

# WEB+DBシステム(応用編)



第4回(2016年10月13日)

ログイン認証・ユーザ管理

# 今日の目標

---

- 前期の復習も含めて、ログインユーザの管理を組み込む。

# 「ユーザ」の設計

---

サイトの性質を考える。

どんな役割の人が使用するか。

- ※ 管理人: 管理人もWEBからデータを操作できる
- ※ スタッフ: サイト運営側ユーザ: 商品データのメンテナンスや、商品の発送処理などを行う。
- ※ ゲスト: そのサイトを通じて情報を得たり、購入してくれる人。

システムからは、全てをユーザとして扱う。

# ショッピングサイトの機能

---

サイトで何が出来るか。

サイトを運営する側の立場から・・・

- 商品を登録する。
- 顧客リストを確認する。
- 顧客の購入品リストを見る。
- 購入済みの商品を発送する。
- などなど

# ショッピングサイトの機能

---

顧客の立場から、どんなことが出来て欲しいか。

- 商品の写真や価格などを画面で見る。
- 名称で商品を検索する。
- 自分を登録する。
- 商品を選択して、カートに入れる。
- などなど

# ショッピングサイトのテーブル設計

---

マスターテーブル – 比較的持続的に内容が変わらないもの

- ◇ Merchandises – 商品
- ◇ Staffs – ログイン認証用
- ◇ Customers – ユーザ認証用

Activityの記録(Transaction) – 販売記録

- ◇ Sales records – 購入(販売)管理
- ◇ Shopping Carts – 購入用/販売プロモーション用

Log – システムの稼働記録

- ◇ Access Log – log in/outの記録,

などなど

# 今日の導入

---

1. サイトのWelcome画面を作る。  
ここから、スタッフ用のログイン画面などに分岐させる。  
サイトのお勧め商品や、お知らせなどを表示する。
2. システムのTOP画面に、認証要求を設定することにより、自動的にログイン画面が開かれることを確認する。
3. ログインしなくても商品を「ショッピング・カート」に入れられるようにし、「支払い」の段階で「会員登録=sing up」させる。

## 準備 (Deviseの確認)

---

必要なgemがインストールされているか確認します。

`gem list devise`

と入力します。もし、Installされていないかったなら、Gemfileに

`gem 'devise'`

`gem 'bcrypt', '~>3.1.11'`

を追加し、

`sudo bundle install`

コマンドを実行します。



# Gemfileの修正

Gemfileは、プロジェクトフォルダの直下にあります。



The image shows a file explorer on the left and a code editor on the right. The file explorer displays a directory structure with folders like bin, config, db, lib, log, public, test, tmp, and vendor, and files like config.ru, Gemfile, Gemfile.lock, Rakefile, and README.md. The Gemfile file is highlighted with a red box. The code editor shows the contents of the Gemfile, with lines 18 and 19 highlighted by a red box:

```
5 # Use sqlite3 as the database for Active Record
6 gem 'sqlite3'
7 # Use Puma as the app server
8 gem 'puma', '~> 3.0'
9 # Use SCSS for stylesheets
10 gem 'sass-rails', '~> 5.0'
11 # Use Uglifier as compressor for JavaScript assets
12 gem 'uglifier', '>= 1.3.0'
13 # Use CoffeeScript for .coffee assets and views
14 gem 'coffee-rails', '~> 4.2'
15 # See https://github.com/rails/execjs#readme for more supported runtimes
16 gem 'therubyracer', platforms: :ruby
17
18 gem 'devise'
19 gem 'bcrypt', '~>3.1.11'
20
21 # Use jquery as the JavaScript library
22 gem 'jquery-rails'
23 # Turbolinks makes navigating your web application faster. Read more: htt
```

# sudoとは

---

`bundle install`

ではなく、

`sudo bundle install`

を実行する理由

Suは、Super User(つまりroot/システム管理者)で

Doは実行、システム管理者モードで実行しろ、という命令です。Bcryptが「暗号処理」を含むため、一部su権限を要求している(らしい)ので、bcryptのinstallがsudoでないと失敗するため、です。

# Bundle installの実行

---

Bundle installは、console上で、プロジェクトフォルダに移動して実行します。

私の場合は、vegetable-marketフォルダです。各自、自分が決めたプロジェクトのフォルダで行って下さい。

```
[WebDB@cisnote vegetable-market]$ sudo bundle install
Don't run Bundler as root. Bundler can ask for sudo if it is needed, and
installing your bundle as root will break this application for all non-root
users on this machine.
Fetching gem metadata from https://rubygems.org/
Fetching version metadata from https://rubygems.org/
Fetching dependency metadata from https://rubygems.org/
Resolving dependencies....
Using rake 11.3.0
Using concurrent-ruby 1.0.2
Using i18n 0.7.0
Using minitest 5.9.1

Using web-console 3.3.1
Using rails 5.0.0.1
Using sass-rails 5.0.6
Installing devise 4.2.0
Bundle complete! 17 Gemfile dependencies, 71 gems now installed.
Use `bundle show [gemname]` to see where a bundled gem is installed.
[WebDB@cisnote vegetable-market]$
```

# 準備（バックアップの作成）

---

Gitのデータベースに、今の時点のファイルをバックアップしておき、問題が起きた時に戻せるようにしておきます。（第2回の資料参照）

わからない場合は、スキップしても構いません。

gitで、checkoutして元に戻した時は、

```
rake db:migrate:status
```

の入力で、状況を確認し、何回rollbackすれば良いか考え、

```
rake db:rollback
```

を必要な回数行い、データベースの復元も行って下さい。

# Gitの初期化と最初の保存

---

```
git init
```

```
git add .
```

```
git commit -m '初期状態'
```

で、最初の状態を保存できます。

特に、新しくバックアップする対象が増えていない場合は

```
git commit -a -m '保存内容のメモ'
```

で保存できます。

# Gitによるバックアップの作成

---

## 第2回スライド参照

```
[root@cisnote ecocar]# git init
Initialized empty Git repository in /home/rails3work/ecocar/.git/
[root@cisnote ecocar]# git var GIT_COMMITTER_IDENT
Ikuo Kobayashi <ikuo.kobayashi.t9@k.hosei.ac.jp> 1350446699 +0900
[root@cisnote ecocar]# git add .
[root@cisnote ecocar]# git commit -m '初期状態'
[master (root-commit) df3fcbc] 初期状態
71 files changed, 2342 insertions(+)
create mode 100644 .gitignore
create mode 100644 .project
create mode 100644 Gemfile
```

# Gitの小まとめ(第2回再掲)

---

gitの初期化 (使い始め)

```
git init [プロジェクトのルートで行う。]
```

コミットの登録 (一度だけ、または、システムで一度)

```
git config --global user.name "私の名前"
```

```
git config --global user.email xxx@xxx.xx
```

