

WEB+DBシステム(応用編)



第14回(2015年12月17日)
人気投票サイトの制作(2/3)

人気投票画面(こんな感じ:再掲)

好きな野菜に投票しよう！

あなたの投票権番号: 12345



トマトに一票



現在は、大根に投票済みです。



カボチャに一票

こんな感じの画面
イメージから、設計
を始めて見る。

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

欲を出すと大変なので、
今回作るのはここまで
にする。

今日の目標

人気投票画面で「投票権番号」の有効を確認。
ユーザの、投票権番号の入力画面の処理。

人気投票画面で「投票」を記録する。

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。

投票権入力の方(再掲)

Voteクラスのインスタンスで、投票権番号を入力させます。このインスタンスデータは、ログイン時のデータ入力の確認にのみ使います。

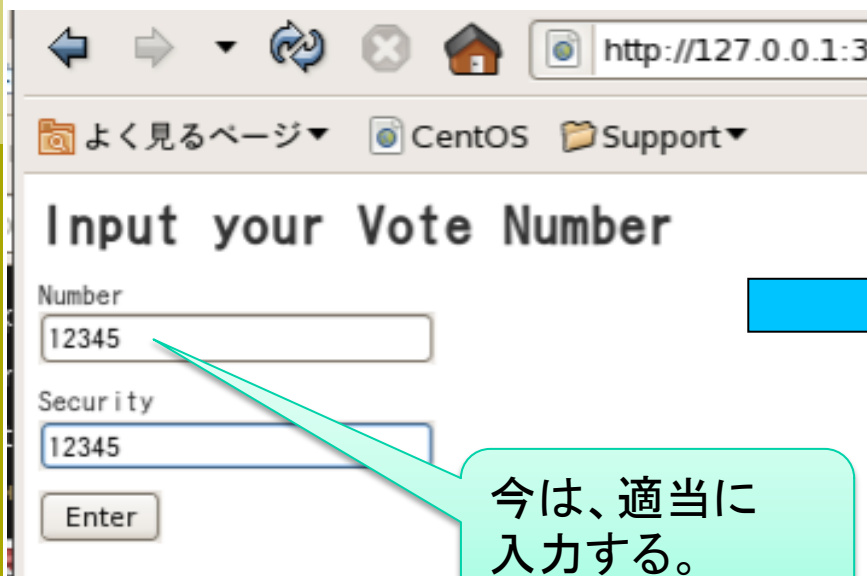
投票権番号とセキュリティコードをデータとして持つので、このクラスのインスタンスを受け取り、Ticketクラスのデータと照合します。

この番号が有効だったら、Ticketsクラスからデータを取得し、vote画面で制御に使います。

前回最終のテストラン

<http://127.0.0.1:3000/votes/login>

で、一般ユーザの「投票権確認」を行います。
番号を入力したら、投票画面へリンクすることを確認します。



よく見るページ ▾ CentOS Support ▾

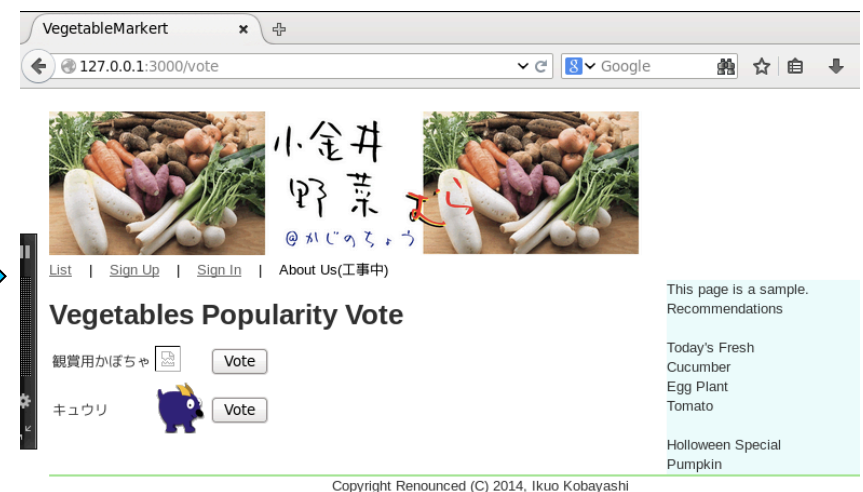
Input your Vote Number

Number
12345

Security
12345

Enter

今は、適当に入力する。



投票権番号の取り出し

login画面では、Voteクラスのインスタンスを
form_forで指定しました。そこで、このパラメータを
取り出します。

```
@vote = Vote.new( params[:vote])
```

