

WEB+DBシステム(応用編)



第6回(2016年10月27日)

Main Controllerの構造と
メニュー分岐、及びテスト

Webプログラミングの基本は？

- 1: ブラウザにHTMLで画面を表示する。
- 2: ブラウザからのリクエストを受け取る。
- 3: パラメータを処理して、レスポンスを返す。

- この1→3に視点を置いたプログラムの読み方
 - 画面上のどこに何を表示し、何を入力させるか。
 - 画面上で入力された何を、どこで受け取るか。
 - 処理した結果をどのように画面に埋め込むか。

- この三つを理解すれば、読めることになる。
 - 今日は、この三つを理解しよう！

TDDの進め方

都合により、1回先送りして、入れ替えました。

元々の予定：

Capbaraを用いた「画面入力」のテストと、モデルのテストなどを追加していきます。

また、Controllerで、一通りすべてのメソッドのテストが実行できるように、追記していきます。

今日やること

流れをトレースしながら、前期の「画面分割」を行います。

memopadで実施した「多国語化」も行います。

(これらは、復習になります。)

画面での「リンク」のクリックからcontrollerがどのように処理して、どう画面を切り替えて行くか、roleを参照しながら「画面アクセスの切り替え」を調べ、実装して行きます。

このため、roleごとのメニューを用意します。

多国語化で修正するファイル群

app/controllers/application_controller.rb

config/initializers/i18n.rb

config/locales/en.yml

config/locales/ja.yml

(第1回スライド参照)

```
1 class ApplicationController < ActionController::Base
2   protect_from_forgery with: :exception
3
4   before_filter :set_locale
5   def set_locale
6     I18n.locale = params[:locale] || I18n.default_locale
7   end
8 end
9
```

```
development.rb  application_controller.rb  i18n.rb ✕
1 Rails.application.config.i18n.default_locale = :ja
2
```

画面分割で修正するファイル群

app/assets/stylesheets/scaffold.scss

app/assets/images/title_en.gif(任意)

app/assets/images/title_ja.gif

app/helpers/application_helper.rb

app/views/layouts/application.html.erb

app/views/shared (新規に作成する)