保存対象の追加

```
git add フォルダ名(またはファイル名)
```

これまでの保存対象の更新を追加

```
git add -u
```

保存(コミット)

```
git commit -m "バックアップの名称"
```

# Git (の課題)についての補足

---

バージョンが分岐したり、「改行」や「空白」で段落などの見た目を調べてしまったりした場合、意図せずに「競合する改修」を加えたことになる場合があります。(また、今回は「指定内容」に若干無理がありました。)

こうした場合は、編集しないでgitの操作だけではバージョンのmergeや、「過去の消去」などを行うことが困難になる、となってしまう、「レポート課題」で指定した「gitの操作だけで」行うのは無理、ということになります。→編集作業は避けて通れない、という前提で課題を提出して下さい。



# ファイルの探し方

---

Railsで作成されたファイルは、大きく

**app**          model/view/controllerなど  
アプリケーションプログラムが存在

**config** 言語環境などの設定ファイル

**db**            sqlite3のデータベースファイル

**doc**        開発中アプリのAPIドキュメント用

などのフォルダがあり、「設定」ならconfig, プログラムならapp  
の中を探して下さい。

※ APIとは、Application Programming Interfaceの略です。

## ファイルの探し方(2)

---

第1回の授業スライドを参照して下さい。

Aptanaを起動した際に表示されるWorkspaceの場所にファイルがあります。

[/home/WebDB/workspace/](#)

などという名称で作成され、この下に「プロジェクト」単位でフォルダが作成されます。

それぞれのプロジェクトをルートとすると、一つ前のページのフォルダが置かれています。

# 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。

```
[WebDB@cisnote vegetable-market]$ rails generate devise:install
Running via Spring preloader in process 3691
  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
  e
  is an example of default_url_options appropriate for a development environm
  ent
  in config/environments/development.rb:

      config.action_mailer.default_url_options = { host: 'localhost', port: 300
  0 }

  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. You can copy Devise views (for customization) to your app by running:

      rails g devise:views

=====
[WebDB@cisnote vegetable-market]$
```

# 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', port: 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. You can copy Devise views (for customization) to your app by running:

```
rails g devise:views
```

```
=====
=====
```

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<%= yeild %>の前に書き加えます。



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

# Welcome画面の生成(前期教材参照)

---

以下のコマンドを入力します。

`rails generate controller welcome index`

```
[WebDB@cisnote vegetable-market]$ rails generate controller welcome index
Running via Spring preloader in process 4165
  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/controllers/welcome_controller_test.rb
  invoke helper
  create  app/helpers/welcome_helper.rb
  invoke test_unit
  invoke assets
  invoke coffee
  create  app/assets/javascripts/welcome.coffee
  invoke scss
  create  app/assets/stylesheets/welcome.scss
[WebDB@cisnote vegetable-market]$
```

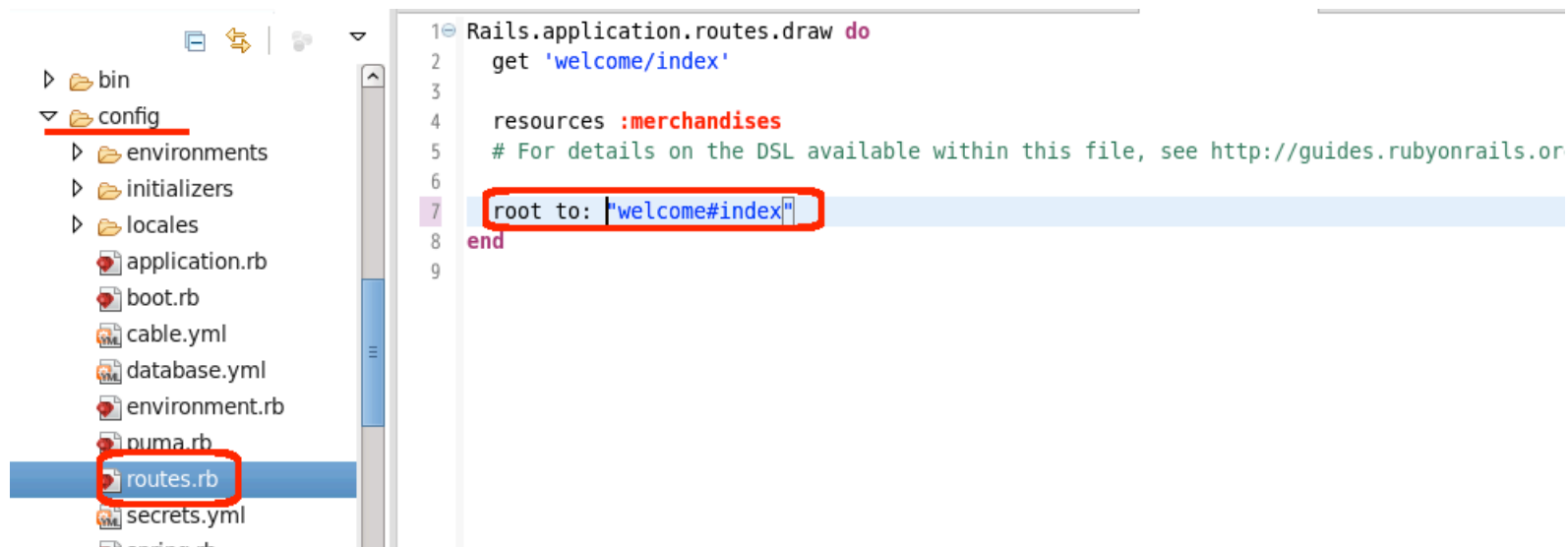
# Welcome画面の登録

作成したWelcomeのindex画面を登録します。

`config/routes.rb`を編集します。

`root to: 'welcome#index'`

を一行追加。



```
1 Rails.application.routes.draw do
2   get 'welcome/index'
3
4   resources :merchandises
5   # For details on the DSL available within this file, see http://guides.rubyonrails.or
6
7   root to: 'welcome#index'
8 end
9
```

# 「ユーザ」の設計(再掲)

---

サイトの性質を考える。

どんな役割の人が使用するか。

- ※ 管理人: 管理人もWEBからデータを操作できる
- ※ スタッフ: サイト運営側ユーザ: 商品データのメンテナンスや、商品の発送処理などを行う。
- ※ ゲスト: そのサイトを通じて情報を得たり、購入してくれる人。

