

WEB+DBシステム (入門編)



第13回 My Twitterサイトの制作

今日のテーマ

- 一気に、My Twitterサイトを作ってみる。
 - プロジェクトの生成からの作業を総復習する。
- その際に、ログイン認証を行い、ユーザの登録を組み合わせる。
 - ログイン認証には(今年は)deviseを用いる。

プロジェクト名を決める

名前なので、何でも構いませんが、各自が自由に決めて下さい。

私のサンプルの場合：

最初、twitterの超軽量版なので、twilite(トワイライト)にしようかと思いましたが、同音の単語twilight(トワイライト)に「たそがれ、(人生の)衰退期」という意味があるので、やめて、twit(小鳥のさえずり)から、同意の別単語(chirp)を選んで、chirpyとつけてみました。

という訳で、chirpyで作ってみます。

生涯現役プログラマを目指しつつ、本気でこの呼び名を嫌うあたりに、自分の年齢を感じてしまいました。

プロジェクト生成

chirpyプロジェクトを生成する。

```
rails new chirpy
```

とコマンドを入力する。

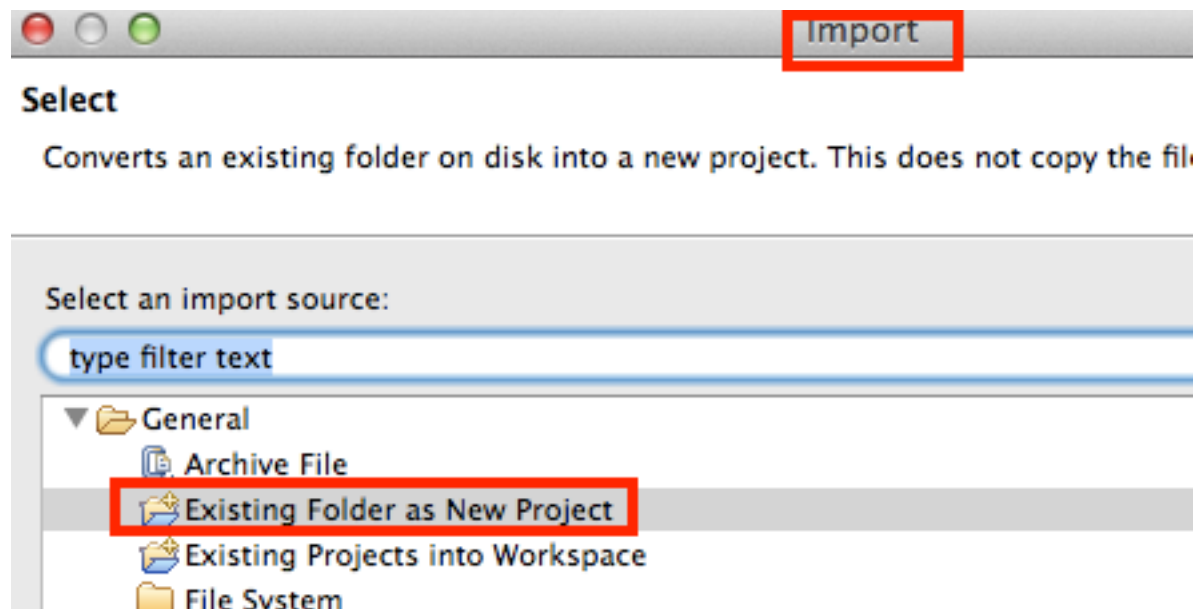
とにかく、数をこなして慣れるという意味では一つのプロジェクトにこだわらず、何度でもいくつでも、プロジェクト生成を試してみてください。

```
kobayashi-ikuo-no-MacBook:Aptana3Work kobayashi$ rails new chirpy
create
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
```

