

Table of Contents

Introduction	0
はじめに	1
Ruby on Rails入門	2
環境構築	2.1
Ruby on Rails基礎	2.2
アプリケーション開発 1	3
基本的なモデルの実装	3.1
画像アップロード(CarrierWave)	3.2
デザインの適用(Bootstrapの導入)	3.3
アプリケーション開発 2	4
ログイン機能(Devise)	4.1
アプリケーション開発 3	5
コメント機能	5.1

Rails Training

- <https://github.com/sadah/rails-training>
- <https://sadah.github.io/rails-training/>
- <https://sadah.github.io/rails-training/book.pdf>

Install

```
npm instal
```

for pdf / ebook

```
brew install Caskroom/cask/calibre
```

Usage

```
npm start # http://localhost:4000
```

Build

```
npm run build
```

Tests

```
npm test
```

はじめに

Rails Training

- <https://github.com/sadah/rails-training>
- <https://sadah.github.io/rails-training/>
- <https://sadah.github.io/rails-training/book.pdf>

Ruby on Rails入門

- 環境構築
- Ruby on Rails基礎

Ruby on Rails 環境構築

この手順は OS X 10.8, 10.9, 10.10 を対象としています。

Xcode インストール

App Store から Xcode をインストールする。一度 Xcode を起動し、ライセンス使用許諾契約に同意する。

Command line tools インストール

Command line tools をインストールする。

```
xcode-select --install
```

Homebrew インストール

[Homebrew](#) をインストールする。

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/brew/install.sh)"
```

rbenv インストール

[rbenv](#) をインストールする。

```
brew update  
brew install rbenv rbenv-gem-rehash ruby-build
```

bashを利用している場合、以下のコマンドで環境変数を設定する。zshなどの場合、適宜書き換えて実行する。

```
echo 'eval "$(rbenv init -)"' >> ~/.bash_profile  
echo 'export PATH="$HOME/.rbenv/shims:$PATH"' >> ~/.bash_profile  
source ~/.bash_profile
```

Ruby インストール

Rubyをインストールする

```
rbenv install 2.2.3
```

デフォルトのRubyバージョンを指定する。

```
rbenv global 2.2.3
```

Ruby on Rails インストール

Ruby on Rails をインストールする。

```
gem install rails --no-ri --no-rdoc
```

確認

以下のコマンドが正しく実行できれば、Railsのインストールは正しくできています。

```
rails new sample-project
```

その他

Windows環境はサポート対象外ですが、[RailsInstaller](#) が便利だと思います。

- <https://s3.amazonaws.com/railsinstaller/Windows/railsinstaller-3.1.0.exe>

Ruby on Rails 基礎

Railsアプリケーションの作成

Railsアプリケーションを作成します。

```
rails new first-app
```

Railsアプリケーションは以下のよう構成になっています。

```
first-app
├── Gemfile # ライブラリを管理するファイル
├── Gemfile.lock
├── README.rdoc
├── Rakefile
└── app # アプリケーションのコードが配置される
    ├── assets
    │   ├── images
    │   ├── javascripts
    │   │   └── application.js
    │   └── stylesheets
    │       └── application.css
    ├── controllers
    │   ├── application_controller.rb
    │   └── concerns
    ├── helpers
    │   └── application_helper.rb
    ├── mailers
    ├── models
    │   └── concerns
    └── views
        ├── layouts
        │   └── application.html.erb
└── bin
```

```
|   └── bundle
|   └── rails
|   └── rake
|   └── setup
|   └── spring
└── config # アプリケーションの設定ファイル
    ├── application.rb
    ├── boot.rb
    ├── database.yml
    ├── environment.rb
    └── environments
        ├── development.rb
        ├── production.rb
        └── test.rb
    ├── initializers
        ├── assets.rb
        ├── backtrace_silencers.rb
        ├── cookies_serializer.rb
        ├── filter_parameter_logging.rb
        ├── inflections.rb
        ├── mime_types.rb
        ├── session_store.rb
        └── wrap_parameters.rb
    ├── locales
        └── en.yml
    ├── routes.rb
    └── secrets.yml
├── config.ru
└── db
    └── seeds.rb
└── lib
    ├── assets
    └── tasks
└── log
└── public # 静的なファイル
    ├── 404.html
    ├── 422.html
    └── 500.html
```

