Devise と twitter-bootstrap を用いた Rails アプリの雛形作成

ログイン認証が簡単にできる"devise"と"twitter-bootstrap"を用いて簡単に Web アプリの雛形を作ります。
ここでは例として Blog サンプルの雛形を作成します。

またこちらのサイト様を参考にさせてもらいました。
INOCCU VISION　http://inoccu.net/blog

記事ページ
　その1　http://inoccu.net/blog/2013/04/28/112225.html
　その2　http://inoccu.net/blog/2013/04/28/122144.html

手順 1　rails アプリの作成

端末上で rails new コマンドをうちアプリを作成します。

```
$ rails new blog-sample
$ cd blog-sample
```

アプリ名は blog-sample とすることにします。
アプリのディレクトリに移動することを忘れずに!

手順 2 Gemfile の編集

自分の環境では、therubyracer を install しないと scaffold ができないので
先に編集します。
Blog-sample/Gemfile のコメントアウトを外し、

```
#gem 'therubyracer' ,  :platforms => :ruby
```

を

```
gem 'therubyracer ' , :platforms => :ruby
```

にします。

手順３　scaffold によるアプリの雛形の作成

scaffold を実行し雛形を作ります。
今回は blog 作成ということで blog テーブルを作成します。
カラムは title, body, auther, posttime の4つとします。
（つくってから気づいたけど, title と body だけでいいかも）

```
$  rails g scaffold blog title:string body:text auther:string posttime:datetime
```

雛形ができたら忘れずに migrate する

```
$  rake db:migrate
```

ここで一回確認してみます。

```
$ rake routes
       blogs GET    /blogs(.:format)          blogs#index
             POST   /blogs(.:format)          blogs#create
    new_blog GET    /blogs/new(.:format)      blogs#new
   edit_blog GET    /blogs/:id/edit(.:format) blogs#edit
        blog GET    /blogs/:id(.:format)      blogs#show
             PUT    /blogs/:id(.:format)      blogs#update
             DELETE /blogs/:id(.:format)      blogs#destroy
```

rake routes でアクセス可能なパスを確認します。
ここでは localhost:3000/blogs にアクセスしてみます。

ちゃんと表示され雛形ができていることを確認できます。

手順４　twitter-bootstrap を対応させる。

まず Gemfile を以下のとおりに編集します。

```
group : assets do
  gem 'sass-rails' ,       ' ~> 3.2.3 '
  gem 'less-rails' ,
  gem 'coffee-rails' ,    ' ~> 3.2.1 '
  gem 'twitter-bootstrap-rails'
  gem 'therubyracer' , :platform => :ruby
  gem 'uglifier' , ' >= 1.0.3 '
end
```

そして忘れずに bundle install !

次に、bootstrap をアプリにインストールします

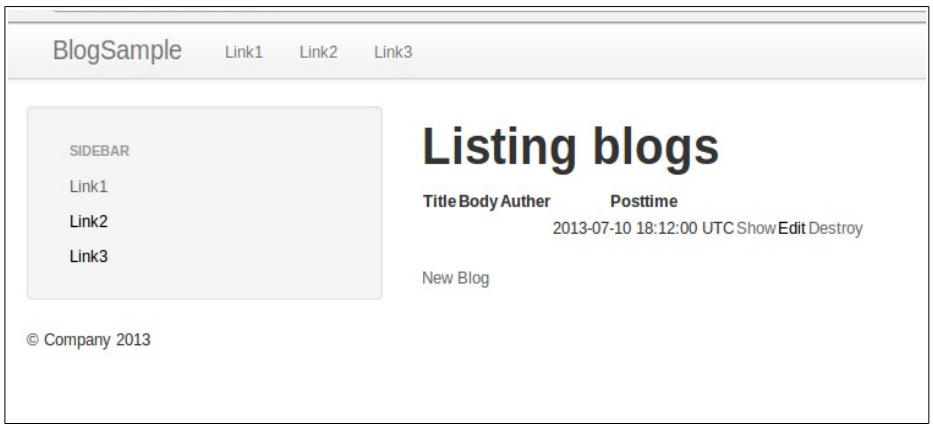
```
$ rails g bootstrap:install
```

そしてレイアウトファイルを作成します。

```
$ rails g bootstrap:layout application fluid
```

途中で上書きするか聞かれるので Y で許可します。
ちなみに fluid オプションをつけているので、レスポンシブ Web デザインになります。

この時点でもう一度確認します。



するとこんなかっこいい感じになります!