Aptanaへの読み込み

生成したプロジェクトを、Aptana3へ読み込みます。

ファイルメニューのインポートで、「存在しているフォルダを新しいプロジェクトとする」項目を選びます。



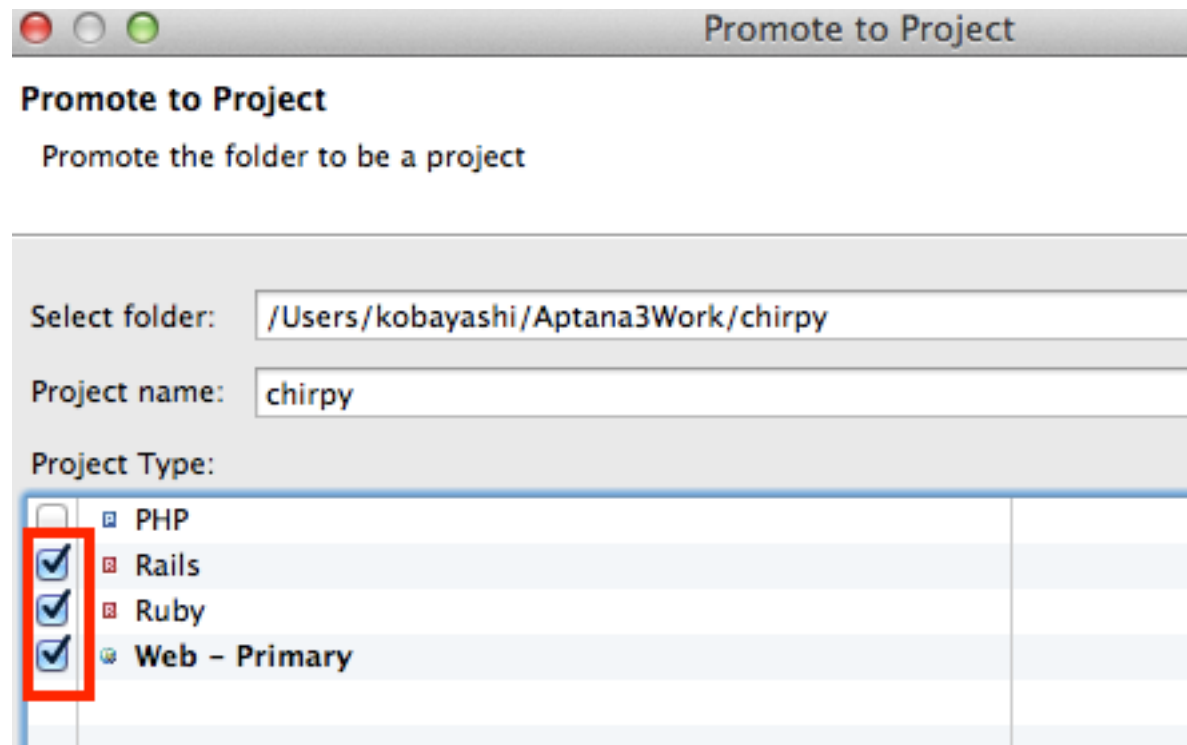
フォルダ指定と読み込み

生成したプロジェクトのフォルダを指定して、そのフォルダを「新プロジェクト」として読み込みます。

Rails

Ruby

の両方に
チェック



ログイン認証用devise

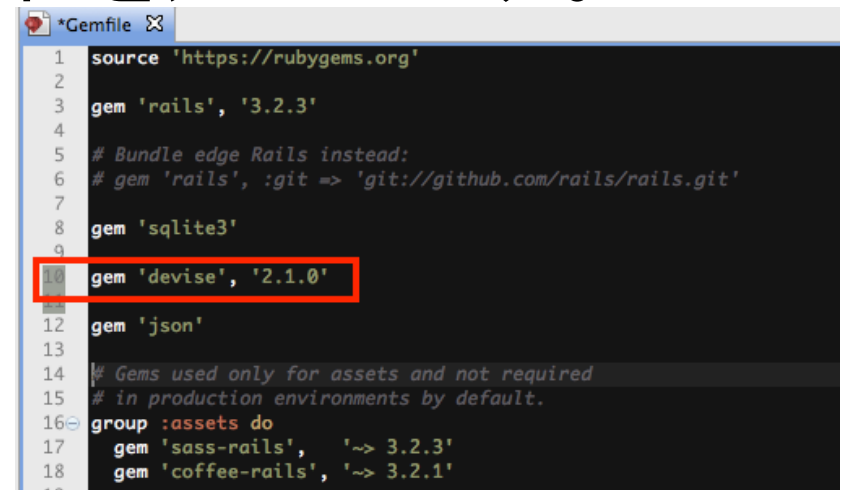
今回利用するのは、ログイン認証の機能です。
Gemfileを開いて、以下の1行を追加します。