app/views/shared/_menu_bar.html.erb

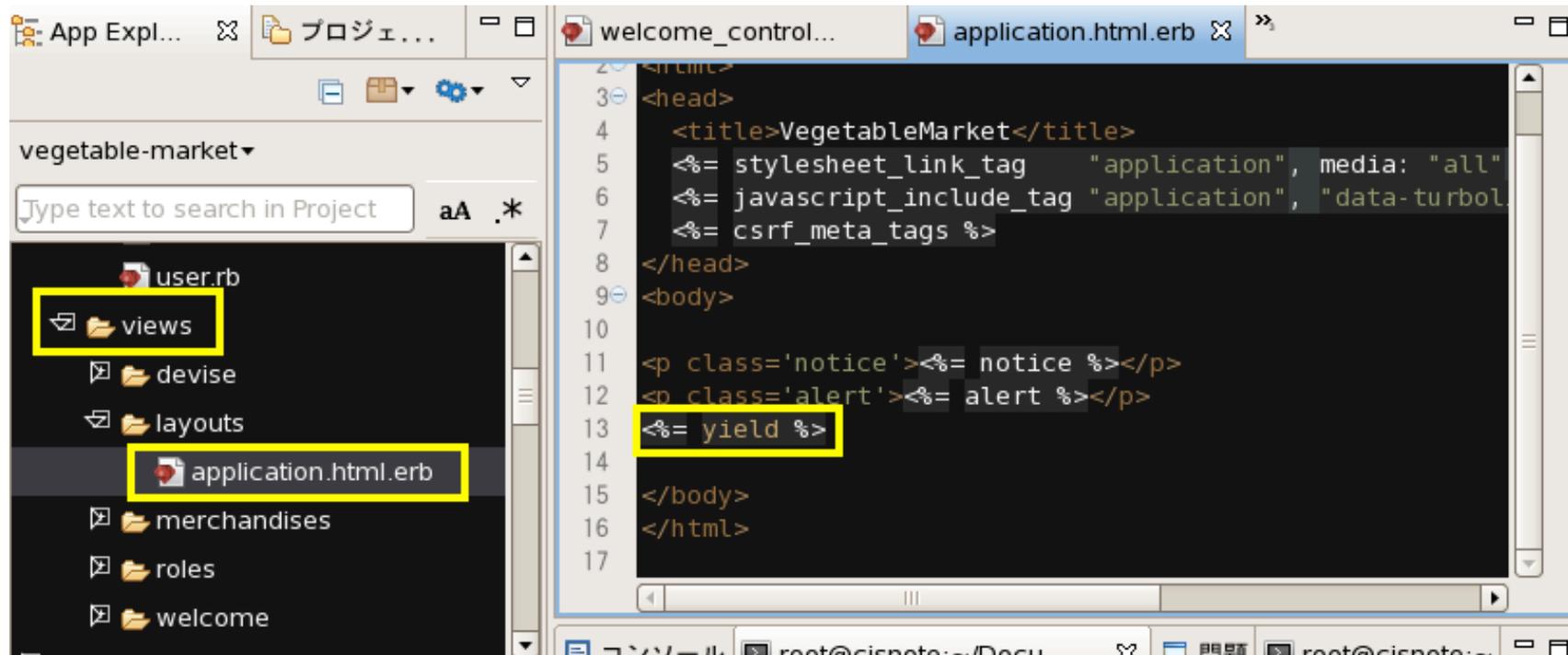
app/views/shared/_right_bar.html.erb

app/views/shared/_footer.html.erb

View: index.html.erb

- ブラウザでURLを指定すると、まず、indexという名前のファイルが探される。
 - index.htm, index.html, index.cgi, index.php
 - index.jsp, index.aspなど
- ここでも、まずindex.html.erbから調べる。
 - その前に...
- app/views/layouts/application.html.erb
 - が、すべてに共通する画面の構造となる。

app/views/layouts/application.html.erb



The screenshot shows a code editor with two windows. The left window displays the file structure of a project named 'vegetable-market'. The 'views' folder is expanded, and 'application.html.erb' is selected. The right window shows the content of 'application.html.erb' with the following code:

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

yieldの部分に、個別のindex.html.erbがはめ込まれる。

app/assets/stylesheets/scaffold.scss

□ 以下の内容を、書き加えてみる

```
div#container{  
  background-color: white;  
  margin: 0 auto;  
  padding: 5px 0 0;  
  width: 800px;  
}
```

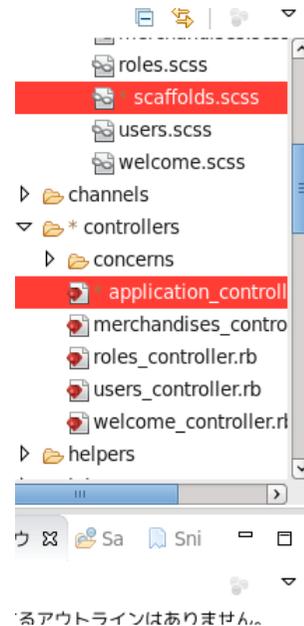
```
div#header {  
  padding-top: 4px;  
  border-top: 4px solid white;  
}
```

```
div#left {  
  float: left;  
  background-color: white;  
  padding: 6px 6px 6px 0;  
}
```

```
div#right {  
  float: right;  
  width: 228px;  
  background-color: #e8ffff;  
}
```

```
div#footer {  
  background-color: white;  
  border-top: 2px lightgreen solid ;  
  text-align: center;  
  clear: both;  
  width: 100%;  
}
```

```
div#menu_bar{  
  color: black;  
}
```



```
46 div {  
47   &.field, &.actions {  
48     margin-bottom: 10px;  
49   }  
50 }  
51  
52 div#container{  
53   background-color: white;  
54   margin: 0 auto;  
55   padding: 5px 0 0;  
56   width: 800px;  
57 }  
58  
59 div#header {  
60   padding-top: 4px;  
61   border-top: 4px solid white;  
62 }  
63  
64 div#left {  
65   float: left;  
66   background-color: white;  
67   padding: 6px 6px 6px 0;  
68 }
```

前期に行ったscreen分割を試す。

前期、第5回のスライドを参照します。

views/sharedのフォルダを作る。

sharedの下に `_menu_bar.html.erb`を置く。

(新規作成:前期第4回:スライドP31~P60)

