

WEB+DBシステム (入門編)



第9回(2016年6月9日)

画面の設計と画面のリンク

第7回のレポートについて

「抽象化」 -- プログラム中には、可能な限り
「個別」のデータに属する名称は使わない。

「課題」で、`:curry => 300`などとサンプルを書いたのは、
「ひっかけ」です。

配列を複数用意して、`index`で括り付ける。

→ ズれた時にデータエラーになる。

「属性」が三つ以上ある場合には、明らかに
Classが有利。

抽象化とカプセル化

【抽象化】

個別のデータを「特別」扱いしない →

データの「属性」ごとに処理を分けるのはOK

「固有の名前」を「プログラム」しない

【カプセル化】

「メニュー」という一般化したものの属性を、できるだけ不必要に外に見せない。

(値段も、名前も、「表示」だけに使う)

Rubyには型がない？

型宣言はありませんが、全ての変数はいずれかのクラスのインスタンスです。どのクラスのインスタンスかを意識しないと、バグの原因になりやすいです。

```
[irb(main):001:0> str = "ABC"  
=> "ABC"  
[irb(main):002:0> str.class  
=> String  
[irb(main):003:0> str = 12.5  
=> 12.5  
[irb(main):004:0> str.class  
=> Float  
[irb(main):005:0> str = [ "ABC", 3.1415, 12, { :a => "123" } ]  
=> ["ABC", 3.1415, 12, {:a=>"123"}]  
[irb(main):006:0> str.class  
=> Array  
[irb(main):007:0> str = { :japan => "日本" }  
=> {:japan=>"日本"}  
[irb(main):008:0> str.class  
=> Hash  
irb(main):009:0> █
```

第7回レポートの採点基準

HashかClassかいずれかで表示 → B(7.5)

その一通りを、ループで出力 (0.5点加点)

両方を試している。 → A(8.5)

両方試したうち、一方をループで出力 → 9

両方とも、ループで出力 → S(9.5)

調べた内容が良い、考察が良いなど → SS(10.5)

感想とコメント

【感想】プログラムを書いている最中には気にならなかったのですが、レポートにコードを写した時点で2つのプログラムの長さの差に気づき、驚きました。自分が作りたいプログラムの構造に応じて適した書き方を調べて実装することが効率化に繋がるんだなと思いました。

【コメント】いいことに気づいたと思います。プログラムの「良さ」は、長さだけではなく、読みやすさ(メンテナンスのしやすさ)、論理的な堅牢性(ロバストネス/バグの出にくさ)、パフォーマンス(実行スピード)、DRYなど様々な要素があります。

今日のテーマ

- WEB画面上で操作した時に、データがどう流れるか、データの流れをたどる。
- また、周囲の人のPCに直接アクセスして、他の人のWEB画面を表示してみよう。
- Vmwareでのサーバ動作では、動作検証できていない部分があり、うまくいく人と、うまく行かない人ができる可能性があります。周囲の人にアクセスしてもらうというテーマで行います。

htmlでのform入力

- 「編集」edit.html.erbでは、_form.html.erbが表示される。
- _formは、form_forのメソッドで始まる。
 - このブロック内がhtmlに変換された時に、<form>タグで囲まれ、クライアントからは、この部分の入力が、サーバに送信される。
- Railsのメソッド
 - form_for / form_tag
 - <http://railsdoc.com/form>

```
edit.html.erb  index.html.erb  memos_contro
1 <h1>メモの編集</h1>
2
3 <%= render 'form' %>
4
5 <%= link_to 'Show', @memo %> |
6 <%= link_to 'Back', memos_path %>
7
```

```
edit.html.erb  _form.html.erb  memos_controlle
1 <%= form_for(@memo) do |f| %>
2   <%= if @memo.errors.any? %>
3     <div id="error_explanation">
4       <h2><%= pluralize(@memo.errors.count, "er
5
6     <ul>
7       <%= @memo.errors.full_messages.each do |me
8         <li><%= message %></li>
9       <%= end %>
10    </ul>
11  </div>
```

