

Ruby on Rails 講義 第20回 CRUD基礎

五十嵐邦明

twitter : igaiga555

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



2013.10.31 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>

休講連絡

11月7日, 21日の講義は
休講です。

課題チェック用の付箋の書き方

上の方に学籍番号と名前を書いてください

学籍番号

名前

※この辺に講師陣がクリアした課題番号を書いていきます。

CRUD基礎

<http://www.flickr.com/photos/uicdigital/8000177297/>

演習1. 本の感想などを管理するアプリを作ってみましょう。

```
$ rails new books_app
```

```
$ cd books_app
```

```
$ bundle exec rails g scaffold books title:string memo:text
```

```
$ bundle exec rake db:migrate
```

```
$ bundle exec rails s
```

http://localhost:3000/books

ブラウザで上記のアドレスへアクセスして動作確認してみましょう。

Listing books	
Title	Memo
ドラえもん 猫型ロボットとの冒険浪漫譚	Show Edit Destroy
君に届け 北海道の高校を舞台にした青春群像	Show Edit Destroy
New Book	

←こんなアプリができます。

※第17回でつくった
アプリと同じ内容です。

Scaffold



scaffold

かんたんにブログページのような
投稿機能を生成する機能

作成、表示、更新、削除
の4つの機能を持つページ群を
動かすコードを生成します。

※実はscaffoldを活用するというのは実際の開発では
そう多くないです。しかし、基本の勉強にはうってつけ
なので、scaffoldで生成したページがどう動くかを中心
に解説を進めていきます。

scaffoldが作った
これらの機能は
Webアプリの基本となるため
CRUD（クラッド）
という名前がついてます。
Create, Read, Update, Destroy
の頭文字です。
(作成・表示・更新・削除)

生成されるコードファイル群

```
$ bundle exec rails g scaffold books title:string memo:text
```

```
invoke  active_record
create    db/migrate/20121029112448_create_books.rb
create    app/models/book.rb
invoke  test_unit
create    test/unit/book_test.rb
create    test/fixtures/books.yml
invoke  resource_route
        resources :books
invoke  scaffold_controller
create    app/controllers/books_controller.rb
invoke  erb
create    app/views/books
create    app/views/books/index.html.erb
create    app/views/books/edit.html.erb
create    app/views/books/show.html.erb
create    app/views/books/new.html.erb
create    app/views/books/_form.html.erb
invoke  test_unit
create    test/functional/books_controller_test.rb
invoke  helper
```

scaffold で生成されるファイルの一部

.rb はRubyコード、.html.erb はHTMLテンプレート

- ▶ **db/migrate/20121029112448_create_books.rb**
- ▶ **app/models/book.rb**
- ▶ **resource_route**
 resources :books
- ▶ **app/controllers/books_controller.rb**
- ▶ **app/views/books**
- ▶ **app/views/books/index.html.erb**
- ▶ **app/views/books/edit.html.erb**
- ▶ **app/views/books/show.html.erb**
- ▶ **app/views/books/new.html.erb**
- ▶ **app/views/books/_form.html.erb**

この後の説明に主に関係してくるのは以下のファイル群

- ▶ **db/migrate/20121029112448_create_books.rb**
- ▶ **app/models/book.rb**
- ▶ **resource_route**
 resources :books
- ▶ **app/controllers/books_controller.rb**
- ▶ **app/views/books**
- ▶ **app/views/books/index.html.erb**
- ▶ **app/views/books/edit.html.erb**
- ▶ **app/views/books/show.html.erb**
- ▶ **app/views/books/new.html.erb**
- ▶ **app/views/books/_form.html.erb**

scaffoldは次のページに示す
作成、表示、更新、削除のための
4つの画面と、
画面のない3つの機能を生成します。

index

一覧

Listing books

Title Memo

ドラえもん 猫型ロボットとの冒険浪漫譚 [Show](#) [Edit](#) [Destroy](#)

君に届け 北海道の高校を舞台にした青春群像 [Show](#) [Edit](#) [Destroy](#)

[New Book](#)

new

新規 入力

New book

Title

Memo

[Create Book](#)

[Back](#)

edit

編集

Editing book

Title

Memo

[Update Book](#)