ついでにトップページを作っておきます。

```
$ rails g controller top index
```

これで index メソッドつきの top_controller.rb が作成されます。
さらに config/routes.rb を編集してトップページにします。

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

ここで、忘れずに public/index.html を削除します。(public ディレクトリの index.html が優先されてしまうため)

```
$  rm -f public/index.html
```

手順５　devise の導入

まず、Gemfile を編集し bundle install。

```
gem  'devise'
```

アプリに Devise をインストールします

```
$  rails g devise:install
```

インストール後、5 つの setup をしろと言われます。
1 つ目は config/environments/development.rb に以下を追記します。

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

2 つ目は先ほど root :to => top#index を記述したので大丈夫です。
3 つ目は注意、警告を表示するためのものですが、twitter-bootstrap で表示できるのでスルーで。
4 つ目は Heroku にデプロイする場合のみ行います
　config/application.rb
　(config/environments/production.rb かも?)にも以下を追記します。

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

5 つ目は devise の view をカスタマイズしたいときに行います。
以下のコマンドを行います。

```
$  rails g devise:views
```

手順６　Userモデルの追加

以下のコマンドでUserモデルを作成します。
（rails g model Userでないことに注意）

```
$  rails g devise User
```

ユーザー登録時のメール認証を行う場合は少し編集します。

app/models/user.rb　に:confirmableを追記します

```
 devise :database_authenticatable, :registerable, :confirmable,
        :recoverable, :rememberable, :trackable, :validatable
```

db/migrateにある(日時)_devise_create_users.rbのComfirmableの下４行の
コメントアウトを外して有効にします。

```
## Confirmable
t.string   :confirmation_token
t.datetime :confirmed_at
t.datetime :confirmation_sent_at
t.string   :unconfirmed_email # Only if using reconfirmable
```

そしてDBにusersテーブルを作成します

```
$  rake db:migrate
```

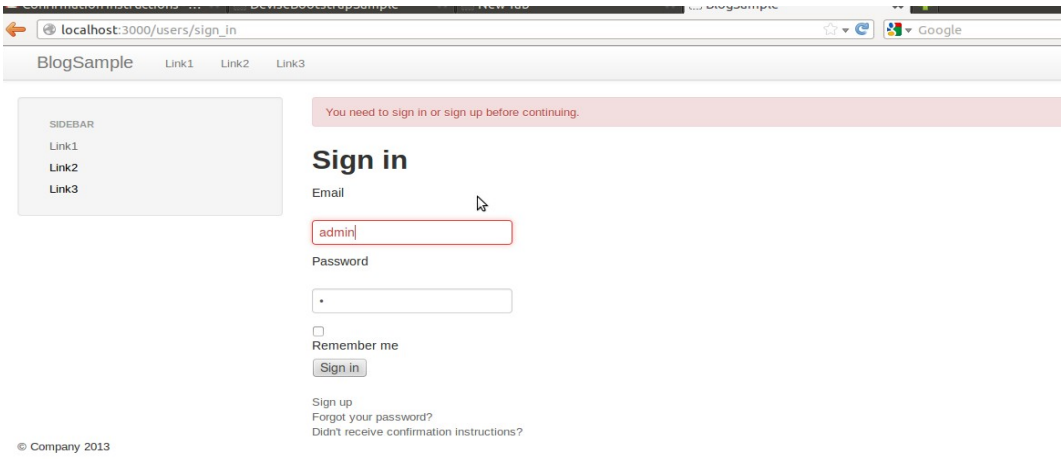
手順７　blogsコントローラに認証をかける

ログインしていない状態でblogsにアクセスしたときに、ログイン画面を出すようにします。

App/controllers/blogs_controller.rb フィルターをかける処理を追記します

```
class EventsController < ApplicationController
  before_filter :authenticate_user!  #ここを追記
  （省略）
end
```