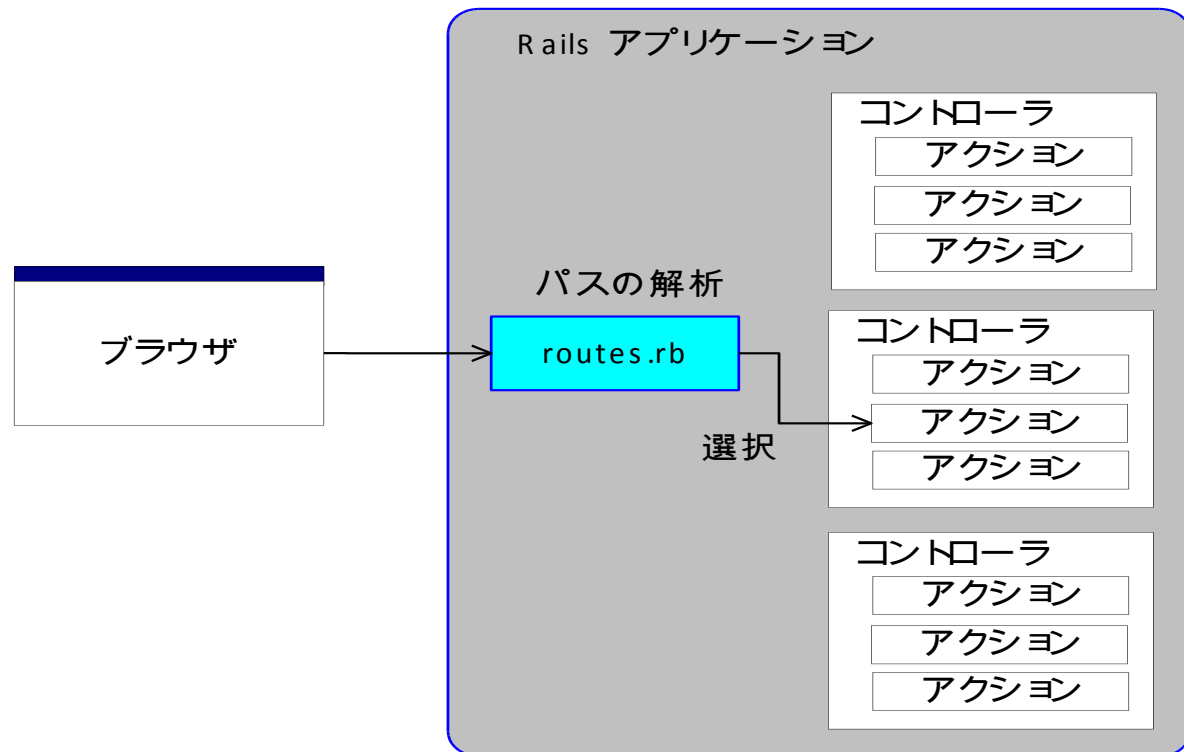

form_for, form_tagメソッド

Ruby on railsで、form_for, form_tagなどのメソッドを記述すると、HTMLのformに変換される。

HTMLのformタグでは、画面からの入力を受け付けて、パラメータをサーバに渡す。
デフォルトで、HTMLのPOSTメソッドになる。

HTMLからrubyへのデータ渡し

- htmlのformタグからPOSTされたデータを routes.rb で受け取って、どのコントローラ、メソッドに渡すか解析



memos_controller.rb

- memosのコントローラは、memos_controller.rb

- わかりやすいネーミング

- routes.rbでの記述

`post 'memos/search' => 'memos#search'`

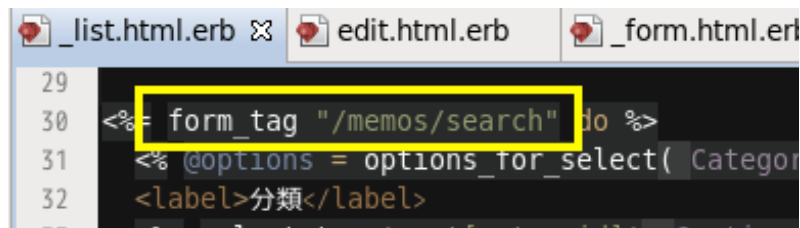
- 指定されたURLは

`http://127.0.0.1:3000/memos/search/`

- 以下のform_tagが対応している。

- デフォルト(省略時)はPOSTメソッドになる。

Controllerはmemosで
actionはsearch



```
_list.html.erb  edit.html.erb  _form.html.erb
29
30 <%= form_tag "/memos/search" do %>
31   <% @options = options_for_select( Categor
32   <label>分類</label>
33
```


POSTとGET

- HTMLからサーバへのデータの渡し方
 - Post: データの更新や書き込みなど。
 - Get: 必要なパラメータを渡して表示する。
- Searchの動作は、表示なので、本来はGETメソッドを使うべき。
 - 今回、POSTメソッドになっているsearchをGETに切り換えます。
 - 修正箇所
 - routes.rb (GETの指定にする。)
 - memos/_list.html.erb GETのパラメータを取り出す。

routesの確認

rake routesで、routesを確認する。

Getの中で、newやsearchのようにアクションを指定しているものは、:idの指定の前に来ているのは、先に解釈するため。

つまり、(バージョンにもよるが)resourcesの後に書くとうまくいかない場合がある。

```
[root@cisnote memopad]# rake routes
categories GET    /categories(.:format)      categories#index
              POST   /categories(.:format)      categories#create
new_category GET    /categories/new(.:format)  categories#new
edit_category GET    /categories/:id/edit(.:format) categories#edit
category GET    /categories/:id(.:format)  categories#show
              PUT    /categories/:id(.:format)  categories#update
              DELETE /categories/:id(.:format)  categories#destroy
memos_search GET    /memos/search(.:format)    memos#search
memos GET    /memos(.:format)           memos#index
              POST   /memos(.:format)           memos#create
new_memo GET    /memos/new(.:format)       memos#new
edit_memo GET    /memos/:id/edit(.:format)  memos#edit
memo GET    /memos/:id(.:format)       memos#show
              PUT    /memos/:id(.:format)       memos#update
              DELETE /memos/:id(.:format)       memos#destroy
[root@cisnote memopad]#
```

config/routes.rb

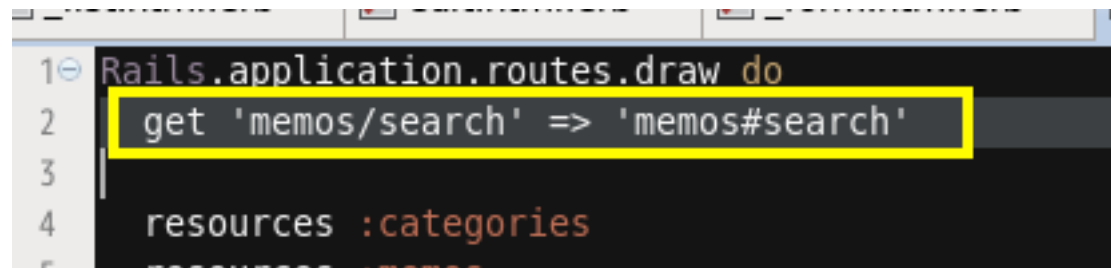
config/routes.rbの修正

修正前

```
post 'memos/search' => 'memos#search'
```

修正後

```
get 'memos/search' => 'memos#search'
```



```
1 Rails.application.routes.draw do
2   get 'memos/search' => 'memos#search'
3
4   resources :categories
5   resources :memos
```

