


# WEB+DBシステム (入門編)



第8回(2016年6月2日)

ruby言語の文法

# 今日のテーマ

- 引き続きRuby言語の文法構造を学ぶ。
  - クラスの定義、メソッドの定義
  - 配列やハッシュの記述
  - シンボルの記法
  - イテレータの記述
  - 正規表現
- など

# Rails環境下のクラス

Rails環境下で、主な役割を担っているのは以下のクラス。

- ❖ Model      ActiveRecord::Baseを継承
- ❖ Controller    ApplicationControllerを継承

# クラスの定義

Rails環境では、

Modelの下に

memo.rb

リレーションをここで記述

Super Classである  
ActiveRecordを継承

重要な機能は、このSuper  
Classで定義されている

```
class Memo < ActiveRecord::Base
  belongs_to :category
end
```

# Object指向での上位クラス

ユーザが定義する個々のクラスやプログラム部品の「基本形(基本機能)」を定義しており、継承することでその機能を利用できる。

- 例:iOSアプリでのUIWindowクラス
  - iPhoneやiPadでの「画面」機能を提供
- 例:rubyでのActiveRecord
  - データベース・アクセスに関する諸機能
- 例:rubyでの ApplicationController
  - 画面とデータベースを結びつける処理群を定義

# 上位クラスのAPI(ruby)


<http://www.ruby-doc.org/>

などのサイトで、最新のAPIを閲覧して、処理内容を確認しながらプログラムします。

新しいものが出たら、公式文書で確認します。

進歩が速いので、常にWEBで確認。

Home Core Std-lib Gems Downloads  Search

Ruby 1.9.3 

Ruby 1.9.3

Classes

- C ARGF
- C ArgumentError
- C Array
- C BasicObject
- C Bignum
- C Binding
- C Class

Methods

- :: === (SystemCallError)
- :: DEBUG (Thread)
- :: DEBUG= (Thread)
- :: [] (Array)
- :: [] (Dir)
- :: [] (ENV)
- :: [] (Hash)

# Railsの文書

<http://railsdoc.com/>

ドキュメントとして基礎知識から学べる。

<http://railsguides.jp/>

教科書の代わりに指定したいサイト

順番としては、このサイトの紹介から始めたかったが、(眠くなるのと、)何の説明かイメージが湧かないと思えたため、いきなりコードに触ることから始めた。

# メソッドの定義

def メソッド名(引数)

end

- 引数がない場合はメソッド名のみ
- 右の例では、二つのメソッドを定義している。

[app/controllers/memos\\_controller.rb](#)

```
# POST /memos
# POST /memos.json
def create
  @memo = Memo.new(memo_params)

  respond_to do |format|
    if @memo.save
      format.html { redirect_to @memo, notice: 'Memo was successfully
        created.' }
      format.json { render action: 'show', status: :created, location: @memo }
    else
      format.html { render action: 'new' }
      format.json { render json: @memo.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /memos/1
# DELETE /memos/1.json
def destroy
  @memo.destroy
  respond_to do |format|
    format.html { redirect_to memos_url }
    format.json { head :no_content }
  end
end
```



# 文字列の表示

- 二重引用符と一重引用符
  - それぞれ異なる引用符を中に持てるので、別の種類の引用符を中に含めるとき、使う。
- 二重引用符の場合は、以下の機能が使える
  - 変数値の展開と埋め込み
  - 制御文字コードの利用