タイトルバナーに画像を出してみる。

(前期第4回:スライドP32)

宿題として、各自試してみてください。

以下、実装済みの画面を提示します。

views/layout/application.html.erb

```
en.yml  ja.yml  *application.html.erb x
6  <%= javascript_include_tag 'application', 'data-turbolinks-track' => true %>
7  <%= csrf_meta_tags %>
8  </head>
9  <body>
10 <div id="container">
11 <div id="header">
12 <%= if I18n.locale == :ja %>
13 <%= image_tag 'title_ja.gif', :size=>'600x133', :alt => 'Koganei Vegetable Vil.' %>
14 <%= else %>
15 <%= image_tag 'title_en.gif', :size=>'600x133', :alt => 'Koganei Vegetable Vil.' %>
16 <%= end %>
17 <%= render :partial => 'shared/menu_bar' %>
18 </div>
19 <div id='left'>
20 <p class="notice"><%= notice %></p>
21 <p class="alert"><%= alert %></p>
22 <%= yield %>
23 </div>
24 <div id='right'>
25 <%= render :partial => 'shared/right_bar' %>
26 </div>
27 <div id='footer'>
28 <%= render :partial => 'shared/footer' %>
29 </div>
30 </div>
31 </body>
32 </html>
zz
```

views/layout/application.html.erb

画面の分割(前期第4回)

```
<body>
  <div id="container">
    <div id="header">
      <% if I18n.locale == :ja %>
        <%= image_tag 'title_ja.gif', :size=>'600x133', :alt => 'Koganei Vegetable Vil.' %>
      <% else %>
        <%= image_tag 'title_en.gif', :size=>'600x133', :alt => 'Koganei Vegetable Vil.' %>
      <% end %>
      <%= render :partial => 'shared/menu_bar' %>
    </div>
    <div id='left'>
      <p class="notice"><%= notice %></p>
      <p class="alert"><%= alert %></p>
      <%= yield %>
    </div>
    <div id='right'>
      <%= render :partial => 'shared/right_bar' %>
    </div>
    <div id='footer'>
      <%= render :partial => 'shared/footer' %>
    </div>
  </div>
</body>
```

言語によって画像を変えるようにしてみました。

私の用意したサンプル画像



*Koganei
Vegetable
Market*



小金井
野菜

@かじのちょう



予めメニューを作る

Viewsの下にsharedフォルダを作成し、その下に

`_menu_bar.html.erb`

`_right_bar.html.erb`

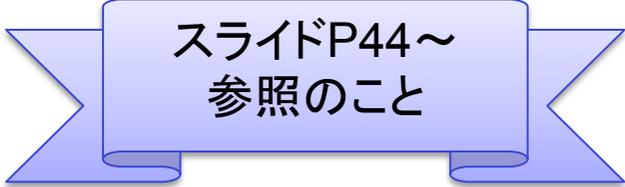
`_footer.html.erb`

などを作成する。

(前期第4回のスライド参照のこと)

役割に応じたメニューをデザイン

```
<% if !current_user %>
  <% menu_items = [
    { :link => merchandises_path, :name => 'List of merchandises' },
    { :link => new_user_registration_path, :name => 'Sign Up' },
    { :link => new_user_session_path, :name => 'Sign In' },
    { :link => "", :name => 'About Us', :disabled => true }
  ] -%>
<% else %>
  <% if current_user.role.ename == "guest" %>
    <% menu_items = [
      { :link => merchandises_path, :name => 'List of merchandises' },
      { :link => "", :name => 'Shopping Cart' },
      { :link => "", :name => 'Orders' },
      { :link => "", :name => 'About Us', :disabled => true },
      { :link => destroy_user_session_path, :name => 'Sign Out', :method => "delete" }
    ] -%>
  <% elsif current_user.role.ename == "staff" %>
    <% menu_items = [
      { :link => merchandises_path, :name => 'List of merchandises' },
      { :link => new_merchandise_path, :name => 'New Merchandise' },
      { :link => "", :name => 'Customer Management' },
      { :link => "", :name => 'Order Management' },
      { :link => destroy_user_session_path, :name => 'Sign Out', :method => "delete" }
    ] -%>
  <% else %>
    <% menu_items = [
      { :link => users_path, :name => 'List Users' },
      { :link => "", :name => 'Master Tables Maintenance', :disabled => true },
      { :link => destroy_user_session_path, :name => 'Sign Out', :method => "delete" }
    ] -%>
  <% end %>
<% end %>
```