app/views/memos/_list.html.erb

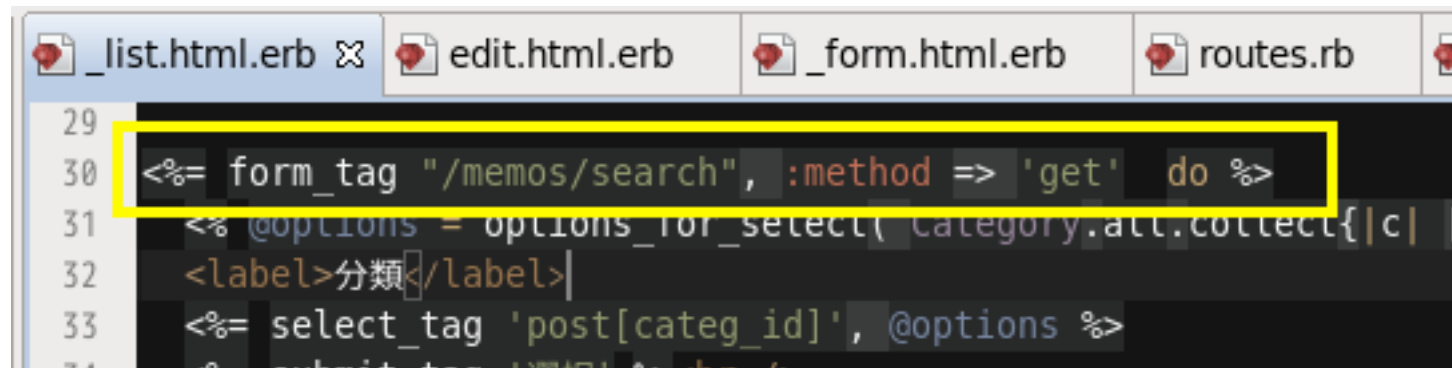
app/views/memos/_list.html.erbの修正

修正前

```
<%= form_tag "/memos/search" do %>
```

修正後

```
<%= form_tag "/memos/search", :method => 'get' do %>
```



```
_list.html.erb ✖ edit.html.erb _form.html.erb routes.rb
29
30 <%= form_tag "/memos/search", :method => 'get' do %>
31 <% @options = options_for_select(Category.all.collect{|c|
32 <label>分類</label>
33 <%= select_tag 'post[categ_id]', @options %>
34 <%= submit_tag '選択' %> %>
```


HTTPメソッドの種類

- GET 情報の要求
- POST 情報の送信
- HEAD ヘッダ部分だけを要求
- PUT アップロード
- DELETE リソースの削除
- TRACE リクエストをそのまま送り返す
- OPTIONS サーバが対応しているメソッド
を取得

パラメータ: params[]配列

- htmlのURLに?name=valueが付け加えられる。
 - erbファイル(HTMLに展開され)で、画面から入力されたパラメータを、rubyのcontrollerで受け取れる。
- GETで渡されるパラメータは、外から見えているので、隠したいコンテンツなどはGETで要求するのは不適切ということになる。

```
Started GET "/memos/search?utf8=%E2%9C%93&post%5Bcateg_id%5D=1&commit=%E9%81%B8%  
E6%8A%9E" for 127.0.0.1 at 2015-06-10 10:14:46 +0900  
Processing by MemosController#search as HTML  
Parameters: {"utf8"=>"✓", "post"=>{"categ_id"=>"1"}, "commit"=>"選択"}  
Category Load (0.2ms) SELECT "categories".* FROM "categories" WHERE "categor  
ies"."id" IS NULL LIMIT 1
```

画面formに記載できるタグ

```
<%= form_for(@foo) do |f| %>
```

f.radio_button など記述する。

- ❑ ラジオボタン(選択式の●ボタン)
 - f.radio_button
- ❑ チェックボックス(□ チェックボックス)
 - f.check_box
- ❑ 画像ファイルのアップロード用
 - file_field
- file_fieldの動作は、次回検証します。

link_to

- linkでは、クリックすると別の画面に移る。

```
<%= link_to '戻る', memos_path %>
```

- 画面に'戻る'と表示し、その文字列をクリックすると、routesのmemosに割り付けられたアクションを実行する。
- rake routesのリストの左側の「名前」に_pathをつけて読む。
- URLは/memosで、controllerはmemos, actionはindex

```
edit_category GET    /categories/:id/edit(.:format) categories#edit
  category GET    /categories/:id(.:format)      categories#show
  PUT          /categories/:id(.:format)      categories#update
  DELETE       /categories/:id(.:format)      categories#destroy
memos_search GET    /memos/search(.:format)        memos#search
  memos GET    /memos(.:format)                memos#index
  POST        /memos(.:format)                memos#create
  new_memo GET    /memos/new(.:format)           memos#new
  edit_memo GET    /memos/:id/edit(.:format)      memos#edit
  memo GET    /memos/:id(.:format)           memos#show
```

```

new.html.erb
1 <h1>New Memo</h1>
2
3 <%= render 'form' %>
4
5 <%= link_to 'Back', memos_path %>
6