```
|   └── favicon.ico  
|   └── robots.txt  
└── test  
    ├── controllers  
    ├── fixtures  
    ├── helpers  
    ├── integration  
    ├── mailers  
    ├── models  
    └── test_helper.rb  
└── tmp  
    └── cache  
      └── assets  
└── vendor  
    └── assets  
      ├── javascripts  
      └── stylesheets
```

Railsアプリケーションを起動します。

```
cd first-app  
rails server
```

ここからさきでは、first-appのディレクトリをRAILS_ROOTと呼びます。

scaffoldによるコードの自動生成

Railsではさまざまなコードを自動生成できます。scaffoldでアプリケーションの雛形を作ります。scaffoldは一覧、詳細、追加、削除、変更のコードを自動生成します。

ここでは **bookmark** を管理する機能を作ります。RAILS_ROOTで以下のコマンドを実行します。

```
rails generate scaffold bookmark title:string description:text ..
```

DBを更新して、アプリケーションを起動します。RAILS_ROOTで以下のコマンドを実行します。

```
bin/rake db:migrate  
rails server
```

<http://localhost:3000/bookmarks>で動作確認ができます。

データの追加、削除、変更などを行ってみましょう。

scaffoldで生成されるファイル

scaffoldで生成されたファイルを確認していきます。scaffoldではこのような実行結果が出力されます。

```
% rails generate scaffold bookmark title:string description:text
  invoke  active_record
  create    db/migrate/20151129091144_create_bookmarks.rb
  create    app/models/bookmark.rb
  invoke  test_unit
  create    test/models/bookmark_test.rb
  create    test/fixtures/bookmarks.yml
  invoke  resource_route
    route   resources :bookmarks
  invoke  scaffold_controller
  create    app/controllers/bookmarks_controller.rb
  invoke  erb
  create    app/views/bookmarks
  create    app/views/bookmarks/index.html.erb
  create    app/views/bookmarks/edit.html.erb
  create    app/views/bookmarks/show.html.erb
  create    app/views/bookmarks/new.html.erb
  create    app/views/bookmarks/_form.html.erb
  invoke  test_unit
  create    test/controllers/bookmarks_controller_test.rb
  invoke  helper
  create    app/helpers/bookmarks_helper.rb
  invoke  test_unit
  invoke  jbuilder
  create    app/views/bookmarks/index.json.jbuilder
  create    app/views/bookmarks/show.json.jbuilder
  invoke  assets
  invoke  coffee
  create    app/assets/javascripts/bookmarks.coffee
  invoke  scss
  create    app/assets/stylesheets/bookmarks.scss
  invoke  scss
  create    app/assets/stylesheets/scaffolds.scss
```

scaffoldでは以下のようなファイルを生成しています。

- active_record
 - テーブルの作成
 - モデルの作成
- routeの設定
- controller
 - viewの作成
 - helperの作成
 - jbuilder(jsonテンプレート)の作成
- assets
 - CoffeeScriptの作成
 - Scssの作成

実際にファイルを開いて確認して見ましょう。

migration

db/migrate/20151129091144_create_bookmarks.rb

```
class CreateBookmarks < ActiveRecord::Migration
  def change
    create_table :bookmarks do |t|
      t.string :title
      t.text :description
      t.string :url

      t.timestamps null: false
    end
  end
end
```

routes

config/routes.rb

```
Rails.application.routes.draw do
```

```
resources :bookmarks

# The priority is based upon order of creation: first created
# See how all your routes lay out with "rake routes".

# You can have the root of your site routed with "root"
# root 'welcome#index'

# Example of regular route:
# get 'products/:id' => 'catalog#view'

# Example of named route that can be invoked with purchase_url
# get 'products/:id/purchase' => 'catalog#purchase', as: :pu