システムからは、全てをユーザとして扱う。



# ユーザモデルの設定

---

権限がなく、ただ閲覧するだけの「ゲスト」は、ここでは「ユーザ」のテーブルには登録しない。

「買い物」などを管理する場合（何らかのアクセス記録をシステムに残す場合）には、「一般ゲスト」として登録する。（ログイン認証の対象とする。）

同じ画面で操作させるため、

1: 管理人、 2: スタッフ、 3: ゲスト

などの role(役割)コードをユーザモデルに与える。

# マスタとは何か

---

「役割」 role などは、必要に応じて拡張される場合があります。

このため、こうした項目も「一般データ」などと同様に、システムに「マスタ」として登録します。

今回の「ユーザ」は、この考え方でいけば「マスタ」としてシステムに持たせるべき情報になります。

今回は、「マスタ」と「トランザクション」は分離しないものとして作って行きます。が、スライド中に「マスタ」という言葉を使うことがあるので、以下に解説します。

# トランザクションとマスタ

---

データベースに記録される情報を、2種類に分けて考える場合があります。

(1) 日常的にアクセスがあって、更新される情報

- ・ メッセージの書き込み
- ・ 商品の注文
- ・ 投票
- ・ 売上
- ・ テストの成績など

(2) 更新頻度が低く、一旦決めておくと数年間変更がなかったりする情報

- ・ 商品名や価格
- ・ ユーザ情報
- ・ 支店情報
- ・ 分類コード

※ スーパーマーケットの野菜の価格などは、「マスタ」としては扱わない性質になります。

# マスタ・メンテナンス

---

頻繁にアクセスされる項目は、それ専用の画面を用意し、TOP画面などから簡単にリンクを辿れるようにします。

たまにしか更新されないマスタデータ群は、「マスタ・メンテナンス」というメニューを用意して、そこからアクセスするように、「トランザクション」管理と分離するのが、一般的な業務アプリケーションの作り方になります。

Scaffoldされる画面が概ね該当します。

# 「vegetable-market」でのuser

---

システム全体の管理者としてadminを設定します。

一部のマスタは、admin権限でのみ操作できるようにします。

購入を管理するスタッフだけが、商品データを登録できるようにします。会計記録などは、admin(システム管理者)や顧客には見られないようにしたい。

一般ユーザは、「商品購入」ができるようにします。

(商品閲覧画面は、ログインしなくてもできるように作りますので、roleのインストールができていなくても、作れるようにします。)

# Roleマスタの導入

---

ユーザ(User)テーブルはログイン認証用として作成されます。

ここで、ユーザテーブルに、role\_idを追加し、ログイン認証の対象となるユーザに、「役割」を追加します。

1: 管理人、 2: スタッフ、 3: ゲスト

などの role(役割)コードをユーザモデルに与えます。

昨年度までは、role\_idでユーザを切り分けていましたが、今年度はenameを使うこととします。

# Roleの生成

---

Scaffoldingによって、roleテーブル(マスタ)を生成します。

識別子としても使うename(English name)と、表示のわかりやすさのためのjname(Japanese name)の両方を持たせます。

```
rails generate scaffold role ename:string  
jname:string
```

```
[WebDB@cisnote vegetable-market]$ rails generate scaffold role ename:string jname:string
Running via Spring preloader in process 4246
  invoke  active_record
  create  db/migrate/20161012091851_create_roles.rb
  create  app/models/role.rb
  invoke  test_unit
  create  test/models/role_test.rb
  create  test/fixtures/roles.yml
  invoke  resource_route
   route  resources :roles
  invoke  scaffold_controller
  create  app/controllers/roles_controller.rb
  invoke  erb
  create  app/views/roles
  create  app/views/roles/index.html.erb
  create  app/views/roles/edit.html.erb
  create  app/views/roles/show.html.erb
  create  app/views/roles/new.html.erb
  create  app/views/roles/_form.html.erb
  invoke  test_unit
  create  test/controllers/roles_controller_test.rb
  invoke  helper
  create  app/helpers/roles_helper.rb
  invoke  test_unit
  invoke  jbuilder
  create  app/views/roles/index.json.jbuilder
  create  app/views/roles/show.json.jbuilder
  create  app/views/roles/_role.json.jbuilder
  invoke  assets
  invoke  coffee
  create  app/assets/javascripts/roles.coffee
  invoke  scss
  create  app/assets/stylesheets/roles.scss
  invoke  scss
  identical  app/assets/stylesheets/scaffolds.scss
[WebDB@cisnote vegetable-market]$
```



# Migrationの実行

---

rolesテーブルを作成しておきます。

rake db:migrate

```
[WebDB@cisnote vegetable-market]$ rake db:migrate
== 20161012091851 CreateRoles: migrating =====
-- create_table(:roles)
   -> 0.0133s
== 20161012091851 CreateRoles: migrated (0.0135s) =====
[WebDB@cisnote vegetable-market]$ █
```

# 役割マスターの登録

```
rails s -b 127.0.0.1
```

で、サーバを起動し、役割マスターデータを登録します。

```
http://127.0.0.1:3000/roles
```

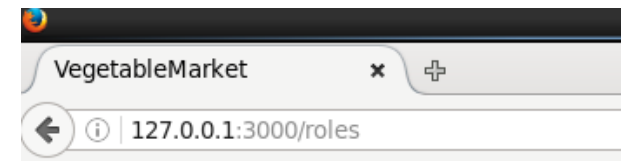
から、rolesを新規登録し、

```
ename:admin      jname:管理人
```

```
ename:staff      jname:スタッフ
```

```
ename:guest      jname:一般客
```

の三種類を登録します。



## Roles

Ename	Jname			
admin	管理人	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
staff	スタッフ	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
guest	一般客	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>

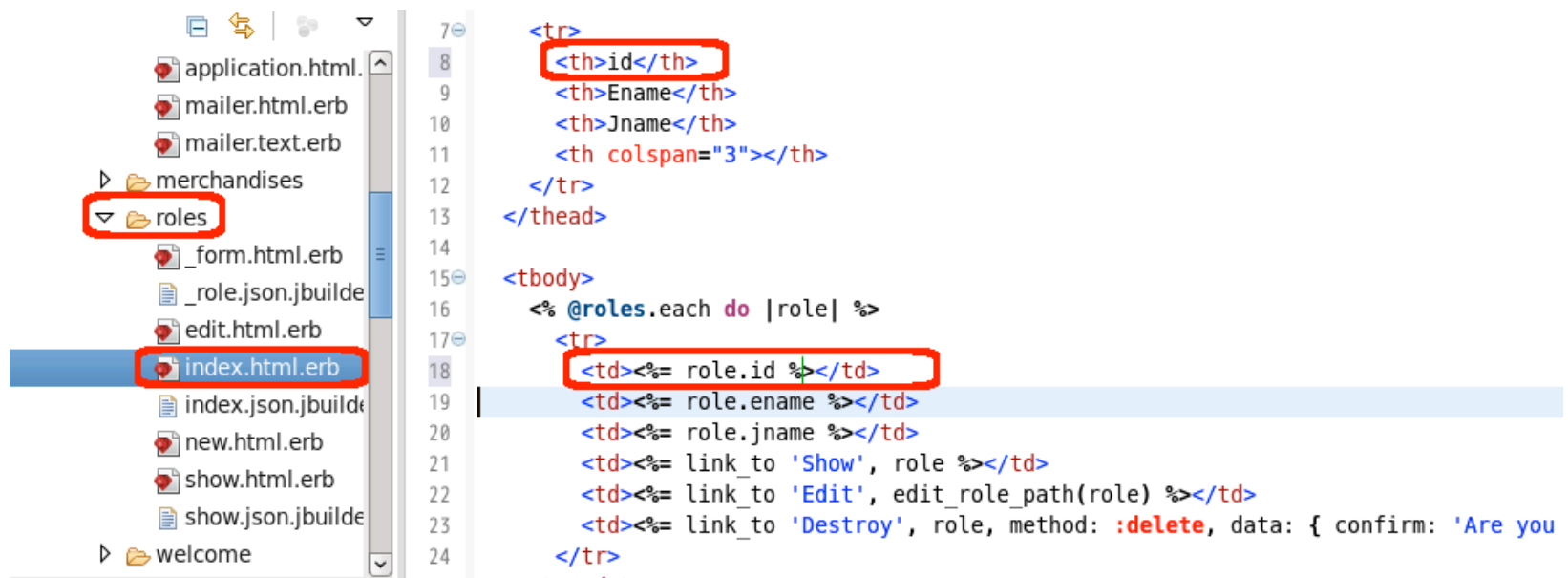
[New Role](#)