```

```

memos_controller.rb
1 class MemosController < ApplicationController
2   before_action :set_memo, only: [:show, :edit, :update, :destroy]
3
4   # GET /memos
5   # GET /memos.json
6   def index
7     @memos = Memo.all
8   end
9

```

edit_category	GET	/categories/:id/edit(.:format)	categories#edit
category	GET	/categories/:id(.:format)	categories#show
	PUT	/categories/:id(.:format)	categories#update
	DELETE	/categories/:id(.:format)	categories#destroy
memos_search	GET	/memos/search(.:format)	memos#search
memos	GET	/memos(.:format)	memos#index
	POST	/memos(.:format)	memos#create
new_memo	GET	/memos/new(.:format)	memos#new
edit_memo	GET	/memos/:id/edit(.:format)	memos#edit
memo	GET	/memos/:id(.:format)	memos#show

@memos = Memo.all
で、全てのメモをデータベースから読み出す。

Methodがindexなので、index.html.erb

```

memos_controller.rb
1 <h3>Listing memos</h3>
2
3 <% @page_title = "私の専用メモ帳" %>
4
5 <%= render 'list' %>
6
7 <%= link_to '新規登録', new_memo_path %>

```

```

index.html.erb
1 <% if @memos.size == 0 then %>
2   メモは登録されていません。
3 <% end %>
4
5 <table>
6   <tr>
7     <th>Title</th>
8     <th>Name</th>
9     <th>Category</th>
10    <th></th>
11    <th></th>
12  </tr>
13
14  <% @memos.each do |memo| %>
15    <tr>
16      <td><%= memo.title %></td>

```

controllerで設定した@memosが、html.erbのここで取り出されている。

Find / where

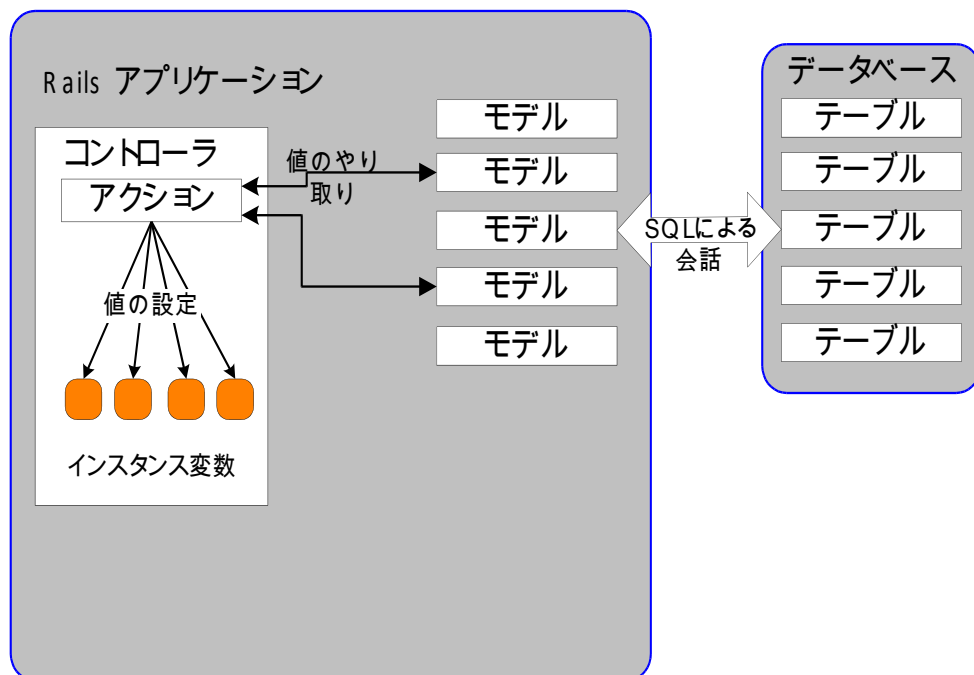
```
@memos = Memo.all      (以前は、Memo.find(:all)だった。)
```

```
@memos = Memo.where( category_id: params[:post][:categ_id])
```

```
@memo = Memo.find(params[:id])
```

```
@memo = Memo.new
```

左辺が複数か、単数かに注意する。/異なるインスタンスのclassを表現している。



- 適切なSQLを発行して、データベースから値を取得し、必要ならハッシュ配列に入れて返す

ハッシュの代入

:param => “abc”

は

param: “abc”

と書くことができ、両方とも同じです。

コロン(:)の位置に注意！

respond_to / format.html

- respond_toは、rails2からの記述
 - respond_to do |format| のformatの中身は memosコントローラのアクション名(メソッド)
 - だから、同名の _____.html.erbに飛ぶ。
- formatについて、htmlメソッドを実行し、引き続き、jsonメソッドを実行する。
- htmlメソッドでは、アドレスにjsonがなかったら index.html.erbを実行し、jsonがついていたなら 何もしない。

```
46
47 # PATCH/PUT /memos/1
48 # PATCH/PUT /memos/1.json
49 def update
50   respond_to do |format|
51     if @memo.update(memo_params)
52       format.html { redirect_to @memo, notice:
53         format.json { render :show, status: :ok,
54       else
55         format.html { render :edit }
56         format.json { render json: @memo.errors, s
57     end
58   end
59 end
```


respond_to / format.json

- format.jsonでは
アドレスに、.jsonがついていなかったら何もしない
アドレスに.jsonがついていたら、

```
{ render json: @memo }
```