スライドP44~
参照のこと

_menu_barでのエラー

Pathが定義されていないと、NameErrorになります。

rake routes

で、pathを確認し、各自の実装状況に合わせて書き換えてください。

Action Controller: Except... x +

127.0.0.1:3000/users

NameError in Users#index

Showing /root/Documents/rails4work/vegetable-market/app/views/shared/_menu_bar

undefined local variable or method `users_index_path` for #<#<Class:0x...>>

Extracted source (around line #27):

```
25 <% else %>
26   <% menu_items = [
27     { :link => users_index_path, :name => 'List Use
28     { :link => "", :name => 'Master Tables Maintena
29     { :link => destroy_user_session_path, :name =>
30     ] -%>
```

```
users GET /users/:id(format) users#index
      POST /users/:id(format) users#create
new_user GET /users/new(:format) users#new
edit_user GET /users/:id/edit(:format) users#edit
user GET /users/:id(:format) users#show
      PATCH /users/:id(:format) users#update
      PUT /users/:id(:format) users#update
      DELETE /users/:id(:format) users#destroy
roles GET /roles/:id(format) roles#index
      POST /roles/:id(format) roles#create
new_role GET /roles/new(:format) roles#new
edit_role GET /roles/:id/edit(:format) roles#edit
role GET /roles/:id(:format) roles#show
      PATCH /roles/:id(:format) roles#update
      PUT /roles/:id(:format) roles#update
      DELETE /roles/:id(:format) roles#destroy
welcome_index GET /welcome/index(:format) welcome#index
user_root GET /merchandises(:format) merchandises#index
merchandises GET /merchandises/:id(format) merchandises#show
      POST /merchandises/:id(format) merchandises#create
```


ログインしていない時と、設定されたroleで、メニューを分ける。

```
<% if !current_user %>
  <% menu_items = [
    (省略)
  ] -%>
<% elsif current_user.role.ename == "guest" %>
  <% menu_items = [
    (省略)
  ] -%>
<% elsif current_user.role.ename == "staff" %>
  <% menu_items = [
    (省略)
  ] -%>
<% else %>
  <% menu_items = [
    (省略)
  ] -%>
<% end %>
```

ログインして
いない時の
メニュー

一般のお客
様のメニュー

従業員の
メニュー

Adminユー
ザのメニュー

Validationで、
roleが1~3
以外にならない
ようにする。

Application_helper

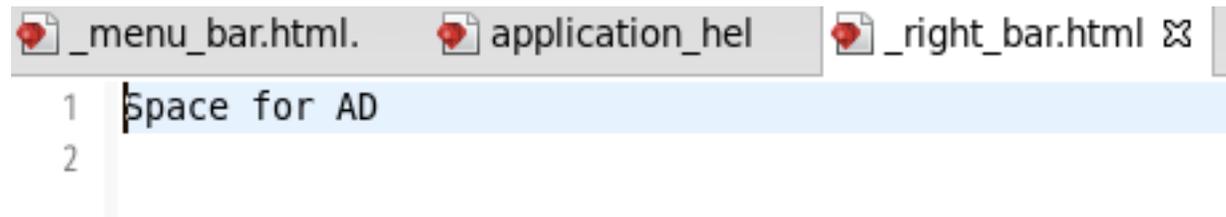
```
#!/ruby
# coding: utf-8
module ApplicationHelper
  def menu_link_to( item )
    if current_page?( item[:link] )
      raw "<span class=¥'current_page¥'>" + item[:name] + "</span>"
    elsif item[:disabled]
      raw "<span class=¥'disabled¥'>" + item[:name] + "(工事中)</span>"
    elsif item[:method]
      link_to(item[:name], item[:link], :method => item[:method])
    else
      link_to(item[:name], item[:link])
    end
  end
end
end
```