# Role表示にIDを追加

app/views/roles/index.html.erb

に、idの桁を追加します。

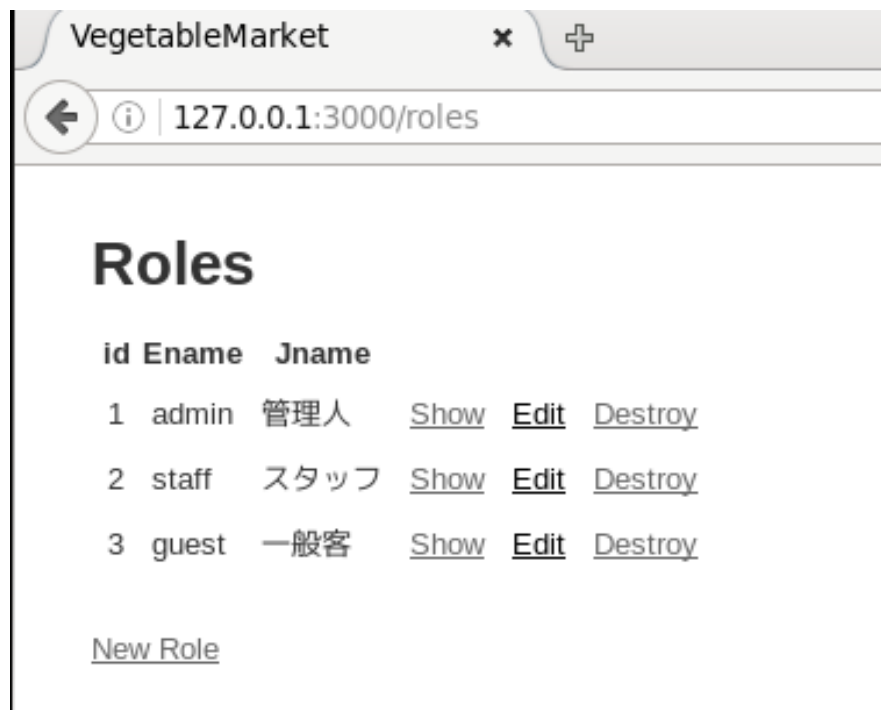
これは、データベースのデフォルトを設定するためです。



```
7 <tr>
8   <th>id</th>
9   <th>Ename</th>
10  <th>Jname</th>
11  <th colspan="3"></th>
12 </tr>
13 </thead>
14
15 <tbody>
16   <% @roles.each do |role| %>
17     <tr>
18       <td><%= role.id %></td>
19       <td><%= role.ename %></td>
20       <td><%= role.jname %></td>
21       <td><%= link_to 'Show', role %></td>
22       <td><%= link_to 'Edit', edit_role_path(role) %></td>
23       <td><%= link_to 'Destroy', role, method: :delete, data: { confirm: 'Are you
24     </tr>
25   </tbody>
26 </table>
27 </div>
28 </div>
29 </div>
30 </div>
31 </div>
32 </div>
33 </div>
34 </div>
35 </div>
36 </div>
37 </div>
38 </div>
39 </div>
40 </div>
41 </div>
42 </div>
43 </div>
44 </div>
45 </div>
46 </div>
47 </div>
48 </div>
49 </div>
50 </div>
51 </div>
52 </div>
53 </div>
54 </div>
55 </div>
56 </div>
57 </div>
58 </div>
59 </div>
60 </div>
61 </div>
62 </div>
63 </div>
64 </div>
65 </div>
66 </div>
67 </div>
68 </div>
69 </div>
70 </div>
71 </div>
72 </div>
73 </div>
74 </div>
75 </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 </div>
81 </div>
82 </div>
83 </div>
84 </div>
85 </div>
86 </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92 </div>
93 </div>
94 </div>
95 </div>
96 </div>
97 </div>
98 </div>
99 </div>
100 </div>
```

# ID表示のあるrolesのindex画面

IDの表示が追加されると、以下のような画面になります。



The screenshot shows a web browser window with the title 'VegetableMarket'. The address bar displays '127.0.0.1:3000/roles'. The main content area is titled 'Roles' and contains a table with three columns: 'id', 'Ename', and 'Jname'. Each row in the table has three links: 'Show', 'Edit', and 'Destroy'. Below the table is a link labeled 'New Role'.

id	Ename	Jname			
1	admin	管理人	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
2	staff	スタッフ	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
3	guest	一般客	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>

[New Role](#)

# ここから先の手順に注意！

---

こんな話が・・・

>> Railsは色々ジェネレータでコードを生成してくれてすごいなあと思うけど、始発で乗り損なったり、乗る電車や順番を間違えると結構戻るのが大変というか、結局てくてく隣駅まで歩くという、そんな感じが意外に多いような気がする。

