

WEB+DBシステム(応用編)



第14回(2017年1月12日)

人気投票サイトの制作(3/3)

WEB+DBシステム制作のまとめ

ランキング画面(これが見たい)

野菜人気Best10!

第1位



トマト 234票

第2位



カボチャ 123票

今日はこの画面まで作る

前回、未完成の部分

第13回の教材を使用します。

今日の目標

投票結果を集計する。

ランキング画面に表示する。

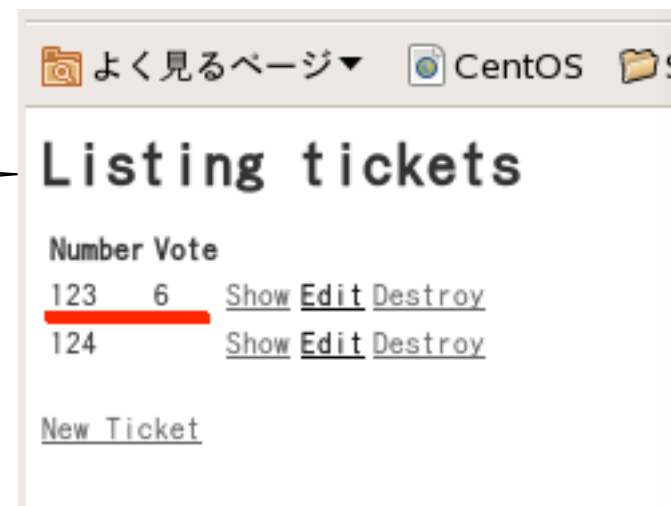
投票権番号の発行機能を、管理者権限のあるログインユーザに限定する。

投票結果の記録(再掲)

@ticket.save

で、投票結果がデータベースに記録されています。
あとは、これを集計するだけで「ランキング」画面が作れます。

今日は、この続きから
いきたいと思います。



Number	Vote	
<u>123</u>	6	Show Edit Destroy
124		Show Edit Destroy

[New Ticket](#)

ランキング画面の設計

登録されている全てのデータについて、イテレータを実行して、ticketsに記録されたvoteの記録を集計します。

画面名は、rankingにしましょう。

投票部分は、投票権番号の制約をかけました。
Rankingは誰でも見られるようにしたいので、分離してみます。

Rankingコントローラの製作

今回は、rankingsというテーブルは作りません。
従って、コントローラ名はrankingという単数形でも問題ないはずですが・・・。
(念のため、何か設計変更がかかっても困らないように・・・)複数形で作っておきます。

```
rails g controller rankings ranking
```

コントローラの生成

```
[root@cisnote vegetable-market]# rails g controller rankings ranking
  create  app/controllers/rankings_controller.rb
  route  get 'rankings/ranking'
  invoke erb
  create  app/views/rankings
  create  app/views/rankings/ranking.html.erb
  invoke rspec
  create  spec/controllers/rankings_controller_spec.rb
  create  spec/views/rankings
  create  spec/views/rankings/ranking.html.erb_spec.rb
  invoke helper
  create  app/helpers/rankings_helper.rb
  invoke rspec
  create  spec/helpers/rankings_helper_spec.rb
  invoke assets
  invoke coffee
  create  app/assets/javascripts/rankings.coffee
  invoke scss
  create  app/assets/stylesheets/rankings.scss
[root@cisnote vegetable-market]#
```


生成結果の確認

`config/routes.rb`

に、`ranking`のパスが追加されています。

```
1 Rails.application.routes.draw do
2
3   get 'rankings/ranking'
4
5   post "votes/check" => "votes#check"
6   get "votes/login" => "votes#login"
7
```

`app/controllers/rankings_controller.rb`

`app/views/rankings/ranking.html.erb`

が生成されています。

```
1 class RankingsController < ApplicationController
2   def ranking
3   end
4 end
5
```

Ranking表示の考え方

まず、merchandisesテーブルから、全部のデータを取得します。

(再三、しつこいようですが、皆さんは自分の「投票」対象のデータを取得して下さい。)

テンプレート(ranking.html.erb)へは、多重配列で値を渡すことにします。

設計方針

投票したチケットの「投票先」が、vegetable.idであるものを抽出する。

(この部分、SQLでcount(*)を使うことも可能だが、インスタンスの情報を表示に使うため、レコードをそのまま抽出)