[Show](#) | [Back](#)

show

詳細

Title: ドラえもん

Memo: 猫型ロボットとの冒険浪漫譚

[Edit](#) | [Back](#)

scaffoldで作られる画面と機能一覧(後で説明していきます)

index

/books

GET

削除 新規入力

new

/books/new

GET

redirect

編集

この図
の見方

アクション名
URL(パス)
HTTPメソッド

画面あり

アクション名
URL(パス)
HTTPメソッド

画面なし

edit

/books/:id/edit

GET

destroy

/books/:id
DELETE

新規登録

create

/books
POST

redirect

show

/books/:id
GET

更新

update

/books/:id
PUT

redirect

では、それぞれの画面と
対応するアクションを見ていきましょう。

※アクションとはコントローラに書いてある
publicなメソッドのことです。基本的には、1つ
の画面を表示するとき、1つのアクションが実行さ
れています。
アクションについてはまた後で説明します。

CRUD画面遷移図

index
/books

GET

削除 新規入力

new
/books/new

GET

destroy
/books/:id

DELETE

新規登録

create
/books

POST

この図
の見方

アクション名
URL(パス)
HTTPメソッド

アクション名
URL(パス)
HTTPメソッド

画面あり

画面なし

更新

redirect

edit

/books/:id/edit

GET

update
/books/:id

PUT

show

/books/:id

GET

編集

redirect

redirect



CRUD画面遷移図

index
/books GET

削除 新規入力

new
/books/new GET

destroy
/books/:id DELETE

新規登録

create
/books POST

編集

この図
の見方

アクション名
url
HTTPメソッド

画面あり

更新

アクション名
url
HTTPメソッド

画面なし

redirect

edit
/books/:id/edit GET

update
/books/:id POST

show
/books/:id GET

redirect

redirect

indexアクション 一覧画面

GET /books

※<http://localhost:3000/books>
のことです。

Listing books

Title	Memo	
ドラえもん 猫型ロボットとの冒険浪漫譚		Show Edit Destroy
君に届け 北海道の高校を舞台にした青春群像		Show Edit Destroy

[New Book](#)

CRUD画面遷移図

index

/books

GET

編集

この図
の見方

アクション名
url
HTTPメソッド

画面あり

edit

/books/:id/edit

GET

update

/books/:id

PUT

削除 新規入力

destroy

/books/:id

DELETE

new

/books/new

GET

新規登録

create

/books

POST

アクション名
url
HTTPメソッド

画面なし

redirect

show

/books/:id

GET

redirect

redirect

更新

redirect

PUT



newアクション

GET

/books/new

新規入力画面

New book

Title

Memo

Create Book

Back

CRUD画面遷移図

index

/books

GET

削除 新規入力

new

/books/new

GET

新規登録

create

/books

POST

編集

この図
の見方

アクション名

url
HTTPメソッド

画面あり

更新

アクション名

url
HTTPメソッド

画面なし

redirect

edit

/books/:id/edit

GET

update

/books/:id

PUT

show

/books/:id

GET

editアクション

GET

/books/:id/edit

編集画面

Editing book

Title

Memo

[Update Book](#)

[Show](#) | [Back](#)