# Example resource route (maps HTTP verbs to controller actions)
# resources :products

# Example resource route with options:
# resources :products do
#   member do
#     get 'short'
#     post 'toggle'
#   end
#   #
#   collection do
#     get 'sold'
#   end
# end

# Example resource route with sub-resources:
# resources :products do
#   resources :comments, :sales
#   resource :seller
# end

# Example resource route with more complex sub-resources:
# resources :products do
#   resources :comments
#   resources :sales do
```

```
#       get 'recent', on: :collection
#   end
# end

# Example resource route with concerns:
# concern :toggleable do
#   post 'toggle'
# end
# resources :posts, concerns: :toggleable
# resources :photos, concerns: :toggleable

# Example resource route within a namespace:
# namespace :admin do
#   # Directs /admin/products/* to Admin::ProductsController
#   # (app/controllers/admin/products_controller.rb)
#   resources :products
# end
end
```

rake routes で定義されているroutesを確認できます。

```
% rake routes
      Prefix Verb    URI Pattern          Controller#Action
bookmarks GET    /bookmarks(.:format)    bookmarks#index
           POST   /bookmarks(.:format)    bookmarks#create
new_bookmark GET   /bookmarks/new(.:format) bookmarks#new
edit_bookmark GET   /bookmarks/:id/edit(.:format) bookmarks#edit
bookmark    GET   /bookmarks/:id(.:format)  bookmarks#show
           PATCH  /bookmarks/:id(.:format)  bookmarks#update
           PUT    /bookmarks/:id(.:format)  bookmarks#update
           DELETE /bookmarks/:id(.:format)  bookmarks#destroy
```

models

app/models/bookmark.rb

```
class Bookmark < ActiveRecord::Base
end
```

controllers

app/controllers/bookmarks_controller.rb

```
class BookmarksController < ApplicationController
  before_action :set_bookmark, only: [:show, :edit, :update, :destroy]

  # GET /bookmarks
  # GET /bookmarks.json
  def index
    @bookmarks = Bookmark.all
  end

  # GET /bookmarks/1
  # GET /bookmarks/1.json
  def show
  end

  # GET /bookmarks/new
  def new
    @bookmark = Bookmark.new
  end

  # GET /bookmarks/1/edit
  def edit
  end

  # POST /bookmarks
  # POST /bookmarks.json
  def create
    @bookmark = Bookmark.new(bookmark_params)
```

```
respond_to do |format|
  if @bookmark.save
    format.html { redirect_to @bookmark, notice: 'Bookmark was successfully created.' }
    format.json { render :show, status: :created, location: @bookmark }
  else
    format.html { render :new }
    format.json { render json: @bookmark.errors, status: :unprocessable_entity }
  end
end

# PATCH/PUT /bookmarks/1
# PATCH/PUT /bookmarks/1.json
def update
  respond_to do |format|
    if @bookmark.update(bookmark_params)
      format.html { redirect_to @bookmark, notice: 'Bookmark was successfully updated.' }
      format.json { render :show, status: :ok, location: @bookmark }
    else
      format.html { render :edit }
      format.json { render json: @bookmark.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /bookmarks/1
# DELETE /bookmarks/1.json
def destroy
  @bookmark.destroy
  respond_to do |format|
    format.html { redirect_to bookmarks_url, notice: 'Bookmark was successfully destroyed.' }
    format.json { head :no_content }
  end
end

private
# Use callbacks to share common setup or constraints between
```

```
def set_bookmark
  @bookmark = Bookmark.find(params[:id])
end

# Never trust parameters from the scary internet, only allow
# parameters we know and expect!
def bookmark_params
  params.require(:bookmark).permit(:title, :description, :url)
end
```

helpers

app/helpers/bookmarks_helper.rb