<http://d.hatena.ne.jp/yarb/20100803/p1>

Userテーブルにhandleなどを登録するために、先にuserをscaffoldし、次にdevise生成します。

# 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 scaffold user email:string  
role_id:integer handle:string
```

```
[WebDB@cisnote vegetable-market]$ rails generate scaffold user email:string role_  
_id:integer handle:string  
Running via Spring preloader in process 4441  
  invoke  active_record  
  create  db/migrate/20161012094205_create_users.rb  
  create  app/models/user.rb  
  invoke  test_unit  
  create  test/models/user_test.rb  
  create  test/fixtures/users.yml  
  invoke  resource_route  
   route  resources :users  
  invoke  scaffold_controller  
  create  app/controllers/users_controller.rb  
  invoke  erb  
  create  app/views/users  
  create  app/views/users/index.html.erb  
  create  app/views/users/edit.html.erb
```

# Userをdeviseで認証に用いる

---

ログインするユーザとして、deviseを生成します。

`rails generate devise user`

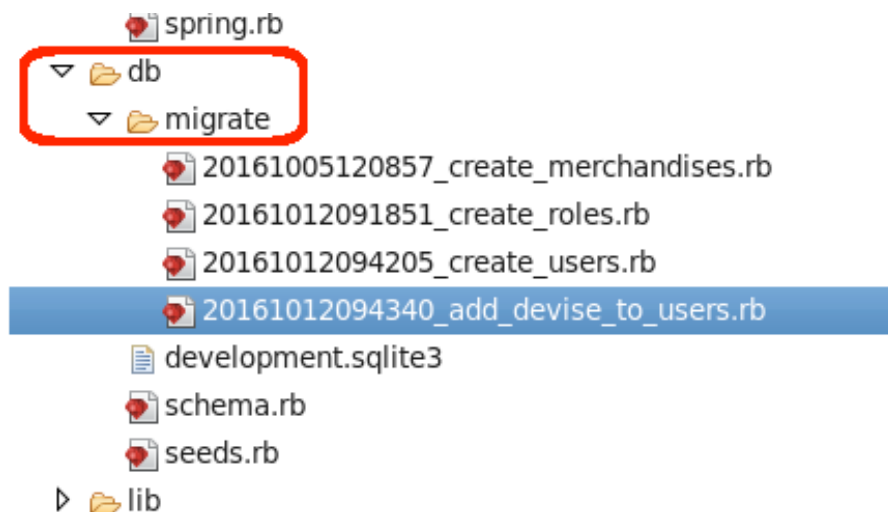
```
[WebDB@cisnote vegetable-market]$ rails generate devise user
Running via Spring preloader in process 4450
  invoke  active_record
  create  db/migrate/20161012094340_add_devise_to_users.rb
  insert  app/models/user.rb
  route  devise_for :users
[WebDB@cisnote vegetable-market]$
```

# deviseのmigrationを探す

ユーザが自分を登録した際に、自己登録が可能なユーザは全て「一般ユーザ」のroleを持つように、デフォルトを設定します。

db/migrate/2016....add\_devise\_to\_users.rb

というファイルを探します。





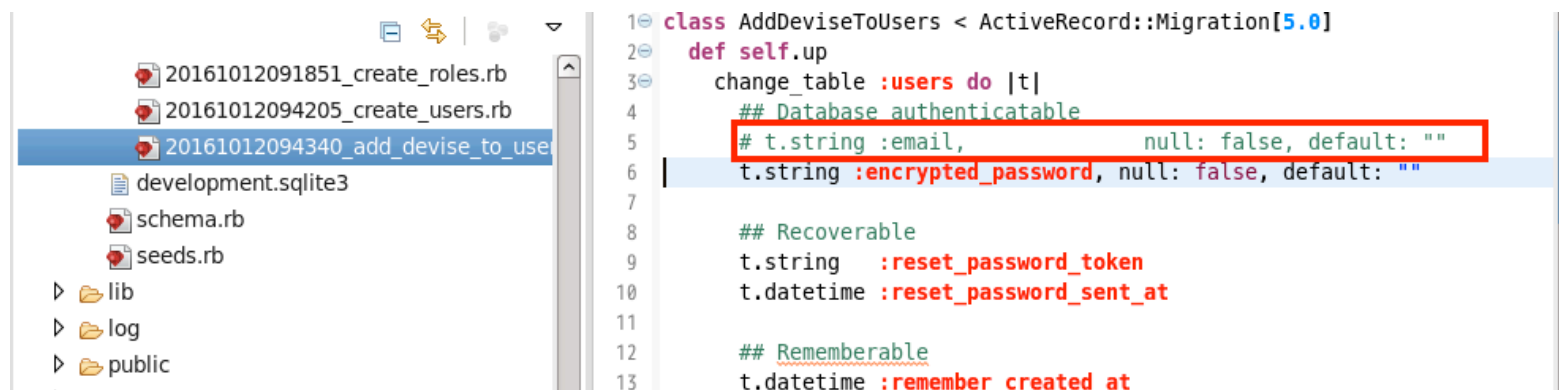
# deviseのmigrationを修正

Userテーブルでscaffoldした際に生成した、emailのフィールドを、deviseでも生成しています。

ここを、コメントアウトします。

5行目付近の

```
# t.string :email, null: false, default: ""
```



```
1 class AddDeviseToUsers < ActiveRecord::Migration[5.0]
2   def self.up
3     change_table :users do |t|
4       ## Database authenticatable
5       # t.string :email, null: false, default: ""
6       t.string :encrypted_password, null: false, default: ""
7
8       ## Recoverable
9       t.string :reset_password_token
10      t.datetime :reset_password_sent_at
11
12      ## Rememberable
13      t.datetime :remember_created_at
```

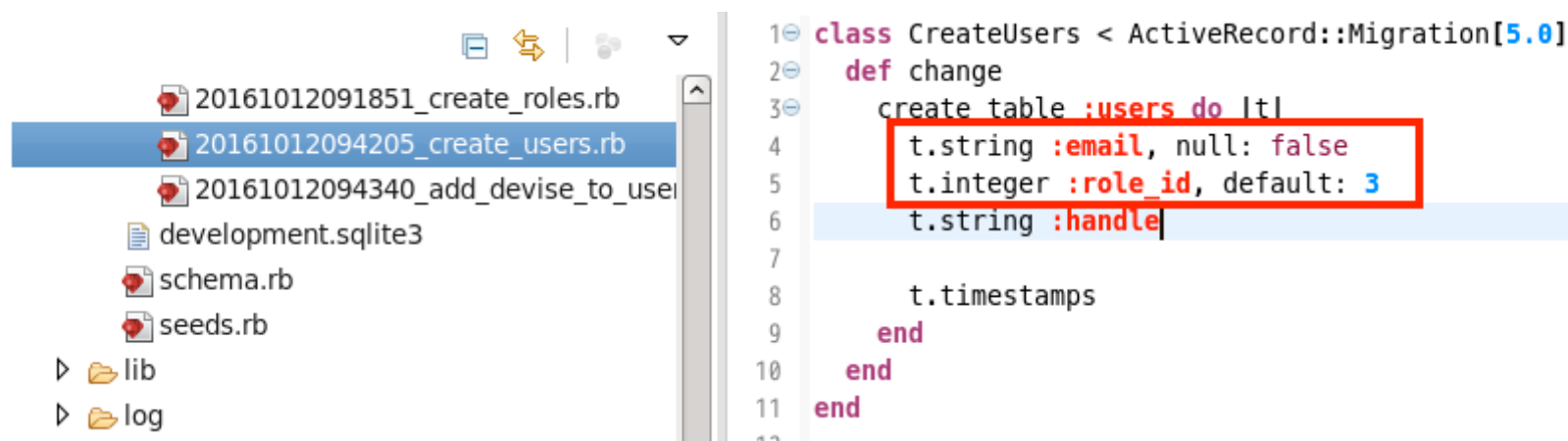
# Userのmigrationのroleを修正

新たにユーザが追加された際に、デフォルトで「一般」になるように、以下のように修正します。

ここで、私は3としていますが、rolesで「一般」として表示されたID番号が3だったので3にしています。

Emailには, `null: false`を追記します。

皆さんの画面では、違う数字になる可能性もあります。



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like `20161012091851_create_roles.rb`, `20161012094205_create_users.rb`, `20161012094340_add_devise_to_user.rb`, `development.sqlite3`, `schema.rb`, and `seeds.rb`. The code editor shows the content of `20161012094205_create_users.rb`, which is a migration class `CreateUsers` inheriting from `ActiveRecord::Migration[5.0]`. The `change` method contains a `create table :users do |t|` block. The `t.string :email, null: false` and `t.integer :role_id, default: 3` lines are highlighted with a red box.