# 演習

irbを起動して下さい。

ファイルを指定しての実行する場合は、irbを抜けて、コマンドプロンプトを使って下さい。

# 文字コードを意識しよう

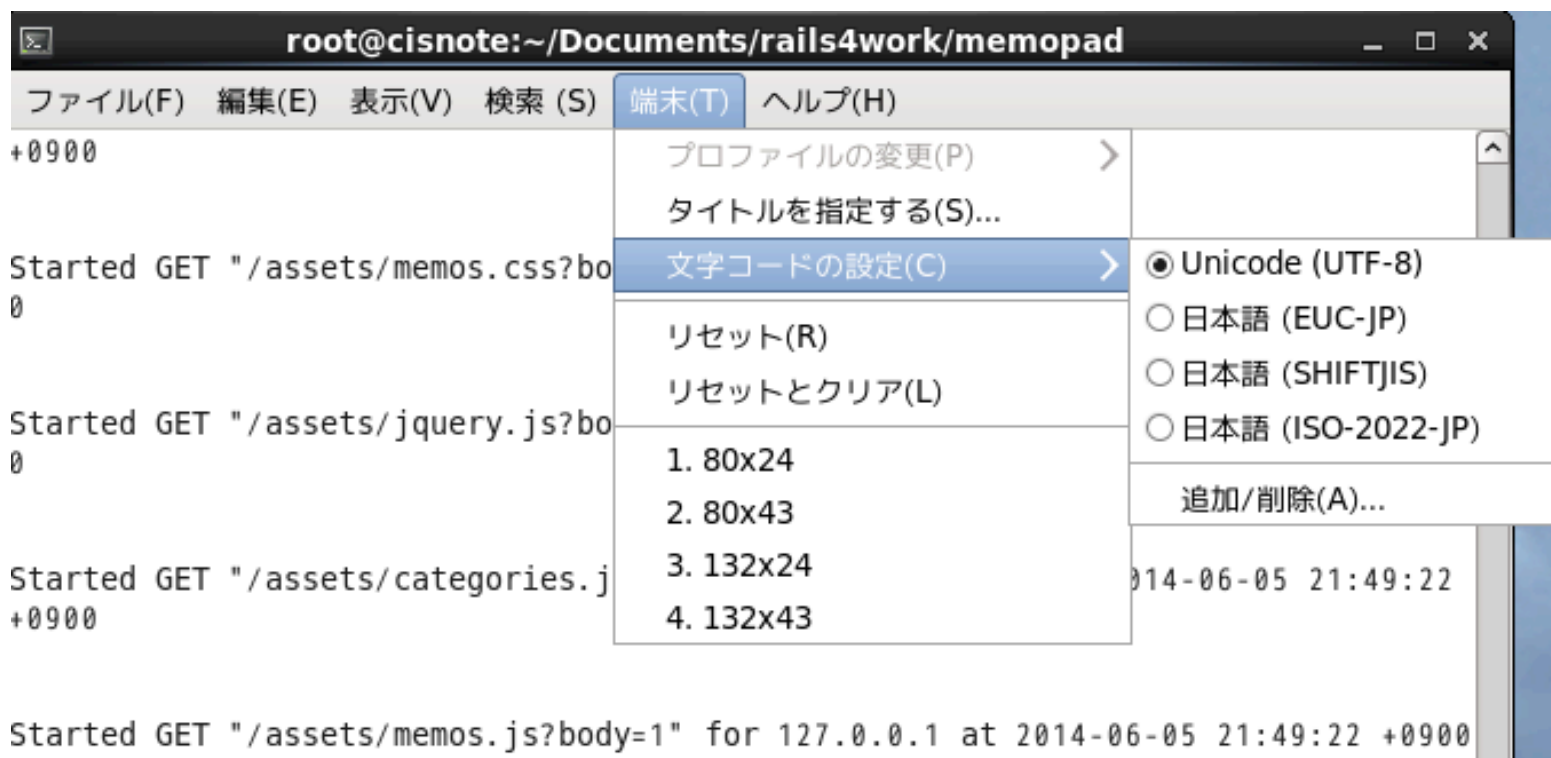
Web環境では、UTF-8をデフォルトで使用します。

テキストエディタを使う場合は、文字コードをUTF-8に設定して下さい。



# コンソール端末の文字コード

## 文字コード設定



# 文字列への式の埋め込み

- irbを起動する
- 変数nameに文字列を代入する。
- 二重引用符の文字列で、#{name} と記すと、その部分に変数nameの値が展開される。

```
root@cisnote:~/Documents/rails4work
ファイル(F) 編集(E) 表示(V) 検索 (S) 端末(T) ヘルプ(H)
[root@cisnote rails4work]# irb
irb(main):001:0> name="指原"
=> "指原"
irb(main):002:0> "#{name}さん、ようこそ"
=> "指原さん、ようこそ"
irb(main):003:0> █
```