CRUD画面遷移図

index

/books

GET

編集

この図
の見方

アクション名
url
HTTPメソッド

画面あり

edit

/books/:id/edit

GET

update

/books/:id

PUT

削除 新規入力

destroy

/books/:id

DELETE

new

/books/new

GET

新規登録

create

/books

POST

アクション名

url
HTTPメソッド

画面なし

redirect

show

/books/:id

GET

redirect

更新

redirect

redirect

showアクション

詳細画面

GET /books/:id

Title: ドラえもん

Memo: 猫型ロボットとの冒険浪漫譚

Edit | Back

CRUD画面遷移図

index

/books

GET

編集

この図
の見方

アクション名
url
HTTPメソッド

画面あり

更新

edit

/books/:id/edit

GET

update

/books/:id

PUT

削除 新規入力

destroy

/books/:id

DELETE

new

/books/new

GET

新規登録

create

/books

POST

アクション名
url
HTTPメソッド

画面なし

redirect

show

/books/:id

GET

redirect

redirect

createアクション 新規登録

POST /books

入力されたデータを新規保存

CRUD画面遷移図

index

/books

GET

削除 新規入力

new

/books/new

GET

destroy

/books/:id

DELETE

新規登録

create

/books

POST

この図
の見方

アクション名

url
HTTPメソッド

画面あり

アクション名

url
HTTPメソッド

画面なし

edit

/books/:id/edit

GET

更新

update
/books/:id
PUT

redirect

show

/books/:id

GET

updateアクション **更新**

PUT /books/:id

入力されたデータで**更新**

CRUD画面遷移図

index

/books

GET

new

/books/new

GET

削除 新規入力

destroy

/books/:id
DELETE

新規登録

create

/books

POST

編集

この図
の見方

アクション名

url
HTTPメソッド

画面あり

更新

アクション名

url
HTTPメソッド

画面なし

redirect

edit

/books/:id/edit

GET

update

/books/:id

PUT

show

/books/:id

GET

redirect

redirect



destroyアクション 削除

DELETE /books/:id

指示されたデータを削除

ここまで説明した部分はこのファイルに書かれています

- ▶ db/migrate/20121029112448_create_books.rb
 - ▶ app/models/book.rb
 - ▶ resource_route
 resources :books
 - ▶ app/controllers/books_controller.rb
 - ▶ app/views/books
 - ▶ app/views/books/index.html.erb
 - ▶ app/views/books/edit.html.erb
 - ▶ app/views/books/show.html.erb
 - ▶ app/views/books/new.html.erb
 - ▶ app/views/books/_form.html.erb
- url+HTTPメソッドと
アクションの対応表
- 7つの
アクション
- 対応する
画面表示

では、処理の流れを追いかけてみ
ましょう。

indexアクション 一覧画面

GET /books

Listing books

Title	Memo	
ドラえもん 猫型ロボットとの冒険浪漫譚		Show Edit Destroy
君に届け 北海道の高校を舞台にした青春群像		Show Edit Destroy

[New Book](#)

登録されているBookのデータを一覧表示する画面です。

index画面が表示されるまでの流れを追いかけてみましょう。

CRUD画面遷移図

index

/books

GET

new

/books/new

GET

削除 新規入力

destroy

/books/:id

DELETE

新規登録

create

/books

POST

編集

この図
の見方

アクション名

url
HTTPメソッド

画面あり

更新

アクション名

url
HTTPメソッド

画面なし

redirect

edit

/books/:id/edit

GET

show

/books/:id

GET

update

/books/:id

PUT

redirect

redirect



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/books`
- ▶ HTTPメソッド : GET

Rails App

Routes

routes.rb

Controller

books_controller.rb indexアクション

View

books/index.html.erb

レスポンス : HTML



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メソッド : GET

Rails App

Routes



routes.rb

Controller



books_controller.rb indexアクション

View



books/index.html.erb

レスポンス : HTML

Routes

Routesはリクエストと処理を行うコントローラの対応表です。
以下のURLでRoutes対応表を見ることができます。
<http://localhost:3000/rails/info/routes>

Helper	HTTP Verb	Path	Controller#Action
<u>Path / Url</u>		<u>Path Match</u>	
books_path	GET	/books(.:format)	books#index
	POST	/books(.:format)	books#create
new_book_path	GET	/books/new(.:format)	books#new
edit_book_path	GET	/books/:id/edit(.:format)	books#edit
book_path	GET	/books/:id(.:format)	books#show
	PATCH	/books/:id(.:format)	books#update
	PUT	/books/:id(.:format)	books#update
	DELETE	/books/:id(.:format)	books#destroy

下線部が今回関係する部分です。

「/books に GET でアクセスされたとき、
BooksController の index アクションへ処理を流す」
という意味になります。

Routes 対応表が書かれたファイル config/routes.rb

```
Appname::Application.routes.draw do
  resources :books
end
```

routesは Appname::Application.routes.draw do から
end の間に書きます。rake routesで表示される8行の対応表は
resources :books の1行だけで記述可能です。
(基本的なroutesであるため、短く書けるようになっています。)