`gem 'devise', '2.1.0'`

この1行を追加したら、

`bundle install`

コマンドを実行します。



```
*Gemfile ✕
1 source 'https://rubygems.org'
2
3 gem 'rails', '3.2.3'
4
5 # Bundle edge Rails instead:
6 # gem 'rails', :git => 'git://github.com/rails/rails.git'
7
8 gem 'sqlite3'
9
10 gem 'devise', '2.1.0'
11
12 gem 'json'
13
14 # Gems used only for assets and not required
15 # in production environments by default.
16 group :assets do
17   gem 'sass-rails', '~> 3.2.3'
18   gem 'coffee-rails', '~> 3.2.1'
19 end
```

```
kobayashi-ikuo-no-MacBook:Aptana3Work kobayashi$ cd chirpy
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ ls
Gemfile      app          doc          script
Gemfile.lock config      lib          test
README.rdoc  config.ru   log          tmp
Rakefile     db          public       vendor
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ bundle install
Using rake (0.9.2.2)
Using i18n (0.6.0)
Using multi_json (1.3.6)
Using activesupport (3.2.3)
```

Devise Gemのインストール確認

`bundle install`

の後で、

`gem list devise`

と入力します。

Devise (2.1.0)

を確認します。

```
13 installed.  
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ gem list devise  
  
*** LOCAL GEMS ***  
  
devise (2.1.0)  
kobayashi-ikuo-no-MacBook:chirpy kobayashi$
```


ProjectへのDeviseインストール

次に、プロジェクトにdeviseをインストールします。

`rails generate devise:install`

と入力します。

```
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ rails generate devise:install
create  config/initializers/devise.rb
create  config/locales/devise.en.yml
```

Some setup you must do manually if you haven't yet:

1. Ensure you have defined default url options in your environments files. Here is an example of default_url_options appropriate for a development environment in config/environments/development.rb:

```
config.action_mailer.default_url_options = { :host => 'localhost:3000' }
```

英語のメッセージ確認

```
create config/initializers/devise.rb
create config/locales/devise.en.yml
```

Some setup you must do manually if you haven't yet:

1. Ensure you have defined default url options in your environments files. Here is an example of default_url_options appropriate for a development environment in config/environments/development.rb:

```
config.action_mailer.default_url_options = { :host => 'localhost:3000' }
```

In production, :host should be set to the actual host of your application.

2. Ensure you have defined root_url to *something* in your config/routes.rb. For example:

```
root :to => "home#index"
```

3. Ensure you have flash messages in app/views/layouts/application.html.erb. For example:

```
<p class="notice"><%= notice %></p>
<p class="alert"><%= alert %></p>
```

4. If you are deploying Rails 3.1 on Heroku, you may want to set:

```
config.assets.initialize_on_precompile = false
```

On config/application.rb forcing your application to not access the DB or load models when precompiling your assets.