@memoをjson形式で表示

を実行する。

- JSON形式や、その他の形式にも対応するための仕掛けで、XMLも記載できる。
- 書き方が `{ render :json => @memo }` から変更された。
 - たったの数文字を削る修正。

当面index.html.erbだけ考えておけばよい。

:noticeの表示エリア

- views / memos / show.html.erbの先頭に<p id='notice'><%= notice %></p>
- がある。これで :notice が表示される。
- createのメモ生成「成功」のメッセージなど

```
58  
59 #notice {  
60   color: green; }  
61
```

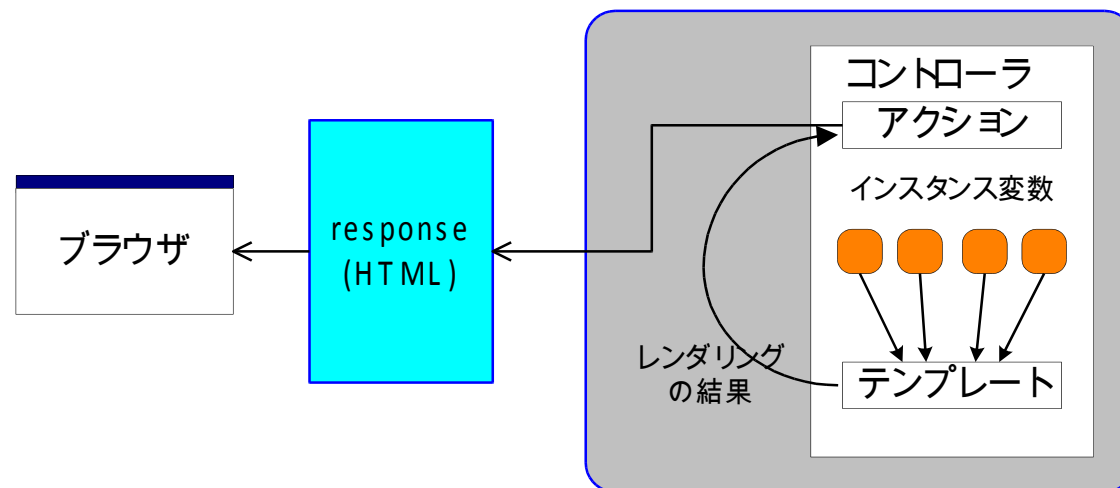
Styleは、
scaffolds.css.scssに
記述済み。

```
show.html.erb x memos_controller.rb _list.html.erb _fc  
1 <p id="notice"><%= notice %></p>  
2  
3 <p>  
4   <strong>Name:</strong>  
5   <%= @category.name %>  
6 </p>  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43
```

```
memos_cont routes.rb scaffolds.scss  
31 # POST /memos  
32 # POST /memos.json  
33 def create  
34   @memo = Memo.new(memo_params)  
35  
36   respond_to do |format|  
37     if @memo.save  
38       format.html { redirect_to @memo, notice: 'Memo was successfully created.' }  
39       format.json { render :show, status: :created, location: @memo }  
40     else  
41       format.html { render :new }  
42       format.json { render json: @memo.errors, status: :unprocessable_entity }  
43     end
```

レンダリング

- 基本的に、アクション名と同じ.html.erbのファイルで、HTMLを生成し(レンダリングし)、結果を表示する。



今日の演習課題

■ 課題

- VmwareのホストOSからLINUXにアクセスして、メモを表示・入力する。
- 可能であれば、友人(知人)のPCにアクセスして、メモを表示・入力する。
- 入力されたパラメータを、コンソール画面で確認する。
- 知人からのアクセスは、うまくいく人とうまくいかない人が、例年出てきます。うまくいかなかった人は、拘らずに諦めてください。原因究明に毎年手こずっていて、結局解決できたことはありません。

課題の進め方

Step 1: VMwareの ネットワーク接続をブリッジに切り替える。

Step 1-1: LINUXを再起動する。

Step 2: LINUXのIPアドレスを確認する。

Step 3: LINUXのファイアウォールを停止する。

Step 4: 外部公開可能モードでサーバを起動

Vmwareのネットワークアダプタ

VMwareの「仮想マシン」の

ネットワークアダプタ

を、「ブリッジ」に変更してください。

これで、次の手順が有効になります。

LINUXが起動済みの場合は、再起動してください。

ネットワークアダプタ設定 (VMwareの画面)

Rails on Scientific Linux 6.0 - VMware Workstation

仮想マシン設定

ハードウェア オプション

デバイス	概要
メモリ	1 GB
プロセッサ	2
ハード ディスク(SCSI)	20 GB
ハード ディスク 2(SCSI)	1 GB (事前に割り当て)
CD/DVD (IDE)	C:%Software%VMware%linux.iso ファ...
フロッピー	自動検出
ネットワーク アダプタ	ブリッジ (自動)
USB コントローラ	あり
サウンド カード	自動検出
プリンタ	あり
ディスプレイ	自動検出

デバイスのステータス

- 接続済み (C)
- パワー オン時に接続 (O)

ネットワーク接続

- ブリッジ: 物理ネットワークに直接接続 (B)**
- 物理ネットワーク接続の状態を複製 (P)
- NAT: ホストの IP アドレスを共有して使用 (N)
- ホストオンリー: プライベートネットワークをホストと共有 (H)
- カスタム: 特定の仮想ネットワーク (U)

