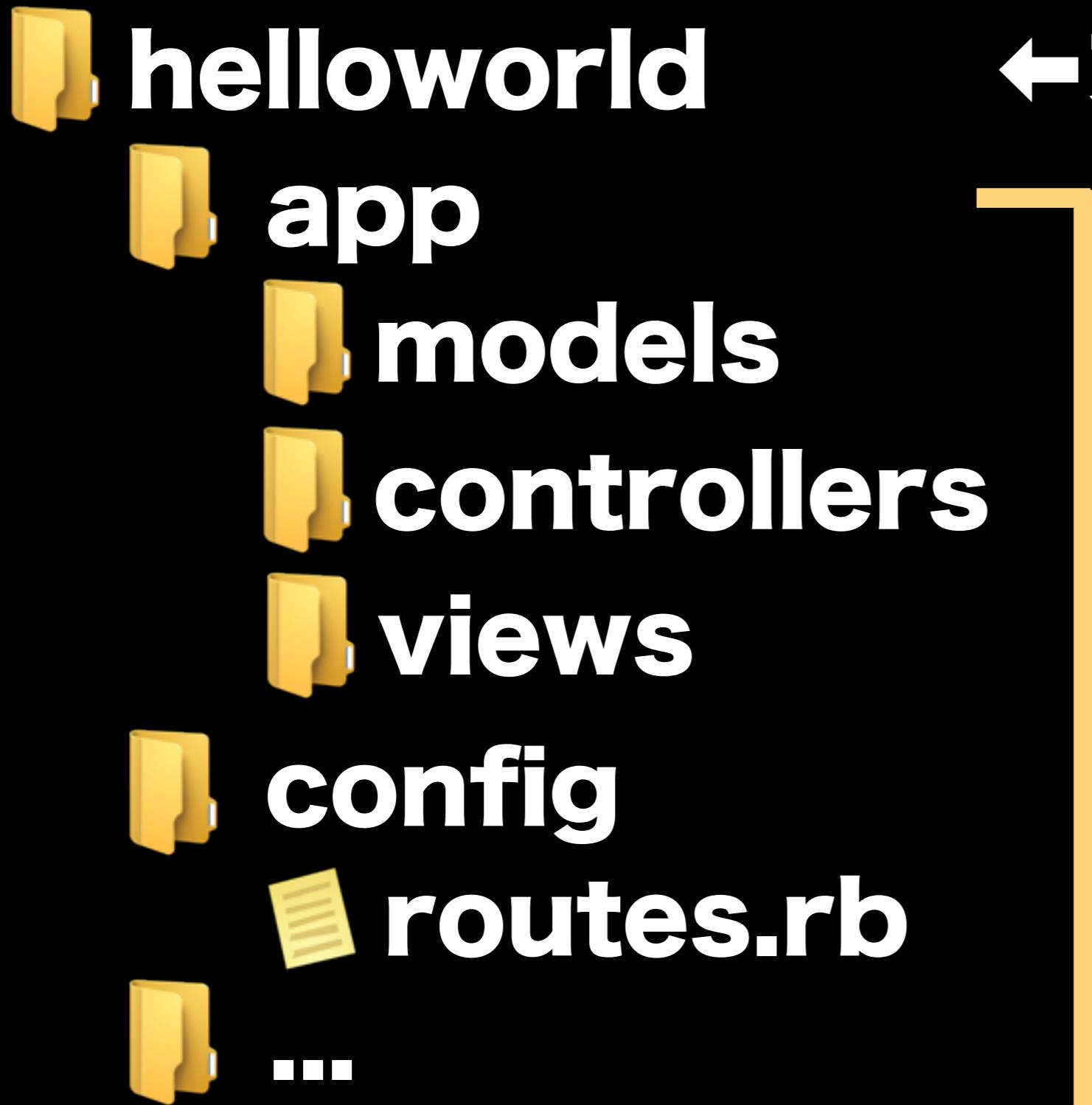
```
create README.rdoc
create Rakefile
create config.ru
create .gitignore
create Gemfile
create app
create app/assets/images/rails.png
create app/assets/javascripts/application.js
create app/assets/stylesheets/application.css
create app/controllers/application_controller.rb
create app/helpers/application_helper.rb
create app/mailers
create app/models
create app/views/layouts/application.html.erb
create app/mailers/.gitkeep
create app/models/.gitkeep
create config
```