```
1 class CreateUsers < ActiveRecord::Migration[5.0]
2   def change
3     create table :users do |t|
4       t.string :email, null: false
5       t.integer :role_id, default: 3
6       t.string :handle
7
8       t.timestamps
9     end
10  end
11 end
```

# Migrationの実行

---

usersテーブルを作成しておきます。

(deviseのmigrationで、emailをコメントアウトしないと、エラーになります。)

`rake db:migrate`

```
[WebDB@cisnote vegetable-market]$ rake db:migrate
== 20161012094205 CreateUsers: migrating =====
-- create_table(:users)
   -> 0.0111s
== 20161012094205 CreateUsers: migrated (0.0112s) =====

== 20161012094340 AddDeviseToUsers: migrating =====
-- change_table(:users)
   -> 0.0146s
-- add_index(:users, :email, {:unique=>true})
   -> 0.0030s
-- add_index(:users, :reset_password_token, {:unique=>true})
   -> 0.0027s
== 20161012094340 AddDeviseToUsers: migrated (0.0206s) =====

[WebDB@cisnote vegetable-market]$ █
```

# これから行うこと

---

- (1) rolesのトップ画面でユーザ認証を要求します。
- (2) ユーザとして、adminを登録します。
- (3) userテーブルを編集して、adminユーザの権限をadmin(管理人)に設定します。
- (4) rolesマスタの編集権限を、adminだけに限定します。

# ホーム画面の設定

---

merchandisesのindex画面を、このショッピングサイト・システムのホームとして設定・登録してみます。  
(今後、段階的に書き改めます。)

config/routes.rbに、以下の記述を追加します。

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

resources :merchandisesの前に書きます。

# Routes.rbの修正

私の場合には、以下のようになります。

```
routes.rb 20161012094340_ 20161012094205_ »_3
1 Rails.application.routes.draw do
2   devise_for :users
3   resources :users
4   resources :roles
5   get 'welcome/index'
6
7   get 'merchandises', :to => 'merchandises#index', :as => :user_root
8   resources :merchandises
9   # For details on the DSL available within this file, see http://guides.
10
11   root "welcome#index"
12 end
13
```

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

- この行の記述が、ログイン後の画面（ユーザルート）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 (merchandises#index)が表示されるように、切り換えます。

App/controllers/welcome\_controller.rb

のindex メソッドに以下の内容を追加します。

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



# welcome\_controller.rb

---

```
routes.rb 20161012094205_ welcome_control ⌘
1 class WelcomeController < ApplicationController
2   def index
3     if current_user
4       redirect_to :user_root
5     return
6   end
7 end
8 end
9
```

## current\_userについて

---

このパラメータが設定されているときは、login済みだと判断できます。

Userをdeviseしたので、current\_userとなります。  
authenticate\_user!も同様です。

もし、memberをdeviseしたなら、  
current\_memberというパラメータが生成され、認証の実行は

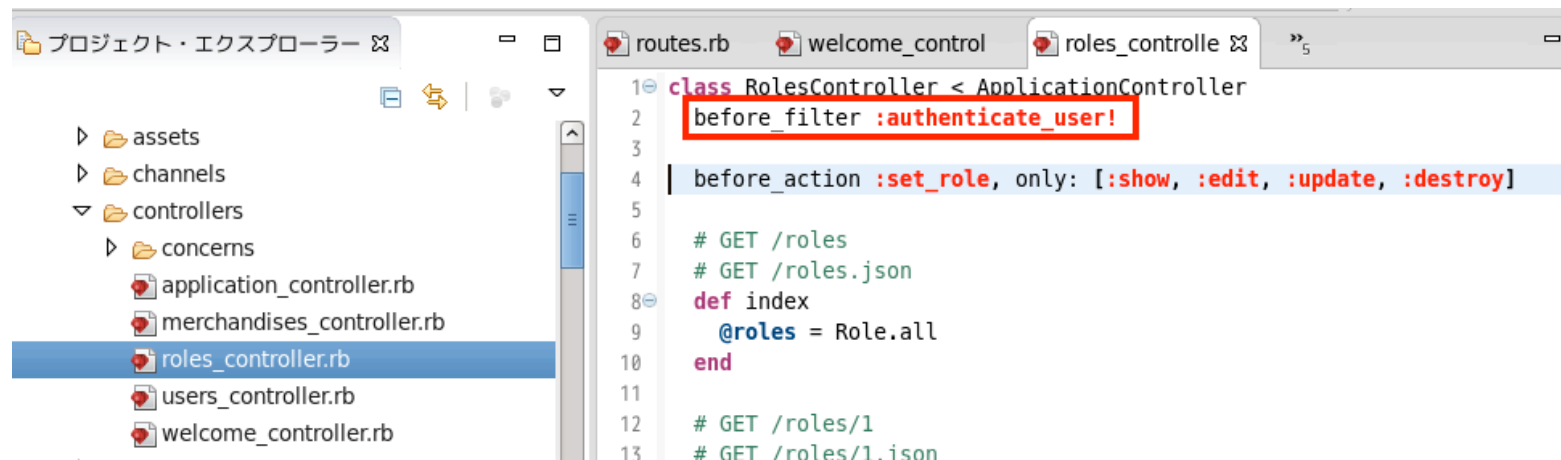
authenticate\_member!

になります。

# 認証の要求

認証を要求したいControllersに、認証の要求を書き加えます。(今回は、roles\_controller.rb)

`before_filter :authenticate_user!`