必要な作業に注目して下さい。

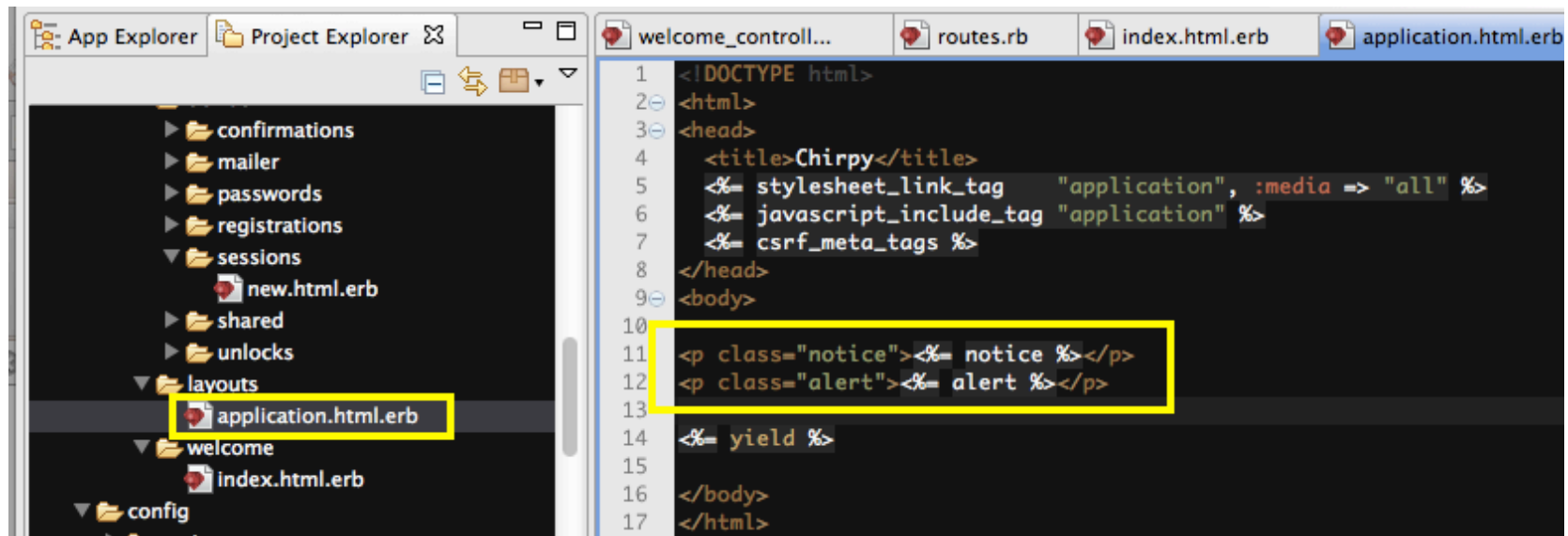
views/layouts/application.html.erb

Deviseのメッセージにあった2行

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

```
<p class="alert"><%= alert %></p>
```

を書き加えます。



The screenshot shows a code editor with two panes. The left pane, labeled 'App Explorer' and 'Project Explorer', displays a file tree with folders like 'confirmations', 'mailer', 'passwords', 'registrations', 'sessions', 'shared', 'unlocks', 'layouts', and 'welcome'. The file 'application.html.erb' under the 'layouts' folder is highlighted with a yellow box. The right pane shows the code for 'application.html.erb', with lines 11 and 12 highlighted by a yellow box:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Chirpy</title>
5   <%= stylesheet_link_tag "application", :media => "all" %>
6   <%= javascript_include_tag "application" %>
7   <%= csrf_meta_tags %>
8 </head>
9 <body>
10
11 <p class="notice"><%= notice %></p>
12 <p class="alert"><%= alert %></p>
13
14 <%= yield %>
15
16 </body>
17 </html>
```

Welcome Screen

今回は「白紙」の状態から一気に作る訳ですが、趣向を変えて「Welcome」画面を用意し、そこから「ログイン」に誘導して、ユーザがログインしたら、個別の処理に入るように作ってみます。

Welcome画面の生成

以下のコマンドを入力します。

```
rails generate controller welcome index
```

```
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ rails generate controller welcome index
  create  app/controllers/welcome_controller.rb
  route   get "welcome/index"
  invoke  erb
  create  app/views/welcome
  create  app/views/welcome/index.html.erb
  invoke  test_unit
  create  test/functional/welcome_controller_test.rb
  invoke  helper
  create  app/helpers/welcome_helper.rb
  invoke  test_unit
  create  test/unit/helpers/welcome_helper_test.rb
  invoke  assets
  invoke  coffee
  create  app/assets/javascripts/welcome.js.coffee
  invoke  scss
  create  app/assets/stylesheets/welcome.css.scss
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ █
```

ログイン画面の登録

作成したWelcomeのindex画面を、ログイン画面として登録します。

`config/routes.rb`を編集します。

53行目付近のコメントを外します。

```
root :to => 'welcome#index'
```

```
51  
52 # You can have the root of your site routed with "root"  
53 # just remember to delete public/index.html.  
54 root :to => 'welcome#index'  
55  
56 # See how all your routes lay out with "rake routes"
```

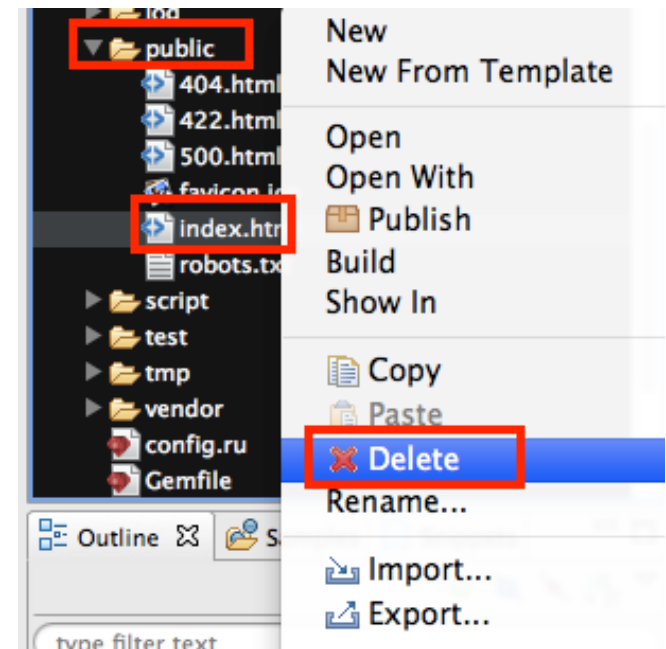
コメントを読む。

root :to => の上に書かれていたコメントを確認して下さい。

You can have the root of your site routed with "root"

just remember to delete public/index.html.

これで、rootの名前でログインルートが使えますが、public/index.htmlの削除を忘れないで下さい、と書かれているので、その指示に従います。



テストラン1回目

ここまでで、とにかく走らせてみます。

rails server

views/welcomeの下の

Index.html.erbが

表示されています。

このアレンジは、

各自にお任せします。



Welcome#index

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

ログイン画面の生成

ログイン画面を生成します。以下のコマンドを入力します。

`rails generate devise:views`