←この画面のときに生成してます。

この状態でもまだ画面はないですが
Webアプリとして動作します

```
$ rails new helloworld
```

以下のフォルダ・ファイルが生成されます。(一部のみ抜粋)



←アプリ名のフォルダ

Railsアプリ
の基本的な
フォルダと
ファイル群

フォルダそれぞれの役割は
今後説明していきます。

rails g controller コマンド

```
$ bundle exec rails g controller hello index
```

次は `rails new` に続いて実行した
`rails g controller` コマンドを見てみましょう。

`controller` と `view` などを作る
`rails g controller` コマンドを実行すると
次ページのファイル群が生成されます。
今回のコマンドはURLが `/hello/index` であるペ
ージを表示するためのファイル群を生成します。

```
$ bundle exec rails g controller hello index
```

以下のフォルダ・ファイルが生成・変更されます。(一部のみ抜粋)



app

models

controllers ←いろいろな処理を行う箇所

hello_controller.rb

views

←表示を行う箇所

hello

index.html.erb

config

routes.rb

http://localhost:3000/hello/index

/hello/index ページを
表示するためのファイル群を
生成します。

←URLとコントローラの対応表

rails g コマンド

ひながたになるファイルを生成するコマンド

```
$ bundle exec rails g controller hello index
```

generateする種類 アクション名
 コントローラ名

※打ち間違えて削除したい場合は、 g の代わりに d で削除できます。

routes、コントローラ、ビューのファイルほかを生成します。

- ─ config/routes.rb
- ─ app/controllers/hello_controller.rb
- ─ app/views/hello/index.html.erb ほか

※ちなみに、前にやった scaffold もgenerateの種類の1つです。

次は
Railsアプリ内で
Rubyのコードを
書いてみます。

6. ViewでRubyのコードを実行させる

Viewである `.html.erb` ファイルにはHTMLを書きますが、Rubyのコードを実行し、結果を埋め込むこともできます。コードを埋め込む場合は `<%=` と `%>` で囲みます。

以下のファイルを変更して、ブラウザからアクセスしてみてください。

`app/views/hello/index.html.erb`

```
<h1>Hello world!</h1>
<p>現在時刻 <%= Time.now %> </p>
```

`http://localhost:3000/hello/index`

Hello world!

現在時刻 2012-07-10 21:48:56 +0900

← `Time.now` の実行結果

7. ControllerでRubyのコードを実行させる

Viewは表示をする役割を受け持つ部分です。表示する内容を計算したり作成したりするのはControllerなど、Viewの前段で行うしきたりです。

以下のControllerのファイルを変更して、ブラウザからアクセスしてみてください。

app/controllers/hello_controller.rb

```
class HelloController < ApplicationController
  def index
    @time = Time.now
  end
end
```

app/views/hello/index.html.erb

```
<h1>Hello world!</h1>
<p>現在時刻 <%= @time %> </p>
```

Hello world!

現在時刻 2012-07-10 21:48:56 +0900

<http://localhost:3000/hello/index>

同じ結果が表示されればOKです。

解説

ブラウザからアクセスしたときのRailsアプリの動作

リクエスト

- ▶ URL : `http://localhost:3000/hello/index`
- ▶ HTTPメソッド : GET

Web Server

Rails App

Routes

Controller

View

次ページでここを詳しく説明します

リクエスト

レスポンス
HTML

Browser

URL入力欄に
アドレスを入力して
Enterを押すと
リクエストが飛びます

Hello World!

Railsアプリがリクエストを受けて レスポンスを返すまでの流れ

Routes, Controller, View の3つの部品を通過します。

リクエスト

- ▶ URL : `http://localhost:3000/hello/index`
- ▶ HTTPメソッド : GET

Rails App

Routes

routes.rb

Controller

hello_controller.rb indexアクション

View

hello/index.html.erb

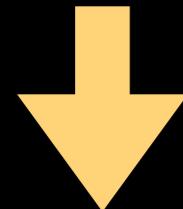
レスポンス : HTML

Routes

リクエストのURLとHTTPメソッドに応じて
処理を行う先を決めます。
リクエストとController のアクションとの対応表です。

リクエスト

- ▶ URL : `http://localhost:3000/hello/index`
- ▶ HTTPメソッド : GET



routesが持っている対応表を見て呼び出すアクションを決定

HelloControllerのindexアクション

※routesからアクセスされるControllerのメソッドのことを
「アクション」と呼びます。

Routes

Routesの対応表の内容を見る場合は以下へアクセスします。
`http://localhost:3000/rails/info/routes`

Helper	HTTP Verb	Path	Controller#Action
<u>Path / Url</u>		Path Match	
hello_index_path	GET	/hello/index(.:format)	hello#index