VMnet0

- LAN セグメント (L)

LAN セグメント (S)... 詳

デバイス

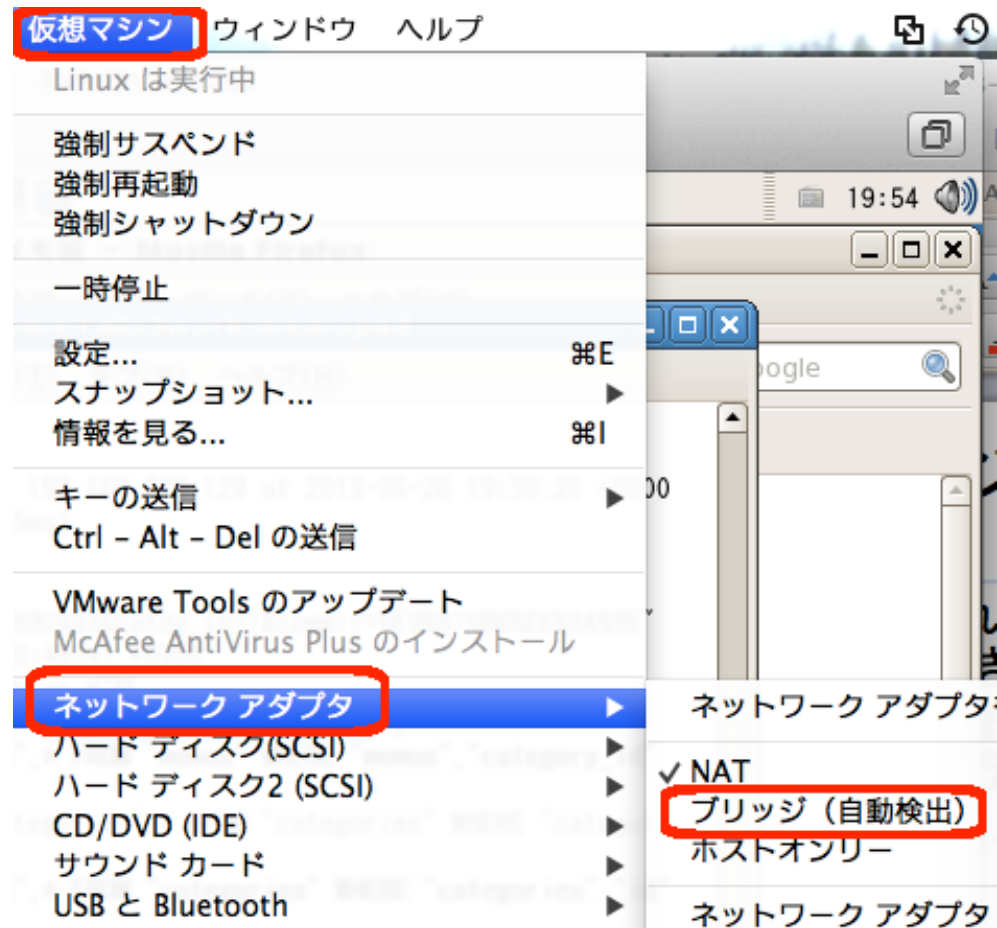
- メモリ 1 GB
- プロセッサ 2
- ハード ディスク (...) 20 GB
- ハード ディスク 2... 1 GB (事前に割...
- CD/DVD (IDE) C:%Software%...
- フロッピー 自動検出
- ネットワーク アダプタ ブリッジ (自動)**
- USB コントローラ あり
- サウンド カード 自動検出
- プリンタ あり
- ディスプレイ 自動検出