POSTされたパラメータを取り出す場合は、

```
params[:name]
```

で取り出します。さらに、Voteクラスのインスタンスとして再編します。

パラメータの有効確認

Checkメソッドを、以下のように書いてみます。

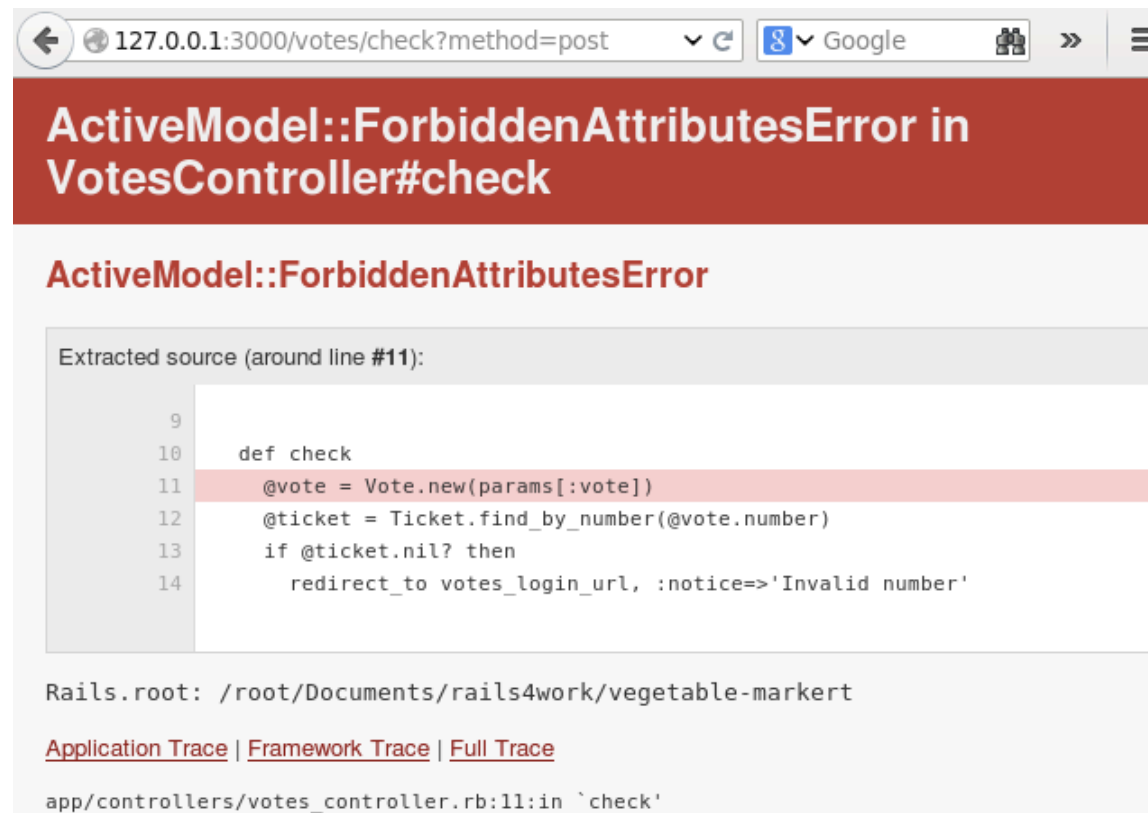
「投票権番号」が、有効かどうかの確認です。

有効なら、投票画面に、無効ならInvalid numberの表示を出して、再びlogin画面に戻ります。

```
def check
  @vote = Vote.new(params[:vote])
  @ticket = Ticket.find_by_number(@vote.number)
  if @ticket.nil? then
    redirect_to votes_login_url, :notice=>'Invalid number'
  else
    redirect_to votes_index_url
  end
end
end
```


自動生成でない、アクセス許可

このままテストランすると、以下のエラーが表示されます。これは、手作業で修正する必要があります。



```
127.0.0.1:3000/votes/check?method=post
```