```
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ rails generate devise:views
  invoke  Devise::Generators::SharedViewsGenerator
  create  app/views/devise/shared
  create  app/views/devise/shared/_links.erb
  invoke  form_for
  create  app/views/devise/confirmations
  create  app/views/devise/confirmations/new.html.erb
  create  app/views/devise/passwords
  create  app/views/devise/passwords/edit.html.erb
  create  app/views/devise/passwords/new.html.erb
  create  app/views/devise/registrations
  create  app/views/devise/registrations/edit.html.erb
  create  app/views/devise/registrations/new.html.erb
  create  app/views/devise/sessions
  create  app/views/devise/sessions/new.html.erb
  create  app/views/devise/unlocks
  create  app/views/devise/unlocks/new.html.erb
  invoke  erb
  create  app/views/devise/mailer
  create  app/views/devise/mailer/confirmation_instructions.html.erb
  create  app/views/devise/mailer/reset_password_instructions.html.erb
  create  app/views/devise/mailer/unlock_instructions.html.erb
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ █
```

ログインユーザモデルの生成

ログインするユーザとして、ユーザクラスを生成します。

`rails generate devise user`

```
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ rails generate devise user
  invoke  active_record
  create  db/migrate/20120711160225_devise_create_users.rb
  create  app/models/user.rb
  invoke  test_unit
  create  test/unit/user_test.rb
  create  test/fixtures/users.yml
  insert  app/models/user.rb
  route  devise_for :users
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ █
```

データベース生成

ここまで、すべて「お任せ」でユーザを作成しました。ユーザモデルのデータベースを作成します。

`rake db:migrate`

```
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ rake db:migrate
== DeviseCreateUsers: migrating =====
-- create_table(:users)
   -> 0.0287s
-- add_index(:users, :email, {:unique=>true})
   -> 0.0009s
-- add_index(:users, :reset_password_token, {:unique=>true})
   -> 0.0010s
== DeviseCreateUsers: migrated (0.0318s) =====

kobayashi-ikuo-no-MacBook:chirpy kobayashi$ █
```

ログイン画面の生成

Welcome画面のデザインは、各自で工夫して下さい。ここで何かパフォーマンスを見せて、ワンクッション置いてから、「ログイン」画面に移行してもいいかも知れません。

そこで、このWelcome画面に、ログイン画面へのリンクをはめ込みます。

Welcome/index.html.erb

app/views/welcome/index.html.erbを修正します。