app/helpers/application_helper.rb

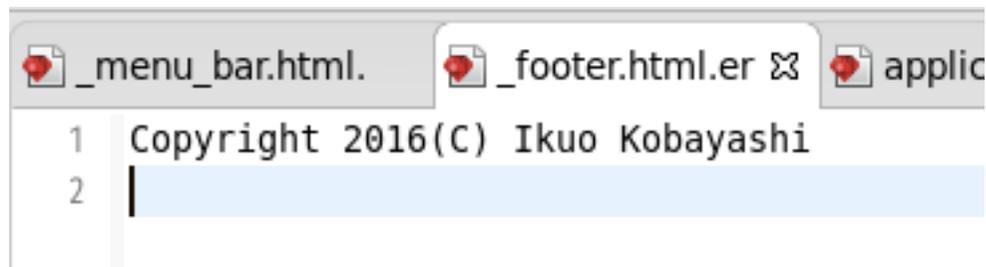
```
_menu_bar.html.  _footer.html.er  application_hel  »7  □
1  #!ruby
2  # coding: utf-8
3  module ApplicationHelper
4  def menu_link_to( item )
5    if current_page?( item[:link] )
6      raw "<span class='current_page'>" + item[:name] + "</span>"
7    elsif item[:disabled]
8      raw "<span class='disabled'>" + item[:name] + "(工事中)</span>"
9    elsif item[:method]
10     link_to(item[:name], item[:link], :method => item[:method])
11   else
12     link_to(item[:name], item[:link])
13   end
14 end
15 end
16
17
```

Right BarとFooter

現在は、仮として入れておきます。



```
_menu_bar.html. application_hel _right_bar.html ✖  
1 |space for AD  
2 |
```



```
_menu_bar.html. _footer.html.er ✖ applic  
1 |Copyright 2016(C) Ikuo Kobayashi  
2 |
```

ここまでで一区切り

画面分割と、他言語化対応は一区切りとします。

ただ、まだ若干問題はあります。

「一般客」が、他のユーザの権限を編集できる。

（一般客が、自分もadminになれてしまう・・・）

「一般客」が、adminユーザを削除できる。

これらへの対応を、今後「テスト」からの記述で考えます。

メニューを整理する

「一般ユーザ」が自分を登録する。

自分のパスワードを変更したい

自分のハンドルは、変更できるようにしたい

「管理者」が、スタッフを含む全ユーザの一覧を見たい。

「スタッフ」が、一般ユーザの一覧を見たい。

これらを目的とする画面切り替えをワンタッチで行えるようにするために、メニューに登録する。

Sign up(自己登録)直後の動作

Sign Up後、user_root(ホーム画面 = merchandisesのindex)に飛びます。

ところが、Devise管理ではemailとパスワード以外のroleやハンドル名が登録されていません。

そこで、merchandises_controllerのindexアクションで、ユーザの編集にリダイレクトします。

後ほど、「一般ユーザは、自分のroleを変更できない」ように修正します。

ログインまでの流れ(復習)

`config/routes.rb` で、

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

を記述した。

そのため、ドメインルートの

```
http://0.0.0.0:3000/
```

を指定すると、まず

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

が表示される。同様に

```
get 'merchandises', :to =>
```

```
'merchandises#index', :as => :user_root
```

を記述したため、`merchandises#index`が`:user_root`となる。



ログイン後のTOPページ

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

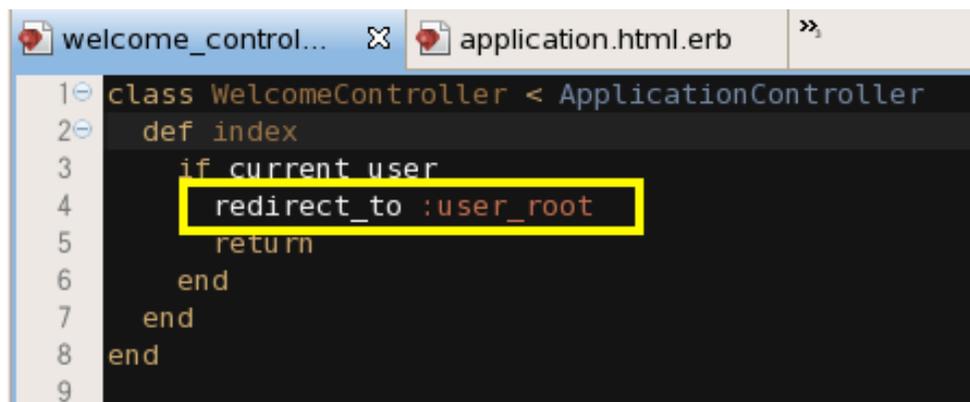
を記述したため、merchandises#indexが:user_rootとなる。