ActiveModel::ForbiddenAttributesError in VotesController#check

ActiveModel::ForbiddenAttributesError

Extracted source (around line #11):

```
9
10 def check
11   @vote = Vote.new(params[:vote])
12   @ticket = Ticket.find_by_number(@vote.number)
13   if @ticket.nil? then
14     redirect_to votes_login_url, :notice=>'Invalid number'
```

Rails.root: /root/Documents/rails4work/vegetable-markert

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

app/controllers/votes_controller.rb:11:in `check'

ForbiddenAttributesError対応

前述のエラーに対応するためには、

```
@vote = Vote.new(params[:vote])
```

を

```
@vote = Vote.new(params[:vote].permit(:number,:security))
```

に修正します。

```
8 def login
9   @vote = Vote.new
10  end
11
12 def check
13   @vote = Vote.new(params[:vote].permit(:number,:security))
14   @ticket = Ticket.find_by_number(@vote.number)
15   if @ticket.nil? then
16     redirect_to votes_login_url, :notice=>'Invalid number'
17   else
18     redirect_to votes_index_url
19   end
20 end
21
22 def vote
```

TicketsControllerのエラー対応

resond_withを使う際に、以下のエラーが出る場合があります。その際には

`respond_to :html, :xml, :json`

を追記します。

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:3000/tickets`. A red error banner at the top reads "RuntimeError in TicketsController#index". Below the banner, a message states: "In order to use respond_with, first you need to in the class level." To the right, the Rails console shows the code for `TicketsController`. The `respond_to :html, :xml, :json` line is highlighted with a yellow box. The console also shows the `index` action code, which calls `respond_with(@tickets)`.

```
class TicketsController < ApplicationController
  before action :set_ticket, only: [:show, :edit, :update, :destroy]
  respond_to :html, :xml, :json

  def index
    @tickets = Ticket.all
    respond_with(@tickets)
  end

  def show
    @ticket = Ticket.find(params[:id])
    respond_with(@ticket)
  end

  def new
    @ticket = Ticket.new
  end
end
```

投票権番号の制約(第13回P16再掲)

今回は、以下のようにする。

「主」となる番号は、3桁(テーブルに保存)とする。

100～999

これに、チェックコードを付加する。

チェックコードでは、bit rotation, EXORなどの演算で、元のコードから類推しにくいものを作成する。

ユーザには、「6桁」の投票権番号が渡るものとする。

例: 100 - 951, 101 - 208(適当ですが…)

見破られにくいチェックコードの合成方法については、各自「暗号論」などで調べて下さい。
この授業の守備範囲外とします。但し、優れたものは加点対象とします。

チェックコード部分は、毎回計算で求める。

設計方針を
再確認

セキュリティコード入力

チェックコードを加えて6桁の入力とするのではなく、
本体コードとチェックコードの両方を入力させる方式
にしてしまいました。(うっかりしていました。)

Rotateのかけ方など、複雑なロジックを作れる人は、
試してみてください。

(レポート評価の際に、加点対象とします。)

こんなページがありました。

http://detail.chiebukuro.yahoo.co.jp/qa/question_detail/q1339774958

detail.chiebukuro.yahoo.co.jp/qa/question_detail/q1339774958

グッドデザイン賞受賞 デザイナーズ住宅の室内を体感。

YAHOO! JAPAN 知恵袋 IDでもっと便利に新規取得 ログイン

トップ カテゴリ ランキング Q&A一覧 回答コーナー ★10周年★

Q キーワードで探す Q&A 検索 + 条件指定 Q 質

☐ すべてのカテゴリ ☒ PHP

知恵袋トップ > コンピュータテクノロジー > プログラミング > PHP

簡単にクラックできないシリアルコードの発行...

シェア ツイート B! ブクマ 1 + 知恵コレ

ssykg422さん 2010/4/21 14:55:47

セキュリティコード生成

app/helpers/application_helper.rb

に、以下のコードを書きました。

```
module ApplicationHelper
  def security_code( code, mask="0110110001011101", rotate=5 )
    maskbit = mask.to_i(2)
    d = code ^ maskbit
    x = "1"*rotate # => "11111"
    d1 = d >> rotate
    x2 = "1"*rotate + "0"*(16-rotate)
    d2 = ((d << (16-rotate)) & x2.to_i(2))
    v = d1 | d2
    return v
  end
end
```