# 書式付きの文字列プリント

- C言語と同様に、書式付きのprintfが使える。

文字コードを確認し、テキストエディタで編集して、保存する。

ruby ファイル名  
で実行。

```
#!/ ruby -Ku  
# -*- coding: UTF-8 -*-
```

```
e = 2.7182818284  
f = 123456789.12
```

```
print "e= #{ e }\n"  
print "eの2倍は、#{ 2 * e }です。 \n"  
prime100 = 541  
print "100番目の素数は#{prime100}です。 \n"
```

```
printf( "e = %5.3f\n", e )  
printf( "f = %4.3f\n", f )
```

# コマンドプロンプトでの実行

コマンドプロンプトで逐次実行させると、以下のようになります。

```
irb(main):003:0> e = 2.7182818284
=> 2.7182818284
irb(main):004:0> f = 123456789.12
=> 123456789.12
irb(main):005:0> print "e=#{ e }\n"
e=2.7182818284
=> nil
irb(main):006:0> print "eの2倍は、#{ 2 * e }です。 \n"
eの2倍は、 5.4365636568です。
=> nil
irb(main):007:0> prime100 = 541
=> 541
irb(main):008:0> print "100番目の素数は#{prime100}です。 \n"
100番目の素数は541です。
=> nil
irb(main):009:0> printf( "e=%5.3f\n", e )
e=2.718
=> nil
irb(main):010:0> printf( "=%4.3f\n", f )
=123456789.120
=> nil
irb(main):011:0> █
```

# Javaしか習っていない人へ

printfは、書式付きの値出力メソッドとして  
C言語でまず定着し、rubyでも採用され、  
Javaでも

`System.out.printf`

として使うことができます。

printfの書式は、覚えておくとかかなり便利です。

私はsprintfで「表示用文字列」の書式指定によく使っていました。



# C#やPython系のformat

#{変数の順番}

という書式指定は、C#(.netのフレームワーク)やpythonなどの言語で用いられています。

コンピュータ内部の状況(特に変数値)を、デバッガを用いずに調べる際に、有用です。

# 配列の書き方

- 配列の要素は、異なる種類のオブジェクトであっても構わない。

```
animals = [ 'dog', 'cat', 'elephant' ]
```

```
puts animals[0]
```

```
puts animals[3]
```

- 名前[ 添字 ]でアクセスする。

```
animals[3] = 'whale'
```

```
puts animals[3]
```

- 要素の追加には

```
animals << 'mouse'
```

名前 << 要素

```
puts animals
```

という書き方もできる。

# ハッシュ(連想配列)

- 定義する時は{ }を使うが呼び出す時は通常の「配列」と同じ。ただ、「添字」に文字列が使える。

```
population = {  
  'France' => 60424213,  
  'Germany' => 82424609,  
  'Italy' => 58057477  
}  
puts "Italy: #{population['Italy']}"  
population['Japan'] = 127767944  
puts "Japan: #{population['Japan']}"
```

# 条件分岐

```
if 条件式1 then
    条件式1が正しい時に実行するプログラム
elsif 条件式2 then
    条件式1が不成立で、
    条件式2が正しい時に実行するプログラム
else
    条件式1も条件式2も正しくない時のプログラム
end
```

- 1行で書く時は、thenの代わりに:が使える。

# 比較演算子・論理演算子

- 条件式に使える比較演算子
  - ==, ===, !=, >, >=, <, <=, <=>, =~, !~ など
  - ===などは左辺と右辺の位置に注意
- 条件式に使える論理演算子
  - &&, ||, !, and, or, notなど
  - &&と||の優先度は&&が高いが、andとorは一緒という点に注意する。