```
module BookmarksHelper
end
```

views

app/views/bookmarks/index.html.erb

```
<p id="notice"><%= notice %></p>

<h1>Listing Bookmarks</h1>

<table>
  <thead>
    <tr>
      <th>Title</th>
      <th>Description</th>
      <th>Url</th>
      <th colspan="3"></th>
    </tr>
  </thead>

  <tbody>
    <% @bookmarks.each do |bookmark| %>
      <tr>
        <td><%= bookmark.title %></td>
        <td><%= bookmark.description %></td>
        <td><%= bookmark.url %></td>
        <td><%= link_to 'Show', bookmark %></td>
        <td><%= link_to 'Edit', edit_bookmark_path(bookmark) %></td>
        <td><%= link_to 'Destroy', bookmark, method: :delete, data: { confirm: 'Are you sure?' } %></td>
      </tr>
    <% end %>
  </tbody>
</table>

<br>

<%= link_to 'New Bookmark', new_bookmark_path %>
```

app/views/bookmarks/_form.html.erb

```
<%= form_for(@bookmark) do |f| %>
<% if @bookmark.errors.any? %>
<div id="error_explanation">
<h2><%= pluralize(@bookmark.errors.count, "error") %> proh
<ul>
<% @bookmark.errors.full_messages.each do |message| %>
<li><%= message %></li>
<% end %>
</ul>
</div>
<% end %>

<div class="field">
<%= f.label :title %><br>
<%= f.text_field :title %>
</div>
<div class="field">
<%= f.label :description %><br>
<%= f.text_area :description %>
</div>
<div class="field">
<%= f.label :url %><br>
<%= f.text_field :url %>
</div>
<div class="actions">
<%= f.submit %>
</div>
<% end %>
```

app/views/bookmarks/index.json.jbuilder

```
json.array!(@bookmarks) do |bookmark|
  json.extract! bookmark, :id, :title, :description, :url
  json.url bookmark_url(bookmark, format: :json)
end
```

console

`rails console` で対話的に操作できます。RAILS_ROOTで以下のコマンドを実行します。

```
% rails console
Loading development environment (Rails 4.2.4)
irb(main):001:0>
```

`Bookmark.last` で最後に登録したBookmarkを取得します。

```
irb(main):001:0> Bookmark.last
  Bookmark Load (0.2ms)  SELECT "bookmarks".* FROM "bookmarks"
=> #<Bookmark id: 1, title: "sadah", description: "My portfolio"
irb(main):002:0>
```

このようにデータを作成することもできます。

```
irb(main):002:0> bookmark = Bookmark.create(title: "test", descr
(4.0ms)  begin transaction
SQL (5.1ms)  INSERT INTO "bookmarks" ("title", "description",
(1.0ms)  commit transaction
=> #<Bookmark id: 2, title: "test", description: "test bookmark"
irb(main):003:0>
```

Active Recordのクエリについてはこちらがわかりやすいです。

- [Active Record クエリインターフェイス | Rails ガイド](#)

debug

pryはirb(標準のRubyコンソール)に代わる、パワフルなコンソールです。

- [pry/pry](#)

Gemfileにpryのgemを追加します。

```
group :development, :test do
  # Call 'byebug' anywhere in the code to stop execution and get
  gem 'byebug'
  gem 'pry'
  gem 'pry-doc'
  gem 'pry-byebug'
  gem 'pry-rails'
  gem 'awesome_print'
end
```

それぞれのgemはこのような機能を持ちます。

gem	
pry	pry本体
pry-doc	ドキュメントやメソッドを表示する
pry-byebug	デバッグができるようにする
pry-rails	Railsのコンソールをpryにする
awesome_print	データ構造などをわかりやすく表示する

Railsコンソールを立ち上げると、pryが起動していることがわかります。
RAILS_ROOTで以下のコマンドを実行します。

```
% rails c
Loading development environment (Rails 4.2.4)
[1] pry(main)>
```

show-docでドキュメントが表示されます。

```
[1] pry(main)> show-doc String.nil?

From: object.c (C Method):
Owner: Kernel
Visibility: public
Signature: nil?()
Number of lines: 4

Only the object nil responds true to nil?.