セキュリティコードの照合

`app/controllers/votes_controller.rb`

の先頭部分に

`include ApplicationHelper`

と書きます。

その上で、`check`メソッドを、次のページのように書き換えます。

checkメソッド

```
def check
  @vote = Vote.new(params[:vote].permit(:number, :security))
  @ticket = Ticket.find_by_number(@vote.number)
  if @ticket.nil? then
    redirect_to votes_login_url, :notice=>'Invalid number'
  else
    if security_code(@vote.number) == @vote.security then
      redirect_to votes_path
    else
      redirect_to votes_login_url, :notice=>'Invalid Security Code'
    end
  end
end
```

Checkメソッド

```
votes_controller.rb login.html.erb tickets_co
1 include ApplicationHelper
2
3 class VotesController < ApplicationController
4   def index
5     @merchandises = Merchandise.all
6   end
7
```

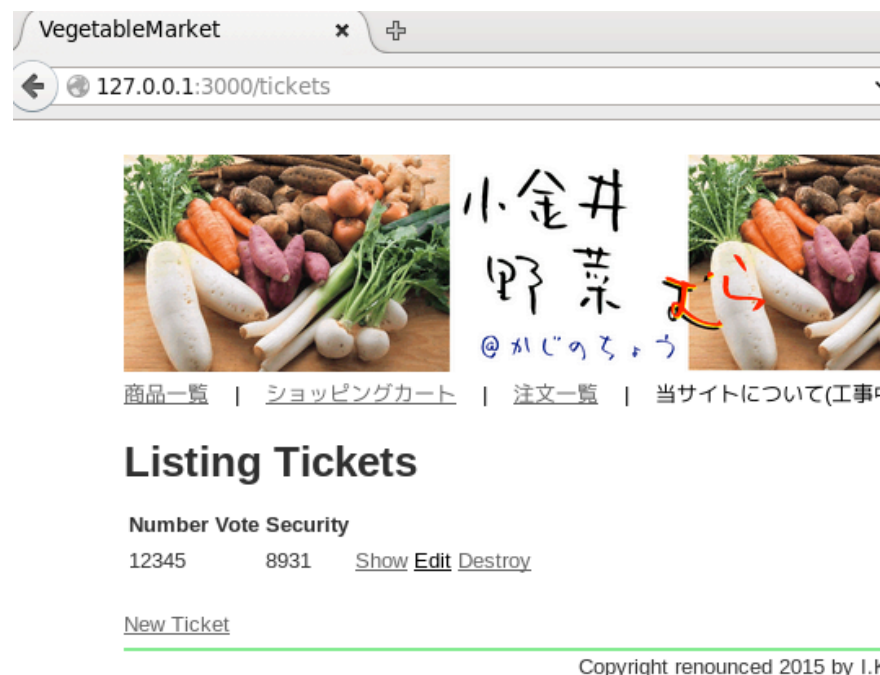
```
14
15 def check
16   @vote = Vote.new(params[:vote].permit(:number, :security))
17   @ticket = Ticket.find_by_number(@vote.number)
18   if @ticket.nil? then
19     redirect_to votes_login_url, :notice=>'Invalid number'
20   else
21     if security_code( @vote.number )==@vote.security then
22       redirect_to votes_path
23     else
24       redirect_to votes_login_url, :notice => 'Invalid Security Code'
25     end
26   end
27 end
28 end
29
```

忘れずに、Ticketsを発行し、確認

<http://127.0.0.1:3000/tickets>

で、scaffoldしたticketsで、「new ticket」を登録し、そのsecurity_codeを画面で見られるように、工夫してください。

(indexを修正すれば、見られます。)



動作確認

ここまでのプログラム修正で、

<http://127.0.0.1:3000/votes/login>

で、Ticketsで登録した「投票権番号」が有効か、確認されて、無効な「セキュリティコード」の場合には、再びログイン画面に戻ることを確認して下さい。

投票時の、チケット確認

これまで、投票画面では「チケット」を無確認でしたが、有効な投票権番号を入力しないと投票できないようにします。

同時に、投票した番号を、記録して行きます。

修正箇所は3カ所です。

(1) checkメソッドの修正

(2) indexメソッドの修正

(3) index.html.erbで、投票権番号の表示

Redirect_toでパラメータを渡す

Checkメソッドから、indexメソッドにパラメータを渡すために、

redirect_to votes_path

を以下のように書き換えます。

```
redirect_to :controller => 'votes', :action=>'index',  
           :number=>@ticket.number
```