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

は、ドメイントップのコントローラになるが、ログイン後(current_userが設定された後)は、

```
redirect_to :user_root
```

で、:user_rootすなわち、merchandises#indexにリダイレクトされる。



```
welcome_control... application.html.erb »  
1 class WelcomeController < ApplicationController  
2   def index  
3     if current user  
4       redirect_to :user_root  
5     end  
6   end  
7 end  
8  
9
```

app/controllers/ merchandises_controller.rb

indexアクションに、以下のように追記する。

```
# GET /merchandises
# GET /merchandises.json
def index
  if current_user
    if !current_user.handle
      current_user.handle = "No Name"
      current_user.save
      redirect_to :controller=>"users", :action=>"edit",
                  :id=> current_user.id
    end
  end
  @merchandises = Merchandise.all
end
```

app/controllers/ merchandises_controller.rb

```
1 class MerchandisesController < ApplicationController
2   before_action :set_merchandise, only: [:show, :edit, :update, :destroy]
3
4   # GET /merchandises
5   # GET /merchandises.json
6   def index
7     if current_user
8       if !current_user.handle
9         current_user.handle = "No Name"
10        current_user.save
11        redirect_to :controller=>"users", :action=>"edit",
12                    :id=> current_user.id
13
14        return
15      end
16    end
17    @merchandises = Merchandise.all
18
19    # GET /merchandises/1
20    # GET /merchandises/1.json
21  def show
22  end
23
```

ユーザ登録直後の動作

2段階になるけれども、この修正でdeviseによるユーザ登録直後に、アプリからのユーザ登録が行われて、不自然な動作ではなく、「登録」ができるはずで
す。

他にも実装方法はあると思います。

慣れてきたら、各自工夫してみてください。

メニューへの登録方法

コマンド:

```
rake routes
```

で、ルーティングのテーブルを確認する。

```
URL → controller (→ view)
```

Pathを見ながら、それぞれの「role」ごとのメニューについて、こちらも各自工夫してみてください。

スタッフ用ページと一般ページ

Welcome#indexから一般ページをそのまま開くようにする。

merchandises_controller.rb(商品の表示ページ)には、
`before_filter :authenticate_user!`

を記述していない。このため、このページは「一般ページ」として、
ログインしなくても、誰もが表示することができる。

今回の設計としては、Welcome
画面にスタッフ入口へのリンクを
埋め込むものとし、そのActionに
認証要求を組み込む。

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

Sign in

Email

kobayashi.ikuo.t9@k.hosei.ac.jp

Password

.....

Remember me

Sign in

[Sign up](#)

[Forgot your password?](#)

全体指定と個別指定

Controller内部のメソッド全てに共通するパラメータの場合には、

`before_filter :authenticate_user!`

のように、メソッド定義の外部(Classの先頭部分)に記述します。

Actionごとに個別指定する場合には、

`authenticate_user!`

のメソッドを呼び出します。

認証要求のタイミング

ログイン認証がいつ要求されるかは

`before_filter :authenticate_user!`

の記述がある controller を最初に開こうとした時点。

スタッフの場合には、スタッフ用「入口」からのAction実行で認証を要求する。

一般客は、「ショッピングカート」に商品を入れ、「精算」に進んだ先の画面で認証を要求する。

このタイミングは、`devise`に任せる。

プログラマは、それぞれのタイミングで `authenticate_user!` を埋め込めば良い。

Welcome画面の改造

Welcome#indexに、
メニューで「ジャンプ先」を登録する。
二つのリンクを埋め込む。

Welcome to Vegetable Market:

- Jump to a Shopping Mall(画像から)
- Staff Entrance(リンクボタン)