Object.new.nil?    #=> false
nil.nil?          #=> true
```

show-methodでメソッドが表示されます。

```
[2] pry(main)> show-method String.nil?

From: object.c (C Method):
Owner: Kernel
Visibility: public
Number of lines: 5

static VALUE
rb_false(VALUE obj)
{
    return Qfalse;
}
```

pryでデバッグしていきます。gemを追加したのでrailsを再起動します。

app/controllers/bookmarks_controller.rb に binding.pry を追加します。

```
def index
  @bookmarks = Bookmark.all
  binding.pry
end
```

<http://localhost:3000/bookmarks> を表示します。railsを起動したターミナルでデバッグができます。

```
% rails s
=> Booting WEBrick
=> Rails 4.2.4 application starting in development on http://loc
=> Run `rails server -h` for more startup options
=> Ctrl-C to shutdown server
[2015-12-04 07:43:32] INFO  WEBrick 1.3.1
[2015-12-04 07:43:32] INFO  ruby 2.2.3 (2015-08-18) [x86_64-darw
[2015-12-04 07:43:32] INFO  WEBrick::HTTPServer#start: pid=35325
```

```
Started GET "/bookmarks" for ::1 at 2015-12-04 07:43:42 +0900
  ActiveRecord::SchemaMigration Load (0.5ms)  SELECT "schema_mi
Processing by BookmarksController#index as HTML
```

```
From: /Users/sada/git/gs/first-app/app/controllers/bookmarks_cor
6: def index
7:   @bookmarks = Bookmark.all
8:   binding.pry
=> 9: end
```

インスタンス変数などを表示できます。

```
[1] pry(#<BookmarksController>) > @bookmarks
      Bookmark Load (1.2ms)  SELECT "bookmarks".* FROM "bookmarks"
[
[0] #<Bookmark:0x007feb883cbfb0> {
      :id => 1,
      :title => "sadah",
      :description => "My portfolio site.",
      :url => "https://sadah.github.io",
      :created_at => Wed, 02 Dec 2015 00:26:31 UTC +00:00,
      :updated_at => Wed, 02 Dec 2015 00:26:31 UTC +00:00
},
[1] #<Bookmark:0x007feb883cbc90> {
      :id => 2,
      :title => "test",
      :description => "test bookmark",
      :url => "http://exapmle.com",
      :created_at => Thu, 03 Dec 2015 00:04:59 UTC +00:00,
      :updated_at => Thu, 03 Dec 2015 00:04:59 UTC +00:00
}
]
```

helpと入力するとpryの使い方が表示されます。

~/.pryrc にこんな感じの設定を書いておくと便利です。

```
# http://qiita.com/Linda_pp/items/d75d7c3953faa34a1f0e
begin
  require "awesome_print"
  AwesomePrint.pry!
rescue LoadError => err
  puts "no awesome_print :("
end

# http://morizyun.github.io/blog/pry-tips-rails-ruby/
# pry-debuggerのショートカット
Pry.commands.alias_command 'c', 'continue'
Pry.commands.alias_command 's', 'step'
Pry.commands.alias_command 'n', 'next'
```

またエラーが発生した場合、エラーが表示された画面でデバッグを行うこともできます。

課題

今週の課題です。

- `link_to`を使ってURLをリンクにしてみましょう。
- タイトルとURLを必須項目にしましょう。

さらに頑張りたい方はbookmarkにコメントできるようにしましょう。
ざっくりとした流れはこんな感じです。

- `comment(user_name, content, bookmark_id)`をscaffoldで作成する
- `has_many, belongs_to`で関連をつける
- `bookmark#show`でコメントを表示する
- `bookmark#show`でコメントを新規登録できるようする

アプリケーション開発 1

今回からは、写真を投稿できるアプリケーションを実装していきます。

- アプリケーション開発 1
 - 基本的なモデルの実装)
 - 画像アップロード(CarrierWave)
 - デザインの適用(Bootstrapの導入)

アプリケーション開発 1

Railsアプリケーションの作成

Railsアプリケーションを作成します。ターミナルで以下のコマンドを実行します。