これで、:numberというパラメータを渡すことができます。

Checkメソッド(パラメータ渡・修正後)

```
12
13 def check
14   @vote = Vote.new(params[:vote].permit(:number, :security))
15   @ticket = Ticket.find_by_number(@vote.number)
16   if @ticket.nil? then
17     redirect_to votes_login_url, :notice=>'Invalid number'
18   else
19     if security_code(@vote.number) == @vote.security then
20       redirect_to :controller => 'votes', :action=>'index',
21                 :number=>@ticket.number
22     else
23       redirect_to votes_login_url, :notice=>'Invalid Security Code'
24     end
25   end
26 end
27
```

Indexメソッド

投票権を確認し、番号を渡されていない場合には
ログイン(投票権番号入力)に画面をredirectする
ようにします。

```
def index
  @merchandises = Merchandise.all
  @ticket = Ticket.find_by_number(params[:number])
  if @ticket.nil? then
    redirect_to votes_login_url
  end
end
```

```
1 class VotesController < ApplicationController
2   include ApplicationHelper
3   def index
4     @cars = Car.all
5     @ticket = Ticket.find_by_number(params[:number])
6     if @ticket.nil? then
7       redirect_to votes_login_url
8     end
9   end
end
```


Index.html.erb

app/views/votes/index.html.erb
に、以下の一行を追加します。

Your Tickets number is <%= @ticket.number %>.
」

```
1 <h1>Vegetables Popularity Vote</h1>
2 <table>
3 Your Tickets number is <%= @ticket.number %>.<br />
4
5 <%= @merchandises.each do |vegetable| %>
6   <tr>
7     <td><%= vegetable.name %></td>
8     <td><%= image_tag url_for({:action => 'merchandises
```

Vegetables Popularity Vote

Your Tickets number is 12345.

ゆず胡椒



Vote



投票結果の受け渡し

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

の画面で、各投票ボタンをクリックした時の動作をプログラムします。

先週のプログラムで、`<% = %>`で、タグに`=`を忘れていました。

23ページと、24ページのスライドです。

まとめて修正します。

app/views/votes/index.html.erb

```
<h1>Vegetables Popularity Vote</h1>
<div align="right">
  Your Tickets number is <%= @ticket.number %>.<br />
  あなたの投票券番号は、<%= @ticket.number %>です。<br />
</div>
<table>
<% @merchandises.each do |vegetable| %>
  <tr>
    <td><%= vegetable.name %></td>
    <td><%= image_tag url_for({:action => 'photo',
      :controller => 'merchandises',
      :id=> vegetable.id,
      :filename => vegetable.file_name}),
      :alt => vegetable.file_name %></td>
    <td>
      <%= form_tag '/votes/vote' do %>
        <%= hidden_field_tag :vegetable_id, vegetable.id %>
        <%= tag :input, {:type=>'hidden', :name=>'ticket',
          :value => @ticket.number } %>
        <%= submit_tag 'Vote', :name=>'vote' %>
      <% end %>
    </td>
  </tr>
<% end %>
</table>
```

app/views/votes/index.html.erb

```
1 <h1>Vegetables Popularity Vote</h1>
2 <table>
3 Your Tickets number is <%= @ticket.number %>.<br />
4
5 <% @merchandises.each do |vegetable| %>
6   <tr>
7     <td><%= vegetable.name %></td>
8     <td><%= image_tag url_for({:action => 'photo',
9       :controller => 'merchandises', :id=> vegetable.id,
10      :filename => vegetable.file_name}),
11      :alt => vegetable.file_name, :size => "80x80" %></td>
12   <td>
13     <%= form_tag '/votes/vote' do %>
14       <%= hidden_field_tag :vegetable_id, vegetable.id %>
15       <%= tag :input, {:type=>'hidden', :name=>'ticket',
16         :value => @ticket.number } %>
17       <%= submit_tag 'Vote', :name=>'vote' %>
18     <% end %>
19   </td>
20 </tr>
21 <% end %>
22 </table>
23
```

パラメータの受け渡し部分

二通りの書き方をしました。

```
<%= hidden_field_tag :vegetable_id,  
vegetable.id %>
```

と

```
<%= tag :input, { :type=>'hidden', :name=>'ticket',  
:value => @ticket.number } %>
```