投票数、vegetableインスタンス、0(順位の初期値)の三要素を持った配列rankを作り、それをさらに上位の配列、@ranksに追加していく。

@ranksを降順に並べ替える。

同率順位に注意しながら、「順位」を与えていく。

順位を与えるロジック

各自、コードを読んで考えてください。

次の順位(count)に、何位を加算するか → add

通常は、1位ずつ順位が下がる。

同率があった場合は、その数だけさらに下がる。

並べ替えた直前の投票数(last)が、自分の投票数と等しかったら、同じ順位に設定する。

Ranking controller

```
class RankingsController < ApplicationController
  def ranking
    merchandises = Merchandise.all
    @ranks = Array.new()
    merchandises.each do |vegetable|
      lists = Ticket.where( vote: vegetable.id )
      rank = Array.new([lists.count, vegetable, 0])
      @ranks << rank
    end
    @ranks.sort!
    @ranks.reverse!

    count = 0; last = 0; add = 1
    @ranks.each do |rank|
      if last==rank[0] then
        add += 1
      else
        count += add ; add = 1
      end
      rank[2] = count
      last = rank[0]
    end
    p @ranks
  end
end
```

Ranking controller

```
1 class RankingsController < ApplicationController
2   def ranking
3     merchandises = Merchandise.all
4     @ranks = Array.new()
5     merchandises.each do |vegetable|
6       lists = Ticket.where( vote: vegetable.id )
7       rank = Array.new([lists.count, vegetable, 0])
8       @ranks << rank
9     end
10    @ranks.sort!
11    @ranks.reverse!
12
13    count = 0; last = 0; add = 1
14    @ranks.each do |rank|
15      if last==rank[0] then
16        add += 1
17      else
18        count += add ; add = 1
19      end
20      rank[2] = count
21      last = rank[0]
22    end
23    p @ranks
24  end
25 end
26
```

Ranking controllerの読み方(1)

3行目で、まず全ての商品を取得します。

4行目:画面に渡す多重配列 @ranksを初期化します。

5~9行目:野菜ごとに以下の処理を行います。

6行目:その野菜に投票した票の一覧を作成

7行目:配列rankに、得票数、Merchandiseインスタンス、順位用の整数の3項目を代入します。

8行目:配列rankを ranksに追加します。

10行目:得票数の昇順で並べ替えます。

11行目:順序を入れ替えて、降順にします。

Ranking controllerの読み方(2)

13行目から22行目で、「順位」を設定します。

考え方:最初の順位は1位とする。

次の順位との間隔をaddで指定し、同じ得票数のデータが連続したら、addを1増やす。

(例:1位5票、2位4票、同数2位4票なら、その次は3位ではなく、4位とします。)

countが「何位」の値、lastは一つ前の得票数、addは、次の順位にいくつ加算したら良いか。

同数ならば、addは増やすが、順位は増やさない。

ranking.html.erb

```
<h1>Rankings#ranking</h1>
```

```
<% @ranks.each do |rank| %>
```

```
  <h2>
```

```
    No. <%= rank[2] %> :
```

```
    <%= rank[1].name %> got <%= rank[0] %> ballots:
```

```
  </h2>
```

```
  <%= image_tag url_for({:action => 'photo', :id=>
    rank[1].id,
```

```
                    :controller => 'merchandises',
```

```
                    :filename => rank[1].file_name}),
```

```
  :alt => rank[1].file_name, :size=>"80x60" %>
```

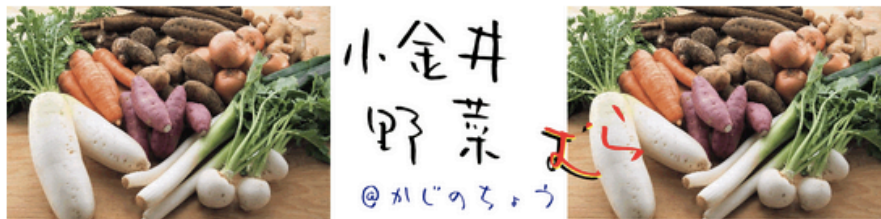
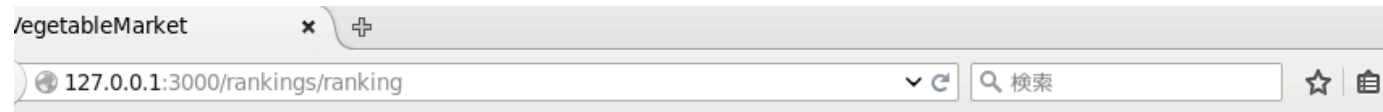
```
  <br />
```

```
<% end %>
```