```
プロジェクト・エクスプローラー
├── assets
├── channels
├── controllers
│   ├── concerns
│   │   ├── application_controller.rb
│   │   ├── merchandises_controller.rb
│   │   └── roles_controller.rb
│   ├── users_controller.rb
│   └── welcome_controller.rb
├── routes.rb
├── welcome_control
└── roles_controlle
    1 class RolesController < ApplicationController
    2   before_filter :authenticate_user!
    3
    4   before_action :set_role, only: [:show, :edit, :update, :destroy]
    5
    6   # GET /roles
    7   # GET /roles.json
    8 def index
    9   @roles = Role.all
    10 end
    11
    12 # GET /roles/1
    13 # GET /roles/1.json
```

# Adminユーザの登録

<http://127.0.0.1:3000/roles>

を開こうとすると、認証画面に飛びます。

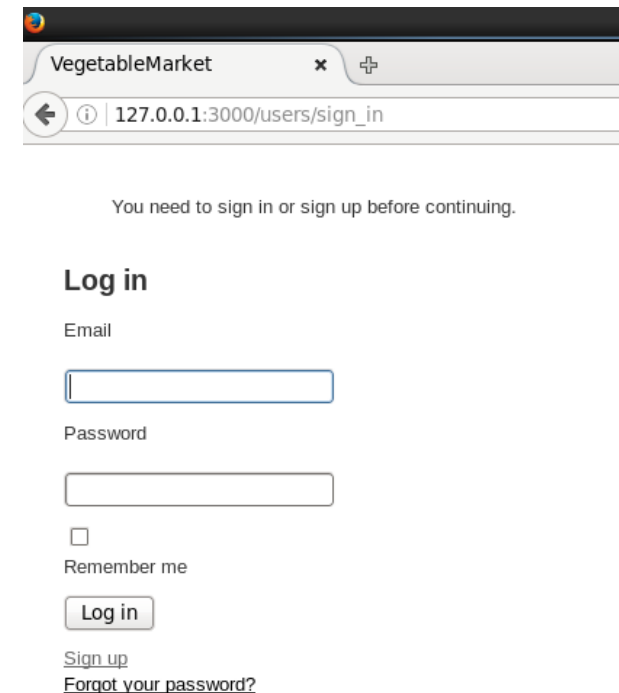
最初に、一般ユーザと同様の手順で

admin

を登録します。ログイン画面から、

sign\_upするとユーザ登録に

移ります。

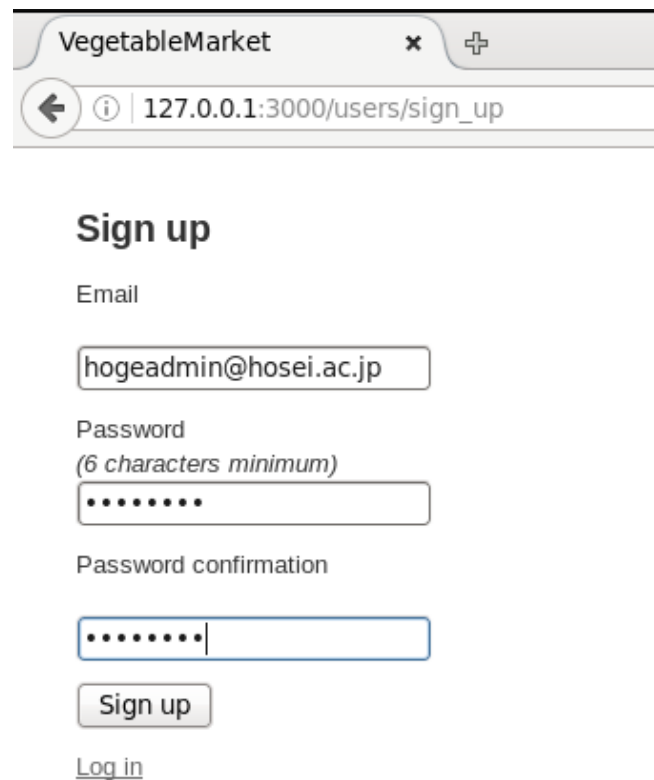


The screenshot shows a web browser window with the title 'VegetableMarket'. The address bar displays '127.0.0.1:3000/users/sign\_in'. The page content includes a message: 'You need to sign in or sign up before continuing.' Below this is a 'Log in' section with an 'Email' label and an input field, a 'Password' label and an input field, a 'Remember me' checkbox, and a 'Log in' button. At the bottom, there are links for 'Sign up' and 'Forgot your password?'.

# adminユーザを登録

---

adminとするユーザを、普通に登録します。  
この段階では、adminのHandleは出てきません。



The screenshot shows a web browser window with the title 'VegetableMarket'. The address bar displays '127.0.0.1:3000/users/sign\_up'. The page content includes a 'Sign up' heading, an 'Email' field with the value 'hogeadmin@hosei.ac.jp', a 'Password' field with a note '(6 characters minimum)' and a masked password '.....', a 'Password confirmation' field with a masked password '.....', a 'Sign up' button, and a 'Log in' link.

VegetableMarket x +

← ⓘ | 127.0.0.1:3000/users/sign\_up

**Sign up**

Email

hogeadmin@hosei.ac.jp

Password  
(6 characters minimum)

.....

Password confirmation

.....

Sign up

[Log in](#)

# Usersテーブルで編集します。

<http://127.0.0.1:3000/users>

を開くと、sign upしたユーザのhandleやroleを編集  
できます。

ここで、roleを1 (admin)に、handleを1に設定します。  
このユーザは、今後adminになります。

## Editing User

Email

Role

Handle

[Show](#) | [Back](#)

[←](#) | [i](#) | [127.0.0.1:3000/users](http://127.0.0.1:3000/users)

## Users

Email	Role	Handle	
hoge@hosei.ac.jp	3		<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>
hogeadmin@hosei.ac.jp	1		<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>

[New User](#)

# 今後の設計方針

---

Usersコントローラを修正し、

sign upしたユーザが、自分の「設定情報 (Handle)」を編集できるようにします。

Roleの変更は、adminだけができるようにします。

(一般ユーザが、勝手にroleを変更すると困る。)

現状では、一般ユーザが他のユーザのハンドル名も編集できます。これはなんとかしないとなりません。

# Model role.rbとuser.rbの修正

ここで、roleとusersの間に、relationを貼ります。

user.rbでは belongs\_to :role

role.rbでは has\_many :users

```
routes.rb index.html.erb roles_controlle role.rb user.rb ✖
1 class User < ApplicationRecord
2   belongs_to :role
3   # Include default devise modules. Others available are:
4   # :confirmable, :lockable, :timeoutable and :omniauthable
5   devise :database_authenticatable, :registerable,
6         :recoverable, :rememberable, :trackable, :validatable
7 end
8
```