です。

上の例では、params[:vegetable_id]という名前で、vegetable.idの値を受け取ります。

下の例では、params[:ticket]という名前で@ticket.numberの値を受け取ります。

受け取ったパラメータの確認

voteメソッドを

```
def vote
  p 'Vote:'.concat(params[:ticket])
  p 'Vegetable:'.concat(params[:vegetable_id])
end end
```

と書き換えてみます。コンソールに値が表示されます。

```
38 def vote
39   p 'Vote:'.concat(params[:ticket])
40   p 'Vegetable:'.concat(params[:vegetable_id])
41 end
42
43
```

Started POST "/votes/vote" for 127.0.0.1 at 2015-01-02 03:18:30 +0900
Processing by VotesController#vote as HTML
Parameters: {"utf8"=>"✓", "authenticity_token"=>"528ygx8yYyZapb0+UT6ZEjNK9luMnrZyv7La6CLXv/s=",
"vegetable_id"=>"2", "ticket"=>"101", "vote"=>"Vote"}
"Vote:101"
"Vegetable:2"
Rendered votes/vote.html.erb within layouts/application (0.5ms)
Rendered shared/_menu_bar.html.erb (0.7ms)
Rendered shared/_right_bar.html.erb (0.1ms)
Rendered shared/_footer.html.erb (0.2ms)
Completed 200 OK in 161ms (Views: 160.1ms | ActiveRecord: 0.0ms)

voteメソッド

app/controllers/votes_controller.rb

のvoteメソッドを、以下のようにします。

```
def vote
  @ticket = Ticket.find_by_number(params[:ticket])
  @ticket.vote = params[:vegetable_id]
  @ticket.save
  @merchandise = Merchandise.find_by_id(@ticket.vote)
end
```

```
32 def vote
33   @ticket = Ticket.find_by_number(params[:ticket])
34   @ticket.vote = params[:vegetable_id]
35   @ticket.save
36   @merchandise = Merchandise.find_by_id(@ticket.vote)
37 end
38
```

投票結果の表示

app/views/votes/vote.html.erb
で、投票結果を表示します。

```
<h1>Votes#vote</h1>
```

```
The vegetable you have voted was : <%= @merchandise.name %>
```

@merchandiseをcontrollerから渡されて、ここで画像を表示することもできます。

```
1 <h1>Votes#vote</h1>
2 The car you have voted was : <%= @car.name %>
3
```

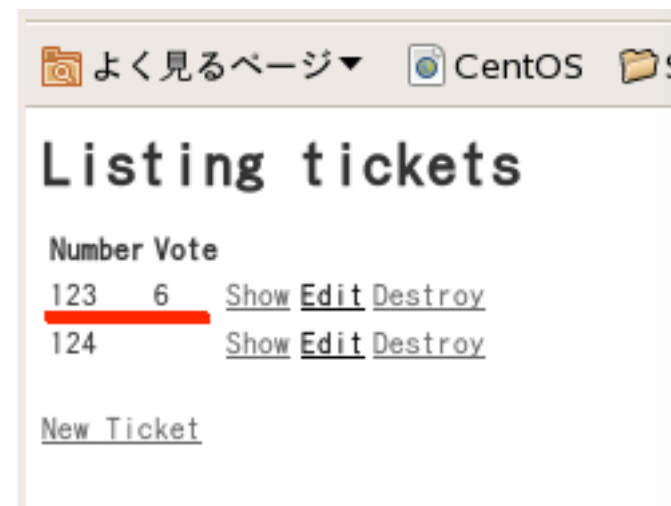
Votes#vote

The car you have voted was : Prius

投票結果の記録

@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。



実習課題

やっと、人気投票結果の保存までたどり着きました。

プログラム各部を読み替えることで、中身が見えてくるはずです。

次回、最終回は、人気投票結果のランキング画面を作ります。

皆さんの課題として、画面の多国語化や、画面修飾も行ってみて下さい。

失敗に学んで下さい。

一通り、テストランさせながら教材を書いています。派手な設計ミスから、初歩的なミス、うっかりミスまで、スライドのあちこちに修正がかかっています。

そのミスも残したまま、修正方法を書くという流れでスライドを作っています。

ただ、簡単なエラーメッセージは、自分で読んで判断し、修正しながら動作させてください。

今日の欠席課題

画像の出る人気投票画面を報告して下さい。

出席に切り換えます。

細かい説明は不要です。画面コピーをつけて下さい。