```
<h1>Chirpy: 私のツイッターにようこそ</h1>
```

```
<p><%= link_to 'ログイン', [ :new, :user_session ] %></p>
```

```
<p><%= link_to 'ユーザの登録', [ :new, :user_registration ] %></p>
```

```
<p><%= link_to 'パスワードの再発行', [ :new, :user_password ] %></p>
```



Chirpy: 私のツイッターにようこそ

[ログイン](#)

[ユーザの登録](#)

[パスワードの再発行](#)

次の段階として この画面を設計するには・・・

The image shows a screenshot of Barack Obama's Twitter profile page. The profile header includes a profile picture of Obama, his name "Barack Obama" with a verified badge, and his handle "@BarackObama". Below this, there is a bio: "This account is run by #Obama2012 campaign staff. Tweets from the President are signed -bo. Washington, DC · <http://www.barackobama.com>". To the right of the bio, there are statistics: "4,748 ツイート", "675,844 フォロー", and "17,396,499 フォロワー". A "フォローする" button is visible.

Overlaid on the left side of the page is a blue sidebar with the Obama-Biden campaign logo and the text "OBAMA BIDEN". Below the logo, it says "FOLLOW THE CAMPAIGN". A list of accounts to follow is shown, including @BarackObama, @MichelleObama, @JoeBiden, @Obama2012, @TruthTeam2, and @LatinosForObama. A "ツイート" button is at the bottom of this list.

In the center, a dark blue banner reads "2 3 DAYS UNTIL RECESS: It's time for Congress to act".

Below the banner, there is a "Barack Obamaさんをフォロー" section with a form containing fields for "名前", "メールアドレス", and "パスワード". Below the form, it says "アカウントをお持ちですか? ログインする。" and "新規登録".

On the right, the "ツイート" section shows two tweets from Barack Obama. The first tweet is from 33 minutes ago: "A warm welcome from a fellow ice cream fan in Cedar Rapids yesterday: pic.twitter.com/OKYbwu6v". The second tweet is from 1 hour ago: "Yesterday in Iowa, President Obama spoke about why Congress needs to extend tax cuts for middle-class families: OFA.BO/23EVUh".

「つぶやき」構造の設計

ツイッターの本体画面を作成します。

どのユーザも、ひたすら自分の「つぶやき」を書き込むだけ…。

今回の設計として、フォローしたら、フォローした相手の「つぶやき」も自分の一連の「つぶやき」に表示する、という構造にしてみます。

誰が自分をフォローしているかは、フォローされた相手も「一覧」で見られるようにします。

ただこれだけのchirpシステムとしてみます。

コミュニケーション手段として

可能ならば、「趣味を共有する仲間」を登録させて、どんなメカニズムのコミュニケーション手段を実現させるかは、各自で「仕掛け」を考えてみてください。

ただ、入門編なので、単に、メッセージと画像を登録できるようにするだけでも全く問題はありません。それだけのシステムで構いません。

「課題制作物」は成績評価の対象となるので、自分でイメージしきれぬ範囲で考えて下さい。

テーブル設計

システムでは、すべての「つぶやき」をフラットに取り込むことにします。

「つぶやき」chirpテーブルは、以下のようになります。

Chirp:

content: 140文字のつぶやき本体

user_id: 「誰が」つぶやいたか

これだけとします。

コミュニケーション

フォロー情報の登録を考えてみます。

後期に、一連の技術的な発展系も考えてみて、
本格的な「つぶやき」コミュニケーション構造を作ってみたい
と思います。

各ユーザに「顔写真」を登録する。

ユーザが、自由に自分のホームをデザインできる。

フォローしている「つぶやき」とリンクを取る。

なるべく、「システムの設計能力を高める」ための題材として、各
学生自身のオリジナリティのあるシステム設計をどう学ぶ
か、シラバスを重視しつつ、アレンジを試みてみます。(予告
編になってしまいました。)

ただ、実現できるかは・・・

Chirpテーブル

以下のように、テーブルを生成します。

```
rails generate scaffold chirp content:string  
user_id:integer
```

```
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ rails generate scaffold chirp content:string user_id:integer  
invoke active_record  
create db/migrate/20120711170110_create_chirps.rb  
create app/models/chirp.rb  
invoke test_unit  
create test/unit/chirp_test.rb  
create test/fixtures/chirps.yml  
route resources :chirps  
invoke scaffold_controller  
create app/controllers/chirps_controller.rb  
invoke erb
```

誰がchirpしたか

リレーション構造を登録します。

全てのchirpは、特定のuserに属します。

Usersには

`has_many :chirps`

Chirpには

`belongs_to :user`

を設定します。

```
1 class Chirp < ActiveRecord::Base
2   attr_accessible :content, :user_id
3   belongs_to :user
4 end
5
```

```
1 class User < ActiveRecord::Base
2   # Include default devise modules. Others available are:
3   # :token_authenticatable, :confirmable,
4   # :lockable, :timeoutable and :omniauthable
5   devise :database_authenticatable, :registerable,
6         :recoverable, :rememberable, :trackable, :validatable
7   has_many :chirps
8
9   # Setup accessible (or protected) attributes for your model
10  attr_accessible :email, :password, :password_confirmation, :remember_me
11  # attr_accessible :title, :body
12 end
13
```

テーブルの生成

Chirpsテーブルを作ります。

```
rake db:migrate
```

```
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ rake db:migrate
== CreateChirps: migrating =====
-- create_table(:chirps)
   -> 0.0077s
== CreateChirps: migrated (0.0078s) =====