```
routes.rb index.html.erb roles_controlle role.rb ✖
1 class Role < ApplicationRecord
2   has_many :users
3 end
4
```



# RoleをDrop Down Listにする

app/views/users/\_form.html.erb  
を編集します。

```
<%= f.number_field :role_id %>
```

を

```
<%= f.select :role_id, Role.all.collect{ |c| [c.jname, c.id]} %>
```

に修正します。

127.0.0.1:3000/users/2/edit

### Editing User

Email

Role

Handle

[Show](#) | [Back](#)

```
40 <div class="field">
41   <%= f.label :email %>
42   <%= f.text_field :email %>
43 </div>
44
45 <div class="field">
46   <%= f.label :role_id %>
47   <%= f.number_field :role_id %>
48 </div>
49
50 <div class="field">
51   <%= f.label :handle %>
52   <%= f.text_field :handle %>
53 </div>
```

```
15   <%= f.label :email %>
16   <%= f.text_field :email %>
17 </div>
18
19 <div class="field">
20   <%= f.label :role_id %>
21   <%= f.select :role_id, Role.all.collect{ |c| [c.jname, c.id]} %>
22 </div>
23
24 <div class="field">
25   <%= f.label :handle %>
```

# パスワードの編集は別!

---

パスワードは、

`http://127.0.0.1:3000/users/edit`  
を起動することで編集できます。

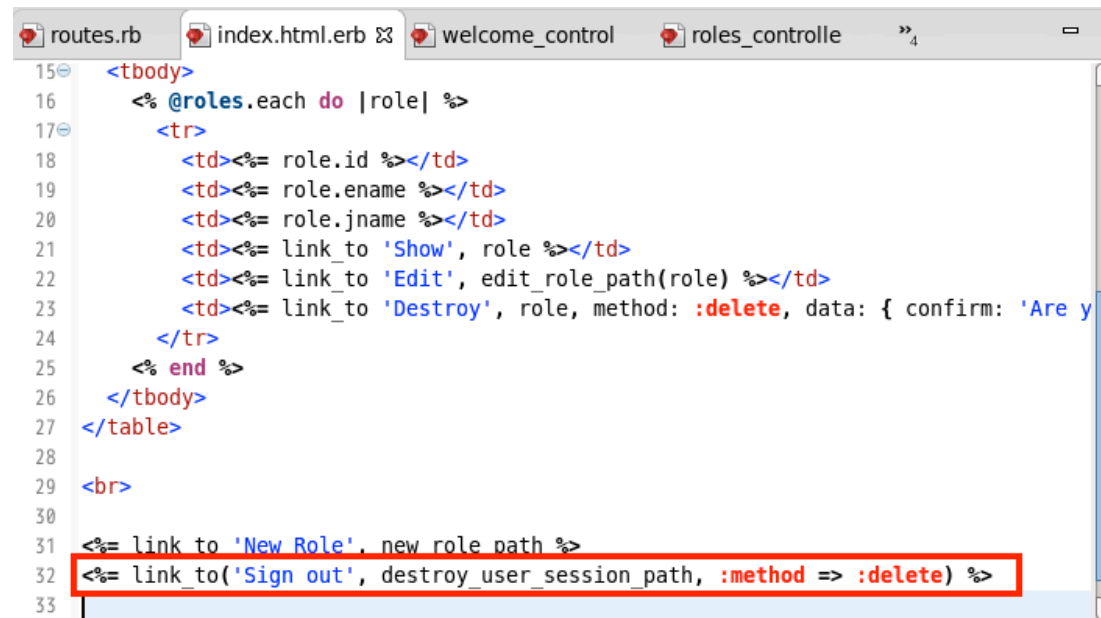
`app/views/devise/registrations/edit.html.erb`  
の編集が反映されています。

# Sign outの追加

現在、まだSign outの処理が入っていません。

画面はあまり調べていませんが、roles のindexにSign outを加えます。

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



The screenshot shows a code editor with several tabs: routes.rb, index.html.erb, welcome\_control, and roles\_controlle. The active tab is index.html.erb, which contains the following code:

```
15 <tbody>  
16   <%= @roles.each do |role| %>  
17     <tr>  
18       <td><%= role.id %></td>  
19       <td><%= role.ename %></td>  
20       <td><%= role.jname %></td>  
21       <td><%= link_to 'Show', role %></td>  
22       <td><%= link_to 'Edit', edit_role_path(role) %></td>  
23       <td><%= link_to 'Destroy', role, method: :delete, data: { confirm: 'Are y  
24     </tr>  
25   <%= end %>  
26 </tbody>  
27 </table>  
28  
29 <br>  
30  
31 <%= link_to 'New Role', new_role_path %>  
32 <%= link_to('Sign out', destroy_user_session_path, :method => :delete) %>  
33
```

The line 32, which is the newly added link\_to call, is highlighted with a red box in the original image.

# Sign outの追加

VegetableMarket x +

← ⓘ | 127.0.0.1:3000/roles

## Roles

id	Ename	Jname			
1	admin	管理人	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
2	staff	スタッフ	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>
3	guest	一般客	<a href="#">Show</a>	<a href="#">Edit</a>	<a href="#">Destroy</a>

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

# 動作確認

---

<http://127.0.0.1:3000/>

にアクセスします。Sign in していない場合には、Welcome画面が表示されることを確認して下さい。Sign in してこのURLにアクセスすると、`user_root(merchandises)`が開きます。

Login認証の要求を、<http://127.0.0.1:3000/roles>を開いて確認して下さい。

ここで、自己登録することで一般ユーザとしてログインできることを確認し、ログイン後に`user_root`画面が表示されることを確認して下さい。

また、`sign out`でログイン前の状態に戻ることを確認して下さい。この画面の確認ができて、今日の課題は完了です。

# うまくいったらバックアップ

---

今回、あれこれフォルダも生成しました。それらのすべてをバックアップ対象にするために、

```
git add -A
```

とコマンド入力します。

バックアップ対象を、保存します。名称は、'ログイン認証まで'とつけました。

```
git commit -m 'ログイン認証まで'
```

# 欠席課題

---

本日分の欠席課題は、ログイン認証の報告です。

- ・ ログイン認証を要求しているデータベースの(私のサンプルでは、roles)テーブルで、「認証要求を行っている」コントローラ記述の部分を、ソースコードを引用して記載し、その結果、一覧を表示しようとした際にログイン認証が要求されている部分のスクリーンショットと、ログイン後に一覧の表示ができている部分のスクリーンショットを添付して下さい。

この結果の報告で、出席扱いに切り換えます。

# 次回、TDDを導入します。

---

バックアップの方法を学び、システムの基本設計を考えてきました。

次は、テスト駆動開発の考え方を学び、システムのTOP画面からの開発を、テストを導入しながら検証していった進めるやり方で学びます。

今回のroleの追加を踏まえて進めて行きます。