Helper <u>Path / Url</u>	HTTP Verb	Path	Controller#Action
books_path	GET	/books(.:format)	books#index
	POST	/books(.:format)	books#create
new_book_path	GET	/books/new(.:format)	books#new
edit_book_path	GET	/books/:id/edit(.:format)	books#edit
book_path	GET	/books/:id(.:format)	books#show
	PATCH	/books/:id(.:format)	books#update
	PUT	/books/:id(.:format)	books#update
	DELETE	/books/:id(.:format)	books#destroy

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メソッド : GET

Rails App

Routes

routes.rb

Controller

books_controller.rb indexアクション

View

books/index.html.erb

レスポンス : HTML

Controller

app/controllers/books_controller.rb

```
class BooksController < ApplicationController
  def index
    @books = Book.all
  end
  ... #以下 new, edit, createなど7つのアクションが書かれている
end
```

indexアクションは **@books = Book.all** の1行です。
Book.all でBookに関する全データが詰まった配列を取得して、
@booksインスタンス変数へ代入します。

インスタンス変数(@はじまり)へ代入するのは、Viewへ表示のために渡したいからです。
(インスタンス変数ではないと、indexアクションを抜けたところで消えてしまいます。)

次はViewに処理が移ります。

特に指定がない場合、app/views/コントローラ名/アクション名
のViewファイルへ処理が移ります。
(ここでは app/views/books/index.html.erb)

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メソッド : GET

Rails App

Routes

routes.rb

Controller

books_controller.rb indexアクション

View

books/index.html.erb

レスポンス : HTML

View

app/views/books/index.html.erb

...

```
<% @books.each do |book| %> ←@booksはbookがいくつか  
<tr>  
  <td><%= book.title %></td> 初回のbook.title → "ドラえもん"  
  <td><%= book.memo %></td> 初回のbook.memo → "  
  <td><%= link_to 'Show', book %></td> ネコ型ロボットとの冒険浪漫譚"  
  <td><%= link_to 'Edit', edit_book_path(book)  
%></td>  
  <td><%= link_to 'Destroy', book, method: :delete,  
data: { confirm: 'Are you sure?' } %></td>  
</tr>  
<% end %>  
...
```

表示は@books.each do |book| ~ end で行っています。

@books内の全データ("ドラえもん", "君に届け")でブロックを繰り返し実行し、titleやmemoを表示したり、show画面やedit画面、削除ボタンのリンクを生成します。

Listing books

Title

Memo

ドラえもん 猫型ロボットとの冒険浪漫譚

Show Edit Destroy

君に届け 北海道の高校を舞台にした青春群像 Show Edit Destroy

New Book

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/books`
- ▶ HTTPメソッド : GET

Rails App

Routes

routes.rb

Controller

books_controller.rb indexアクション

View

books/index.html.erb

レスポンス : HTML



indexアクション 一覧画面

GET /books

Listing books

Title	Memo	
ドラえもん 猫型ロボットとの冒険浪漫譚		Show Edit Destroy
君に届け 北海道の高校を舞台にした青春群像		Show Edit Destroy

[New Book](#)

以上の処理を経て一覧画面が表示されています。

index

/books

GET

いま説明した箇所

編集

この図
の見方

アクション名
url
HTTPメソッド

画面あり

更新

アクション名
url
HTTPメソッド

画面なし

redirect

edit

/books/:id/edit

GET

新規入力

削除

destroy

s/:id
DELETE

new

/books/new

GET

新規登録

create

/books

POST

redirect

update

/books/:id

PUT

show

/books/:id

GET

index

/books

GET

新規入力

new

/books/new

GET

削除

destroy

/books/:id

DELETE

新規登録

redirect

編集

この図
の見方

アクション名

URL(パス)
HTTPメソッド

画面あり

アクション名

URL(パス)
HTTPメソッド

画面なし

create

/books

POST

更新

redirect

redirect

edit

/books/:id/edit

GET

update

/books/:id

PUT

show

/books/:id

GET

のこりはまた次回以降で

この図
の見方

画面あり

画面なし

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>
- ▶ ドットインストール(web)(動画)
http://dotinstall.com/lessons/basic_rails_v2
- ▶ **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/4797372273/>



講義資料置き場

過去の資料がDLできます。

<https://github.com/igaiga/hitotsubashi-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