説明

ここにこの仮想マシンの説明を入力します。

Macでのbridge設定

Macを使っている場合は、以下を参照してください。

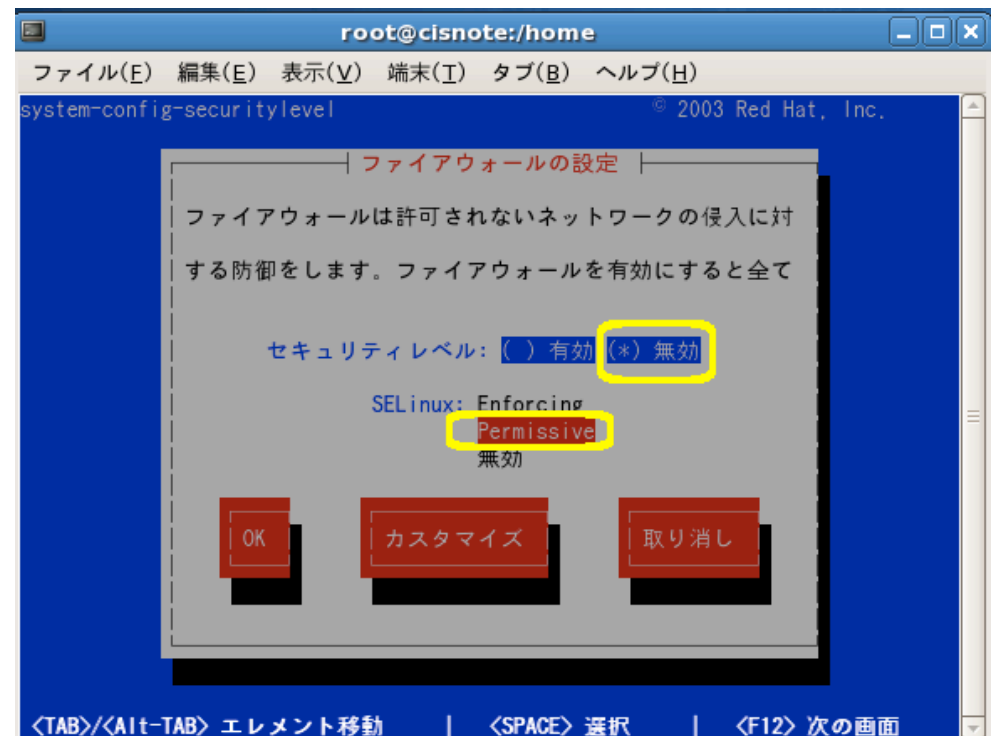


LINUXファイアウォールの停止

VMware外部からのアクセスを許可します。
GNome端末で、以下のコマンドを叩きます。

```
sudo system-config-firewall-tui
```

ファイアウォールを
無効にし、
Linuxを
再起動します。



自分のPCのIPアドレス (不要な手順です。参考まで)

- ローカルエリア接続をクリックし、「詳細」からIPv4アドレスを読み取る。

The screenshot shows two windows from the Windows 7 network control panel. The left window, titled 'ネットワーク接続の詳細' (Network Connections Details), displays the properties for the 'Apple USB Ethernet Adapter'. The 'IPv4 アドレス' (IPv4 Address) is highlighted with a red line and shows the value '192.168.0.11'. The right window, titled 'ローカル エリア接続の状態' (Local Area Connection Status), shows the status of the 'ローカル エリア接続' (Local Area Connection). The 'IPv4 接続' (IPv4 Connection) is listed as 'インターネット' (Internet). The '詳細(E)...' (Details...) button is highlighted with a red box. The 'ローカル エリア接続' (Local Area Connection) icon is also highlighted with a red box. The background shows the Windows 7 desktop with the network control panel open.

プロパティ	値
接続固有 DNS サフィックス	
説明	Apple USB Ethernet Adapter
物理アドレス	10-9A-DD-43-00-EE
DHCP 有効	(はい)
IPv4 アドレス	192.168.0.11
IPv4 サブネット マスク	255.255.255.0
リースの取得日	2011年6月29日 13:07:24
リースの有効期限	2011年7月2日 22:07:28
IPv4 デフォルト ゲートウェイ	192.168.0.1
IPv4 DHCP サーバー	192.168.0.1
IPv4 DNS サーバー	202.238.95.24

接続	インターネット
IPv4 接続:	インターネット
IPv6 接続:	インターネット アクセスなし
メディアの状態:	有効
期間:	02:39:41
速度:	100.0 Mbps

ブリッジでLINUXが使うIPアドレス

Windows上から見える VMwareのIPアドレスは、「公開」されていません。

ブリッジでは、LINUXが直接外部からIPアドレスを取得しているため、LINUXが使い、公開するアドレスは、Windowsの管理下ではありません。

LINUX上で確認し、そのIPアドレスを直接Windows上で打ち込みます。

ifconfig

LINUX上で、ifconfig のコマンドを実行して、eth0のポートを確認してください。

この eth0のIPアドレスが、外部からアクセス可能となります。

```
[root@cisnote ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:39:9A:0A
          inet addr:192.168.2.4  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:30 errors:27 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2330 (2.2 KiB)  TX bytes:684 (684.0 b)
          Interrupt:225 Base address:0x2024

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

[root@cisnote ~]# █
```

ifconfigコマンド

```
root@cisnote:~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@cisnote ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:0E:FE:DF
          inet addr:172.16.247.129  Bcast:172.16.247.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:34 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12073 (11.7 KiB)  TX bytes:1815 (1.7 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:400 (400.0 b)  TX bytes:400 (400.0 b)

[root@cisnote ~]# █
```

公開する際のサーバ起動

外部からアクセス可能にする場合には、サーバの起動コマンドに `-b 0.0.0.0` のオプションを付けます。

アクセスは、

<http://172.16.247.129:3000/memos>

例(前ページのIPの場合):

```
rails server -b 172.16.247.129
```

まず、自分のWindowsで確認

LINUX内部のブラウザで、

`http://127.0.0.1:3000/`

と指定していたアドレスを、

`http://172.16.247.129:3000/`

などのように、LINUXが外部に開いたアドレスに、Windows上のブラウザで開いてみる。

これで開ければ、ほとんど、外からもOK。

知人のPCで自分のサイトを表示

- 友人に自分のIPアドレスを伝え、自分のアプリを開いてもらう。
- これまで、127.0.0.1:3000と入力していたアドレスを、読み取ったものに切り替える。例：

http://192.168.238.129:3000

- 友人のPC上で表示できればOK



console上でのチェック

- 友人からアクセスがあると、コンソールに反応がある。

アクセスして来た
PCのIPアドレス

```
Started GET "/memos" for 192.168.11.5 at 2015-06-10 11:19:20 +0900
Cannot render console from 192.168.11.5! Allowed networks: 127.0.0.1, ::1, 127.0.0.0/127.255.255.255
Processing by MemosController#index as HTML
  (0.3ms) SELECT COUNT(*) FROM "memos"
  Memo Load (0.2ms) SELECT "memos".* FROM "memos"
  Category Load (0.1ms) SELECT "categories".* FROM "categories" WHERE "categories"."id" = ? LIMIT 1 [{"id", 2}]
  Category Load (0.1ms) SELECT "categories".* FROM "categories" WHERE "categories"."id" = ? LIMIT 1 [{"id", 1}]
  Category Load (0.2ms) SELECT "categories".* FROM "categories"
  Rendered memos/_list.html.erb (14.3ms)
  Rendered memos/index.html.erb within layouts/application (15.4ms)
  Rendered shared/_menu_bar.html.erb (1.8ms)
Completed 200 OK in 81ms (Views: 77.9ms | ActiveRecord: 0.8ms)
```