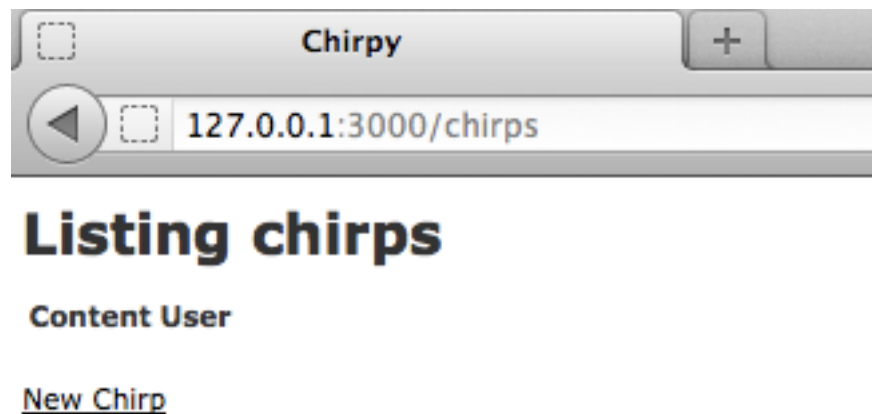
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ █
```

Chirps画面の確認

ここで、一旦動作を確認します。

<http://127.0.0.1:3000/chirps>

で走らせてみます。



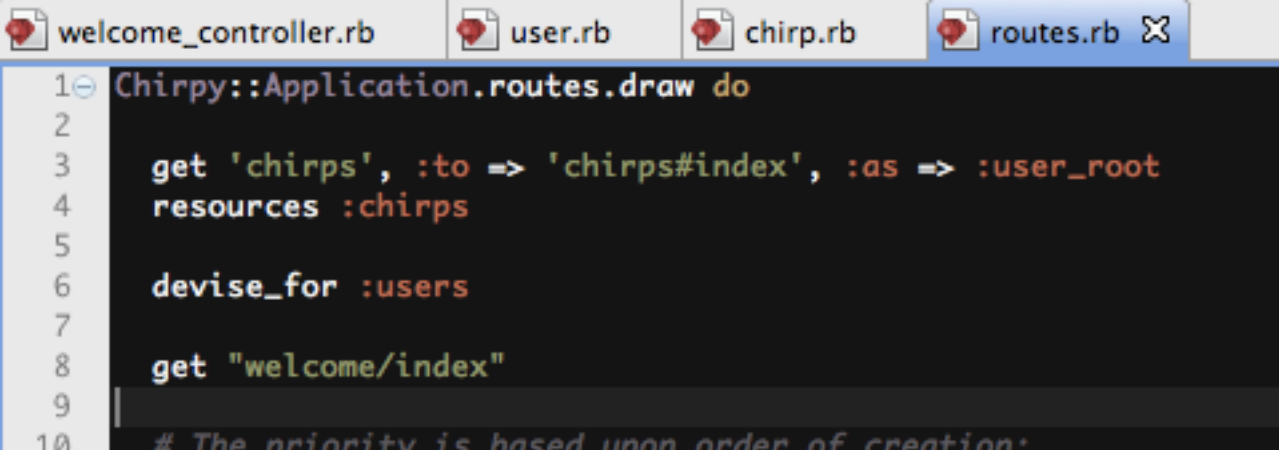
ホーム画面の設定

生成されたchirpのindex画面を、このchirpシステムのホームとして設定・登録してみます。

config/routes.rbに、以下の記述を追加します。

```
get 'chirps', :to => 'chirps#index', :as => :user_root
```

resources :chirpsの前に書きます。