app/views/rankings/ranking.html.erb

```
1 <h1>Rankings#ranking</h1>
2
3 <% @ranks.each do |rank| %>
4   <h2>
5     No. <%= rank[2] %> :
6     <%= rank[1].name %> got <%= rank[0] %> ballots:
7   </h2>
8     <%= image_tag url_for({:action => 'photo', :id=> rank[1].id,
9                           :controller => 'merchandises',
10                          :filename => rank[1].file_name}),
11                          :alt => rank[1].file_name, :size=>"80x60" %>
12
13   <br />
14 <% end %>
15
```

Ranking画面



[商品一覧](#) | [自己登録](#) | [ログイン](#) | [当サイトについて\(工事中\)](#)

Rankings#ranking

No. 1: キュウリ got 1 ballots:



No. 1: ゆず胡椒 got 1 ballots:



Koganei Vegetable Market

[Sign In](#)
[Sign Up](#)
[About Us](#)

Copyright renounced 2015 by I.Kobayashi

すみません、得票数
1票で1位です...

問題(1)

現在、

`http://127.0.0.1:3000/rankings/ranking`
で順位を表示していますが、

`http://127.0.0.1:3000/ranking`
で順位を表示するには、どうしたらよいでしょうか？

問題(1) 答え

config/routes.rb

の

get `rankings/ranking' => `rankings#ranking'

を

get `ranking' => `rankings#ranking'

に書き換える。

ログイン時のエラーメッセージ

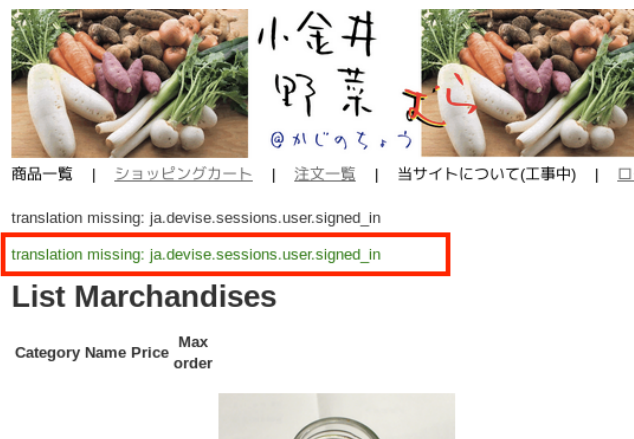
ずっと気になっていたと思いますが・・・

ログインする際に、先頭部分に以下のメッセージが表示されています。

translation missing: ja.devise.failure.user.unauthenticated

translation missing: ja.devise.sessions.user.signed_in

これは一体、何のメッセージでしょうか？



日本語化を完成させる

あちこちの画面で、「英語」のままで作りました。

これらを日本語にするには、どうしたら良いでしょうか。

人気

表示のメッセージを、翻訳のためのシンボルに置き換えます。

例えば、ランキング画面の見出しは

```
<h1>Rankings#ranking</h1>
```

でしたが、これを

```
<h1><%= t :ranking_title %></h1>
```

と、シンボル呼び出し+翻訳に切り換えます。ただ、そのままでは日本語辞書がないため、

```
config/locales/ja.yml, config/locales/en.yml
```

などに、翻訳情報を登録します。

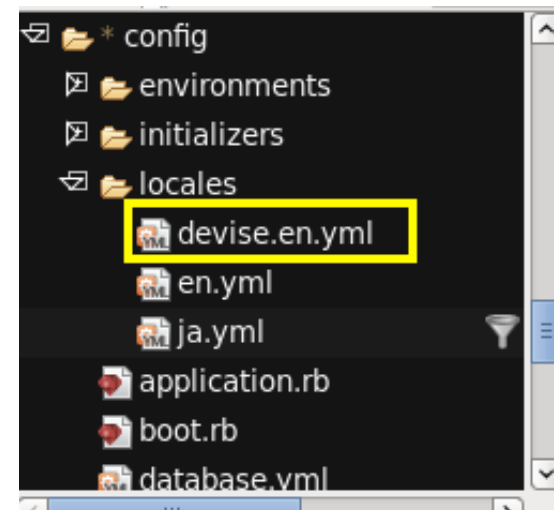
Devise用の辞書ファイル

メッセージが言っているのは、

ja.devise...