アクセスして来たIPアドレス

Formタグ
からの
パラメータ

```
Started GET "/memos/search?utf8=%E2%9C%93&categ_id=1&commit=%E9%81%B0%E5%8A%9E"
for 192.168.238.129 at 2013-06-26 19:48:47 +0900
Processing by MemosController#search as HTML
Parameters: {"utf8"=>"✓", "categ_id"=>"1", "commit"=>"選択"}
Memo Load (43.2ms) SELECT "memos".* FROM "memos" WHERE "memos"."category_id"
= 1
Category Load (0.8ms) SELECT "categories".* FROM "categories" WHERE "categori
es"."id" = 1 LIMIT 1
CACHE (0.0ms) SELECT "categories".* FROM "categories" WHERE "categories"."id"
= 1 LIMIT 1
Category Load (0.2ms) SELECT "categories".* FROM "categories"
Rendered memos/_list.html.erb (3.4ms)
Rendered memos/search.html.erb with helpers (0.0ms)
Rendered shared/_menu_bar.html.erb (0.0ms)
Rendered shared/_right_bar.html.erb (0.0ms)
Rendered shared/_footer.html.erb (0.0ms)
Completed 200 OK in 130ms (Views: 25.8ms | ActiveRecord: 44.2ms)
5
```

Search.html.erbでレンダリング

要求を受けて
実行した
SQL

外部と接続している場合、
自分のIPアドレスがこの下に
表示される。

外部からアクセスできなかった人

自分のサーバに、友人から(外部から)アクセスできなかった場合でも、友人のサーバがアクセス可能になっている場合には、その人のアドレスには接続して、友人のサイトを見ることができます。

つながっている人のサーバを開いてみて、そのコンソールを確認してください。

第3レポート (最後から2番目)

- C・B評価までの課題
 - 自分で、自分のWEBサイトにアクセスしたときの、コンソールの画面をコピーして、添付して下さい。
- A評価までの課題
 - index画面(初期画面)にアクセスがあった時のコンソールの表示で、記述のどの部分がどの処理に対応しているかを説明して下さい。
- 発展課題(S評価対象)
 - 上記までの説明に、SQLによるデータベースアクセスの解説を含めてください。
- 発展課題(1点加点)
 - LINUX上のfirewallを停止し、PCをまたいだ接続で、友人から自分のWEBサイトにアクセスしてもらって下さい。または、Vmwareのホスト(WindowsまたはMac)からアクセスして下さい。その時のコンソール画面を添付し、外部のIPアドレスを示して下さい。

設定の復活

今日の課題では、サーバとしての動作設定を行いましたので、「ファイアウォールの停止」などを選択肢に含めました。

演習が終わったら、ファイアウォールを停止を解除して、再起動してください。（演習だけ使っている人も、ファイアウォールの停止は、演習後に元に戻しましょう。）

system-config-firewall-tui失敗

設定は失敗しました

```
/usr/sbin/lokit -f -v --enabled --service=ssh
```

```
iptables: ファイアウォールルールを消去中: [ OK ]
```

```
iptables: チェインをポリシー ACCEPT へ設定中filter [ OK ]
```

```
iptables: モジュールを取り外し中:[ OK ]
```

```
ip6tables: モジュールを取り外し中:[ OK ]
```

```
iptables: ファイアウォールルールを適用中: [ OK ]
```

```
ip6tables: ファイアウォールルールを適用中: ip6tables-restore v1.4.7: ip6tables-restore: unable to initialize table 'filter'
```

Error occurred at line: 3

Try `ip6tables-restore -h` or `ip6tables-restore --help` for more information.

[失敗]

ip6tables の起動に失敗しました。

続行するにはエンターを押してください。

^C

```
[root@cisnote ~]# █
```

Ip6tables失敗について

System-config-firewall-tuiを起動し、firewallを有効に戻した際に、ip6tablesの起動に失敗する可能性が大きいです。

ですが、これ自体は実害がないと判断されるため、ここで表示される「失敗」は気にせずに次に進んで下さい。

Ctrl+Cを入力すると、コマンドを終了できます。

今日の授業を休んだ人は・・・

- レポート課題の報告を、欠席課題に兼ねます。



192.168.238.135:3000/memos

My Memopad

Web + DB入り 授業資料
2019年6月

一覧表示 | [新規追加](#) | [ruby 公式サイト](#) | [自分の課題ページ\(準備中\)](#)

メモの一覧

メモ見出し	作成者	分類	
食事に出かける	兄貴上京	至急	Show Edit Destroy
新宿の事務所訪問	山田さん	至急	Show Edit Destroy
ABCコーポ最終面接の日時	ご担当：大島さん	要連絡	Show Edit Destroy
WEB+DB第6回の欠席課題	森田	提出物	Show Edit Destroy

次回予告

- 次回は、今回までの「メモ帳」に「写真」のアップロード機能(と写真の表示)を追加します。
- 第11回は、ログイン機能(登録者のみ閲覧可能)の機能を追加します。
- この次からは、総復習として「MyTwitter」として、ゼロから作り直してみます。
- 皆さんは、自分なりのアレンジを行って下さい。