私の作ったEntry画面の例

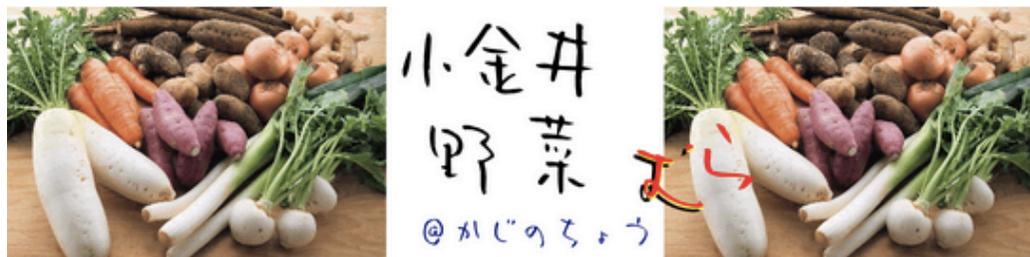
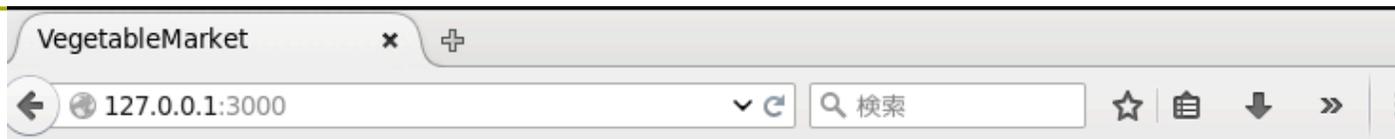
app/views/welcome/index.html.erb

```
<%= link_to image_tag('pumpkin2.jpg', { :border => '0',  
    :size => '300x200', :alt => '商品一覧'}), merchandises_path %><br />  
<%= t :staff_entrance %> ==>  
<%= link_to (t :click_here ), staff_entry_path %>
```

画像の表示はこちらに書いても構いません。

```
1 <h1><%= t :welcome_top %></h1>  
2 <%= link_to image_tag('pumpkin2.jpg', { :border => '0',  
3     :size => '300x200', :alt => '商品一覧'}), merchandises_path %><br />  
4 <%= t :staff_entrance %> ==>  
5 <%= link_to (t :click_here ), staff_entry_path %>  
6
```

Welcome画面



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

広告募集中

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

小金井野菜むらへようこそ！



スタッフ専用入り口 ==> [ここをクリック！](#)

Config/routes.rb

ルーティングテーブルに追加

```
get 'welcome/staff_entry', :to => 'welcome#staff_entry', :as => :staff_entry
```

URLは"welcome/staff_entry"

Welcomeコントローラの、staff_entryメソッドで処理する。

別名として、staff_entryをつける。これを、rubyのプログラムから呼び出すときは、staff_entry_pathと記載する。(一つ前のスライド)

```
3 resources :users
4 resources :roles
5 get 'welcome/index'
6 get 'welcome/staff_entry', :to => 'welcome#staff_entry', :as => :staff_entry
7
8 get 'merchandises', :to => 'merchandises#index', :as => :user_root
9 resources :merchandises
10 # For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html
11
12 root "welcome#index"
```

staff_entryアクションを用意

スタッフ入口では、

認証要求を行い、index画面に切り替える。

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

```
def staff_entry
  authenticate_user!
  if current_user
    redirect_to :user_root
  else
    redirect_to :root
  end
end
```

app/controller/ welcome_controller.rb

```
1 class WelcomeController < ApplicationController
2   def index
3     if current_user
4       redirect_to :user_root
5     return
6   end
7 end
8
9 def staff_entry
10  authenticate_user!
11  if current_user
12    redirect_to :user_root
13  else
14    redirect_to :root
15  end
16 end
17
18 end
19
```

画面表示のための辞書を用意

メニューその他から、多国語化したい辞書の見出し語を用意する。

welcome_top: Welcome to Koganei Vegetable Market!

staff_entrance: Staff Entrance

click_here: Click Here

list_merchandises: List of Merchandises

sign_up: Sign Up

sign_in: Sign In

sign_out: Sign Out

about_us: About Us!