試しにlocalhost:3000/blogsにアクセスすると・・・。



このようにsign_in画面へリダイレクトされ、ログインを促されます。

手順８　SMTPを設定する

ユーザー登録時にメールアドレス認証のため、Railsアプリからメールを送信する場合、
STMPを設定することが必要となります。

config/environment/development.rbに下記を追記します。

```
config.action_mailer.delivery_method = :smtp
config.action_mailer.smtp_settings = {
:address => 'smtp.gmail.com',
:port => 587,
:authentication => :plain,
:user_name => “メールアドレス”,
:password => “パスワード”,
}
```

この場合はGmailから送信することを前提としています。
なので送信元のアドレスをGmailで作成しておくといいと思います。

手順９　ユーザー登録してみる。

http://localhost:3000/users/sign_upにアクセスするとユーザ登録画面が表示されます。
（ログイン画面の下部にもリンクがあります。）

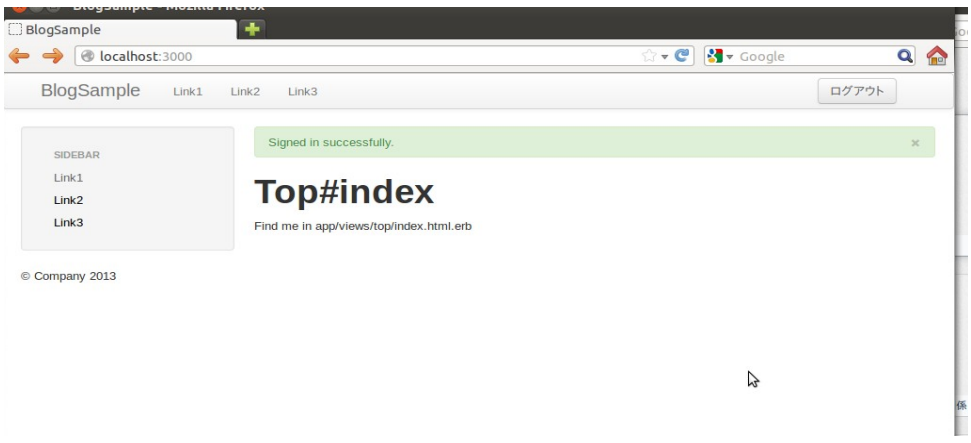
アドレスとパスワードを入力してSign upボタンをクリックすると下記の画面に遷移し、入力したアドレスに確認メールが届きます。
アドレスに記載された、Confirm my accountをクリックすると、ログインできちゃいます!!

ログイン、ログアウトのリンクをはっていないため、行います。

そこでログアウトは
　users/sign_out、
ログインは
　users/sign_in
というパスへのリンクをヘッダーに追加することにします。
また devise ではそれぞれへのパスが
　”new_user_session_path”
　”destroy_user_sission_path”
として、ログインしているかどうかを調べるメソッドが
　“user_signed_in?”
として用意されているのでこれを利用します。
app/views/layouts/application.rb に下記のコードを追記します。

```
<div class=”container-fluid nav-collapse”>
  <ul class=”nav”>
    <li><%= link_to ”Link1”, ”/events” %></li>
    <li><%= link_to ”Link2”, ”/events” %></li>
    <li><%= link_to ”Link2”, ”/events” %></li>
  </ul>
  <!-- ここからが追記部分-->
  <div class=”pull-right”>
    <% if user_signed_in? %>
    <%= link_to ”ログアウト”, new_user_session_path, :method => ”DELETE”, :class
=> ”btn” %>
    <% else %>
      <%= link_to ”ログイン”, destroy_user_session_path, :class => ”btn” %>
    <% end %>
  </div><!-- .pull-right -->
  <!-- ここまでが追記部分 -->
</div><!--/.nav-collapse -->
```

すると、



このようにログアウトボタンも作られいい感じになりました。