```
welcome_controller.rb  user.rb  chirp.rb  routes.rb ✕
1 Chirpy::Application.routes.draw do
2
3   get 'chirps', :to => 'chirps#index', :as => :user_root
4   resources :chirps
5
6   devise_for :users
7
8   get "welcome/index"
9
10 # The priority is based upon order of creation:
```

config/routes.rb

devise_for :users

- 自動的に生成されているはずですが、なかった場合は自分で書き加えて下さい。これによってdevise規定の「ログイン」画面や、ユーザ登録画面が自動的に組み込まれます。

get 'chirps', :to => 'chirps#index', :as => :user_root

- この行の記述が、ログイン後の画面(ユーザールート)になります。

welcome_controller.rb

ログイン済みの場合には、welcome#indexではなく、:user_root (chirps#index)が表示されるように、切り換えます。

[App/controllers/welcome_controller.rb](#)

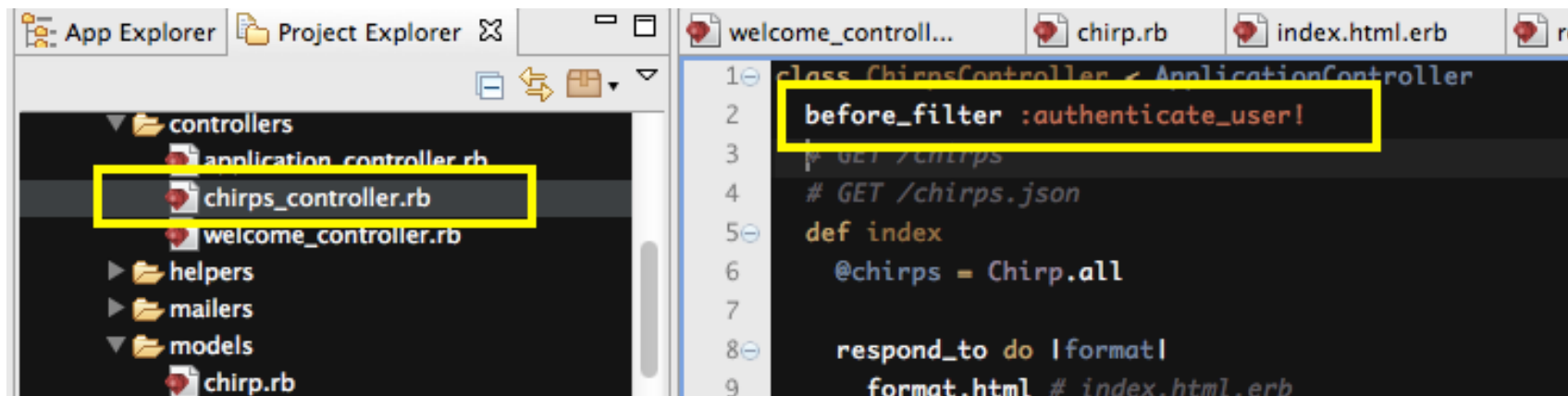
に以下の内容を追加します。

```
if current_user
  redirect_to :user_root
  return
end
```

認証の要求

Controllersのそれぞれのメソッドに、認証の要求を書き加えます。

`before_filter :authenticate_user!`



The screenshot shows an IDE interface with two panes. The left pane, labeled 'App Explorer' and 'Project Explorer', displays a file tree with a folder named 'controllers'. Inside this folder, three files are listed: 'application_controller.rb', 'chirps_controller.rb', and 'welcome_controller.rb'. The 'chirps_controller.rb' file is highlighted with a yellow box. The right pane shows the code for 'chirps_controller.rb'. The code is as follows:

```
1 class ChirpsController < ApplicationController
2   before_filter :authenticate_user!
3   # GET /chirps
4   # GET /chirps.json
5   def index
6     @chirps = Chirp.all
7
8     respond_to do |format|
9       format.html # index.html.erb
```

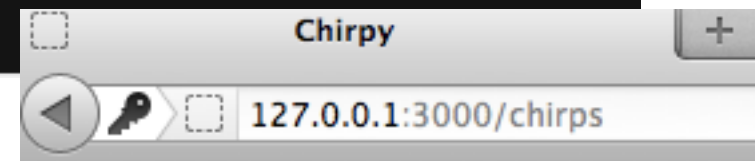
ログアウト処理

メニューバーなどにログアウト処理を起きたいところですが、まずは[views/chirps/index.html.erb](#)に以下の1行を追加します。

```
<%= link_to('Sign out', destroy_user_session_path, :method => :delete) %>
```

```
<br />
```

```
23 <br />
24
25 <%= link_to('Sign out', destroy_user_session_path, :method => :delete) %><br />|
26
27 <%= link_to 'New Chirp', new_chirp_path %>
28
```



Listing chirps

Content User

[Sign out](#)
[New Chirp](#)

一連の動作確認

ここまでの処理で、

Welcome画面（ログイン前）

ログイン画面

ログイン後のルート画面

のそれぞれが動作している事を確認して下さい。

今日欠席した人

ログイン画面(sign inなどの要求の画面)と、ログイン後のroot画面(sign outリンクのある画面)のスクリーンショットを報告して下さい。

できれば、作業内容なども簡単に説明して下さい。その報告で、「出席」に切り換えます。

次回予告

- 最終回は、ひたすらつぶやくだけの「MyTwitter」を作り、「友人関係」の設定のレーションを組み込んで、全体を総括します。
- 今週から「最終課題」に取り込んでみて下さい。今回の「My Twitter」は、あくまでも一例です。