ざっくり説明すると、`/hello/index` へHTTPSメソッドGETでアクセスされたときに HelloController の indexアクションに処理が移るという意味です。

リクエスト

▶ URL : `http://localhost:3000/hello/index`

▶ HTTPメソッド : GET

HelloControllerの
indexアクション(メソッド)へ処理が渡る

Controller

さまざまな処理を行い、つぎの View に処理を渡します。

HelloController の ソースは
app/controllers/hello_controller.rb です。

app/controllers/hello_controller.rb

```
class HelloController < ApplicationController
  def index
    @time = Time.now
  end
end
```

ポイントは、Viewにデータを渡す際は、インスタンス変数(@はじまりの変数)を使うことです。ここでは、@time が次のViewへ渡されます。

次は、特に指定がない場合はコントローラ名/アクション名的なviewへ移動します。
→次はViewである app/views/hello/index.html.erb に処理が渡ります。

View

ユーザーの目に届く部分（ここではHTML）をつくります。

app/views/hello/index.html.erb

```
<h1>Hello world!</h1>
<p>現在時刻 <%= @time %> </p>
```

index.html.erb は、HTMLのもとになるファイルです。Rubyのコードを実行した結果を埋め込むことができます。（このようなファイルをテンプレートと呼びます。）埋め込まれたRubyのコードを実行した結果でHTMLを作ります。

Hello world!

現在時刻 2012-07-10 21:48:56 +0900

← Controllerで代入した
@time の中身が表示される

（実際には、Viewテンプレートファイルから作られたHTMLに、Railsがその他の加工を加えてレスポンスとして送出します。）

Controllerでコードを実行して、結果をViewへ渡すときはインスタンス変数を使う

リクエスト

- ▶ URL : `http://localhost:3000/hello/index`
- ▶ HTTPメソッド : GET

Rails App

Routes

Controller コードを実行して`@time`に代入

結果を渡すにはインスタンス変数(`@はじまり`)を使う

View

`@time`の中身を表示する

レスポンス : HTML

Hello world!
現在時刻 2012-07-10 21:48:56 +0900

參考資料

書式

Rubyコード

`puts "abc"`

実行結果

"abc"

実行結果

文中では `#=>` で書きます

`p 1+2 #=> 3`

shellコマンド

`$ ls`

自習用Rails資料

特に教科書は使いませんが、自習用には以下の資料をお勧めします。

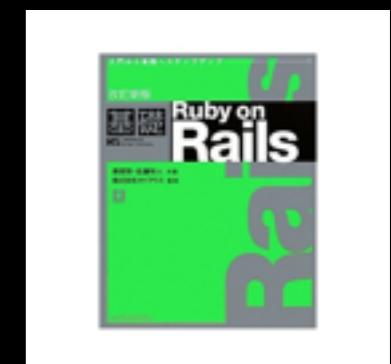
- ▶ **RailsTutorial (web)**
- ▶ **<http://railstutorial.jp/?version=4.0>**

- ▶ **Rails Guide (web)(English)**
- ▶ **<http://guides.rubyonrails.org/>**

- ▶ **RailsによるアジャイルWebアプリケーション開発 第4版**
- ▶ **<http://www.amazon.co.jp/dp/4274068668/>**



- ▶ **改訂新版 基礎Ruby on Rails**
- ▶ **<http://www.amazon.co.jp/dp/4844331566/>**



- ▶ **たのしいRuby**
- ▶ **<http://www.amazon.co.jp/dp/4797357401/>**



講義資料置き場

過去の資料がDLできます。

<https://github.com/hitotsubashi-ruby/lecture2013>

or

http://bit.ly/iga_ruby_2013

雑談・質問用facebookグループ

<https://www.facebook.com/groups/hitotsubashi.rb>

- 加入/非加入は自由です
- 加入/非加入は成績に関係しません
- 参加者一覧は公開されます
- 参加者はスタッフ(講師・TA)と昨年、今年の受講者です
- 書き込みは参加者のみ見えます
- 希望者はアクセスして参加申請してください
- 雑談、質問、議論など何でも気にせずどうぞ~
- 質問に答えられる人は答えてあげてください
- 講師陣もお答えします
- 入ったら軽く自己紹介おねがいします