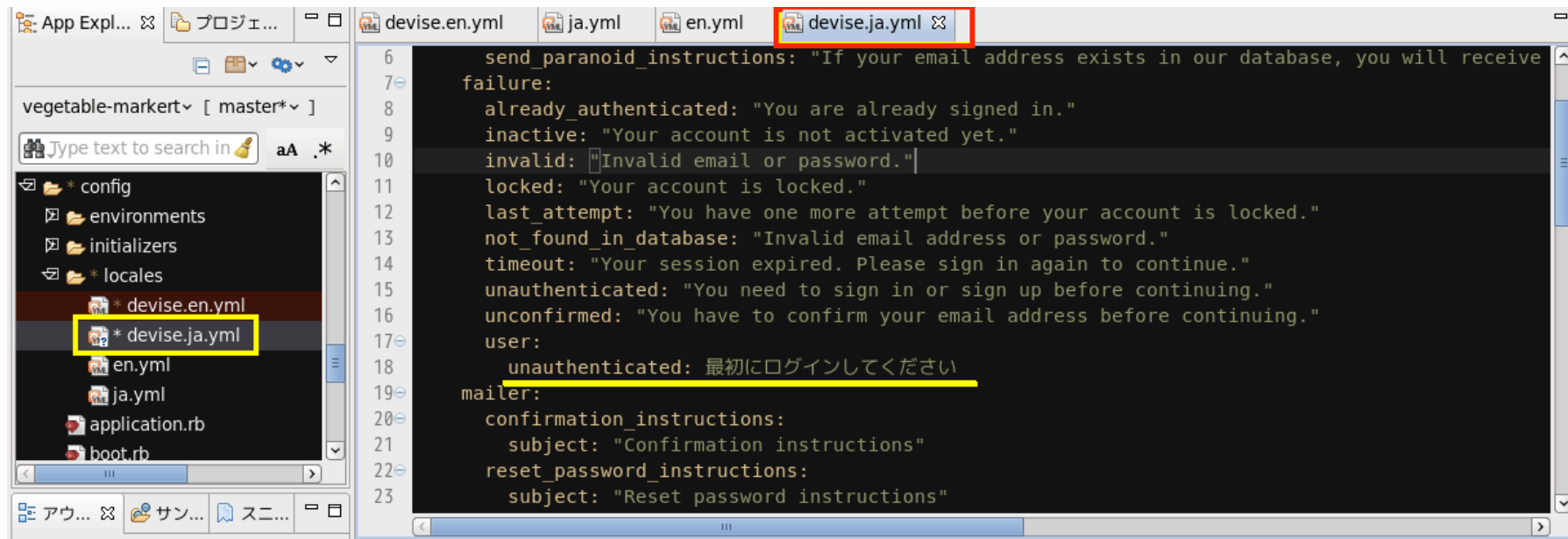
がない、ということです。 **config/locales**の下を見てみると、`devise.en.yml`はあるものの、`devise.ja.yml`がありません。

`devise.en.yml`を丸ごとコピーして
`devise.ja.yml`にしてしまいましょう。



devise.ja.ymlへのコピー

devise.en.ymlからコピーし、日本語にしていきます。
丸ごとコピーした後、先頭のenをjaにします。



```
6  send_paranoid_instructions: "If your email address exists in our database, you will receive
7  failure:
8  already_authenticated: "You are already signed in."
9  inactive: "Your account is not activated yet."
10 invalid: "Invalid email or password."
11 locked: "Your account is locked."
12 last_attempt: "You have one more attempt before your account is locked."
13 not_found_in_database: "Invalid email address or password."
14 timeout: "Your session expired. Please sign in again to continue."
15 unauthenticated: "You need to sign in or sign up before continuing."
16 unconfirmed: "You have to confirm your email address before continuing."
17 user:
18   unauthenticated: 最初にログインしてください
19 mailer:
20 confirmation_instructions:
21   subject: "Confirmation instructions"
22 reset_password_instructions:
23   subject: "Reset password instructions"
```