```
rails new photolog  
cd photolog
```

scaffoldでphotoに関するファイルを作成します。

image、captionというカラムをもたせます。imageには画像ファイルのURLを入れます。ターミナルで以下のコマンドを実行します。

```
rails g scaffold photo image:string caption:text  
rake db:migrate
```

サーバーを起動して、正しく動作するか確認してみましょう。ターミナルで以下のコマンドを実行します。

```
rails s
```

課題

- 前回導入したPryを、このRailsアプリケーションにも導入しましょう
- image と caption を必須項目にしましょう

画像アップロード(CarrierWave)

CarrierWaveというgemを使うと、簡単に画像のアップロードが実装できます。

- [carrierwaveuploader/carrierwave](#)

エディタでGemfileを編集します。この行の後に

```
# bundle exec rake doc:rails generates the API under doc/api.  
gem 'sdoc', '~> 0.4.0', group: :doc
```

この1行を追加します。

```
gem 'carrierwave'
```

`bundle install` を実行してgemをインストールします。ターミナルで以下のコマンドを実行します。

```
bundle install
```

画像をアップロードするためのコードを生成します。ターミナルで以下のコマンドを実行します。

```
rails generate uploader Image
```

エディタでapp/models/photo.rbを編集します。この行の後に

```
class Photo < ActiveRecord::Base
```

この行を追加します。

```
mount_uploader :image, ImageUploader
```

画像ファイルをアップロードできるようにします。エディタで
app/views/photos/_form.html.erbを編集します。

この行を

```
<%= f.text_field :image %>
```

このように変更します。

```
<%= f.file_field :image %>
```

画像を表示できるようにします。エディタで
app/views/photos/show.html.erbを編集します。

この行を

```
<%= @photo.image %>
```

このように変更します。

```
<%= image_tag(@photo.image_url) if @photo.image.present? %>
```

サーバーを再起動します。サーバーすでに起動している場合は Ctrl + c で
停止してください。

サーバーを起動してください。ターミナルで以下のコマンドを実行しま
す。

```
rails s
```

サーバーが起動したら、写真のアップロードを試してみてください。

デザインの適用(Bootstrapの導入)

Bootstrapを導入して、デザインを適用していきます。

- Bootstrap · The world's most popular mobile-first and responsive front-end framework.

レイアウトファイルの編集

エディタで app/views/layouts/application.html.erb を編集します。この行の後に

```
<title>Photolog</title>
```

この2行を追加します。

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css">
```

この行の前後を

```
<%= yield %>
```

このように変更します。

```
<div class="container">
  <%= yield %>
</div>
```

この行の前に

```
</body>
```

この行を追加します

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/
```

show.html.erb の編集

エディタで `app/views/photos/show.html.erb` を編集します。

この行以降の内容を

```
<p id="notice"><%= notice %></p>
```

この内容に変更します。

```
<div class="row">
  <div class="col-lg-12">
    <%= image_tag(@photo.image_url) if @photo.image.present? %>
    <div>
      <h4><%= @photo.caption %></h4>
      <p>
        <%= link_to 'Edit', edit_photo_path(@photo) %>
        <%= link_to 'Destroy', @photo, method: :delete, data: {
          </p>
        </div>
      </div>
    </div>
</div>
```

課題

- `app/views/photos/index.html.erb` でも画像ファイルを表示しましょう
- `app/views/photos/index.html.erb` でも表示を整えてみましょう。
- 画像のサイズを指定しましょう

さらに頑張りたい人は

- Media Queriesを使って、レスポンシブウェブデザインを適用しましょう
- 画面が一定サイズより小さい場合は、画像が横幅いっぱいに表示されるようにしましょう

画面が大きい場合はこのように表示され

photolog



Goodpatch 4th Anniversary!!

画面が小さい場合はこのように表示されるようにしてみましょう。

photolog



Goodpatch 4th Anniversary!!