shopping_cart: Shopping Cart

orders_list: List of Orders

new_merchandises: New Merchandises

customer_service: Customer Service

order_management: Order Management

list_users: List of Users

master_table_maintenance: Master Table Maintenance

画面表示のため、辞書を用意

ja.ymlや、en.ymlを作成する。

```
1
2 en:
3   welcome_top: Welcome to Koganei Vegetable Market!
4   staff_entrance: Staff Entrance
5   click_here: Click Here
6   list_merchandises: List of Merchandi
7   sign_up: Sign Up
8   sign_in: Sign In
9   sign_out: Sign Out
10  about_us: About Us!
11  shopping_cart: Shopping Cart
12  orders_list: List of Orders
13  new_merchandises: New Merchandises
14  customer_service: Customer Service
15  order_management: Order Management
16  list_users: List of Users
17  master_table_maintenance: Master Tab
18  activerecord:
19    errors:
20    messages:
21      record_invalid: "Validation fa
22  --
1
2 ja:
3   welcome_top: 小金井野菜むらへようこそ!
4   staff_entrance: スタッフ専用入り口
5   click_here: ここをクリック!
6   list_merchandises: 商品一覧
7   sign_up: 自己登録
8   sign_in: ログイン
9   sign_out: ログアウト
10  about_us: 当サイトについて
11  shopping_cart: ショッピングカート
12  orders_list: 注文一覧
13  new_merchandises: 新規商品登録
14  customer_service: 顧客サービス
15  order_management: 注文管理
16  list_users: ユーザー一覧
17  master_table_maintenance: マスター管理
18  activerecord:
19    errors:
20    messages:
21      record_invalid: "バリデーションに失敗しました: %{errors}"
22      restrict_dependent_destroy:
23        has_one: "%{record}が存在しているので削除できません"
```

Ja.yml

ja:

welcome_top: 小金井野菜むらへようこそ！
staff_entrance: スタッフ専用入り口
click_here: ここをクリック！
list_merchandises: 商品一覧
sign_up: 自己登録
sign_in: ログイン
sign_out: ログアウト
about_us: 当サイトについて
shopping_cart: ショッピングカート
orders_list: 注文一覧
new_merchandises: 新規商品登録
customer_service: 顧客サービス
order_management: 注文管理
list_users: ユーザー一覧
master_table_maintenance: マスター管理

実装例 – スタッフ用メニュー

スタッフ用処理選択画面を用意し、この画面でログイン認証を求める。(deviseに任せる。)

(1) 商品一覧

顧客と同様に、登録済みの商品一覧を表示

(2) 新規商品の追加

いきなり「商品追加」を行う。

(3) 顧客管理

(4) 注文処理

(5) ログアウト

実装例 – 管理者用メニュー

管理者用処理選択画面を用意し、この画面でログイン認証を求める。(deviseに任せる。)

(1) 登録ユーザー一覧

管理者だけが「ユーザ」のroleを変更出来る。

(2) マスターメンテナンス

よくある管理機能

※ これ以外は、必要に応じて実装する。

Authentication

ページへのアクセスを、roleによって制限するようになっています。

メソッドとして、application_helper.rbに
`allow_to_admin_user_only`

と

`allow_to_shop_clerk_only`

とを準備します。

Application_helper

```
#!/ruby
# coding: utf-8
module ApplicationHelper
  def menu_link_to( item )
    (省略)
  end

  def allow_to_admin_user_only
    if !current_user || current_user.role.ename != "admin"
      redirect_to :user_root
      return
    end
  end

  def allow_to_shop_clerks_only
    if !current_user || current_user.role.ename != "staff"
      redirect_to :user_root
      return
    end
  end
endend
```

今日の修正はここまで

allow_to_admin_user_onlyメソッドや、
allow_to_shop_clerks_onlyメソッドなどを書く。

次のステップ

Capbaraによる、画面入力のテストを書き、本日作成した`allow_to_admin_user_only`や、`allow_to_shop_clerks_only`などが機能するか、テストを通していきます。

ボリュームが大きくなったため、一旦ここまでに区切ります。

今日のまとめ

- ViewとControllerとの関係動作についてまとめた
- 最初に、Controller経由で特定の画面を呼び出し、そこでユーザに値の入力などを求め、controllerに値を返して、保存したり、計算処理するという一連の流れを辿った。

今日のレポート課題

レポート課題はありません。

今日の欠席課題

Roleの異なる3名のユーザを登録して下さい。

Roleによって、メニューが異なることを確認して下さい。

Rspec/Capybara入門

来週の教材は、以下のサイトを参照します。

[http://www.oiax.jp/rails/
rspec_capybara_primer.html](http://www.oiax.jp/rails/rspec_capybara_primer.html)

テストには強力なツールです。事前に目を通しておいてもらおうと良いと思います。