メッセージの修正

ja.devisе.failure..とノードを辿ります。
failureの下の深さで、例えば一番下の
17行目に userを、そこから一段下げて
18行目に unauthenticatedのメッセージを
追加します。これが

ja.devisе.failure.user.unauthenticated

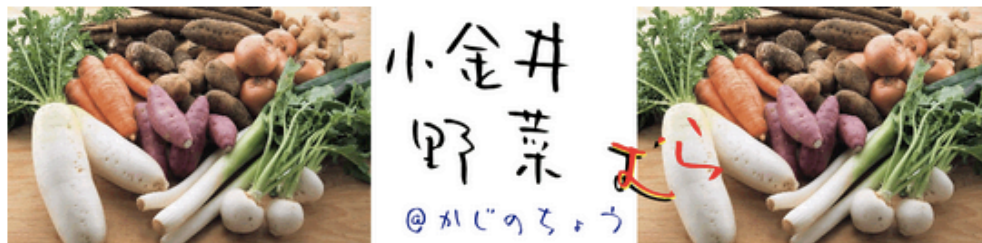
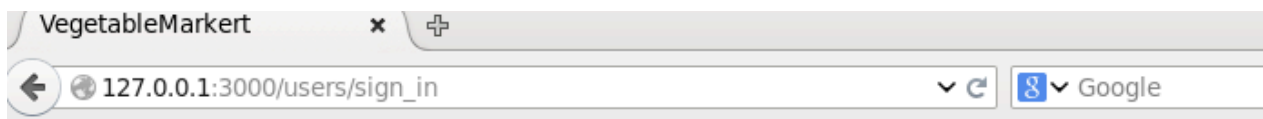
のノードになります。内容は

最初にログインして下さい。

と書き直します。これで、テストランします。

エラーメッセージの日本語化

エラーメッセージが一部日本語化された画面



[List](#) | [Sign Up](#) | [Sign In](#) | [About Us\(工事中\)](#)

最初にログインしてください

Log in

Email

Password

Remember me

[Sign up](#)

[Forgot your password?](#)

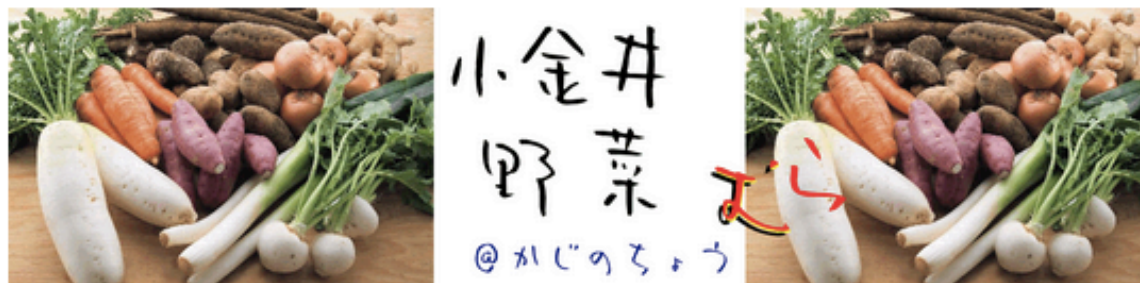
This page is
Recommend

Today's Fres
Cucumber
Egg Plant
Tomato

Halloween S
Pumpkin

取り急ぎの、日本語化(多国語化)

たったの1票で一位ですが・・・



[商品一覧](#) | [ショッピングカート](#) | [注文一覧](#) | [当サイトについて\(工事中\)](#) | [ログ](#)

人気投票結果

No. 1 : キュウリ got 1 ballots:



No. 1 : ゆず胡椒 got 1 ballots:

引き続きの画面修飾

画面の全体を分割して、ヘッダー、メニュー、サイドバーなどを作成した画面修正を、段階的に追加して行って、一応の形は整います。

画面修飾は、各自が行って下さい。
レポート評価に含めます。

ショッピング・サイトの発展形

後期の題材で取り上げたショッピング・サイトは、あくまでも「考えた内容を具体化する」題材の一つです。

こういうことがあるから、こうする、という流れを経験することが主題なので、「ショッピング・サイト」としての実用性にまでこだわる必要はありませんが、まず雰囲気をつかんだら、自分なりに「こういうことも必要ではないか」という内容を盛り込んで、最後のレポートに仕上げてください。

最終課題

自分で作成したシステムを報告して下さい。

どんなシステムを作ろうとしたか。

何が出来るか。

どんな画面を作ったか。

最も工夫した点はどこか。

特に工夫した部分のソースコードはどうなって、どの部分でどんな処理をしたか。

書式には特に制約をつけませんが、一般的な「報告」として形になっている(と自分が理解している)形式で書いて下さい。

欠席課題

Rankingの画面を仕上げて下さい。