[http://www.ruby-lang.org/ja/old-man/html/Ruby\\_A4C7BBC8A4EFA4ECA4EBB5ADB9E6A4CEB0D5CCA3.html](http://www.ruby-lang.org/ja/old-man/html/Ruby_A4C7BBC8A4EFA4ECA4EBB5ADB9E6A4CEB0D5CCA3.html)

# 正規表現の例

```
aisatsu='おはようございます'
```

```
if /おはよう/=~ aisatsu then henji = 'おはよう'
```

```
else henji='こんにちは'
```

```
end
```

```
puts henji
```

- ファイルに記載して、aisatsuを変えながら実行してみよう。
- 上記の例では、文字列に「おはよう」が含まれていることを判定している。

# 正規表現

正規表現では、もっと複雑な表現が可能です。

例えば、メールアドレスの書式を調べるのに

```
/^[a-zA-Z0-9_%.+~]*@[a-zA-Z0-9-]*?(\.[a-zA-Z0-9_-.]*)$/
```

という記述との一致で、メールアドレスとして正しい書式かどうか、調べる事ができます。

後期に扱います。前期では、「文字列が含まれている」ことの確認でのみ、使います。

# 正規表現記号

## 公式サイトを確認

<http://doc.ruby-lang.org/ja/2.1.0/doc/spec=2fregexp.html#regexp>

特定の文字列パターンが、対象文字列に含まれているかどうかを検出する際に使われます。

この授業科目では、こうした「表現法」がある、という紹介にとどめます。



# ループによる反復

forやwhile, untilなどを一応使うこともできるが、通常はイテレータを使うことが多い。

```
array1 = [ '三鷹', '立川', '八王子' ]  
i = 0  
while array1[i]  
  puts array1[i]  
  i += 1  
end
```

# イテレータによる反復

- times, upto, each, each\_with\_indexなどメソッドとして使用する。

```
10.times { |i| print i, ', ' }
```

```
array1 = [ '三鷹', '立川', '八王子' ]  
array1.each do |item|  
  print item + ', '  
end
```

# 今日の演習課題

- メモを、分類コードで検索する。
- 特定の「分類コード」のメモだけを抽出して、画面に表示させる。
- **【準備】**
  - 分類コードを3種類程度登録する。
  - メモを5件程度用意し、そのうちの2件程度に、「検索」させる分類コード(例えば「至急」)を設定しておく。

# memos\_controller.rb

<http://railsdoc.com/references/find>

```
# POST /memos/search
# POST /memos/search.json
def search
  @memos = Memo.where( category_id: params[:post][:categ_id])
  @category = Category.find(params[:post][:categ_id])
end
```

- indexの下に searchメソッドを追加した例

```
14
15 # POST /memos/search
16 # POST /memos/search.json
17 def search
18   @memos = Memo.where( category_id: params[:post][:categ_id])
19   @category = Category.find(params[:post][:categ_id])
20 end
21
22 # GET /memos/new
23 def new
24   @memo = Memo.new
25 end
26
```

# Railsのfindメソッド

Rails3からRails4にかけて、変更があります。

[http://railsdoc.com/references/find\\_all\\_by](http://railsdoc.com/references/find_all_by)

公式文書で、findメソッド、find\_byメソッド、find\_all\_byメソッドなどを参照してドキュメントを読んでください。

# index.html.erb (1)

- 先頭部分に、if節を挿入する。

```
<% if @memos.size==0 then %>  
  メモは登録されていません。  
<% end %>
```

- メモがない時に表示される。



```
category.rb  memos_controller.rb  *index.html.erb x  
1 <h1>Listing memos</h1>  
2  
3 <%= @page_title = '私の専用メモ帳' %>  
4  
5 <%= if @memos.size==0 then %>  
6   メモは登録されていません。  
7 <%= end %>  
8  
9 <table>  
10 <tr>  
11   <th>Title</th>  
12   <th>Name</th>  
13   <th>Category</th>  
14   <th></th>  
15   <th></th>
```

# index.html.erb (2)

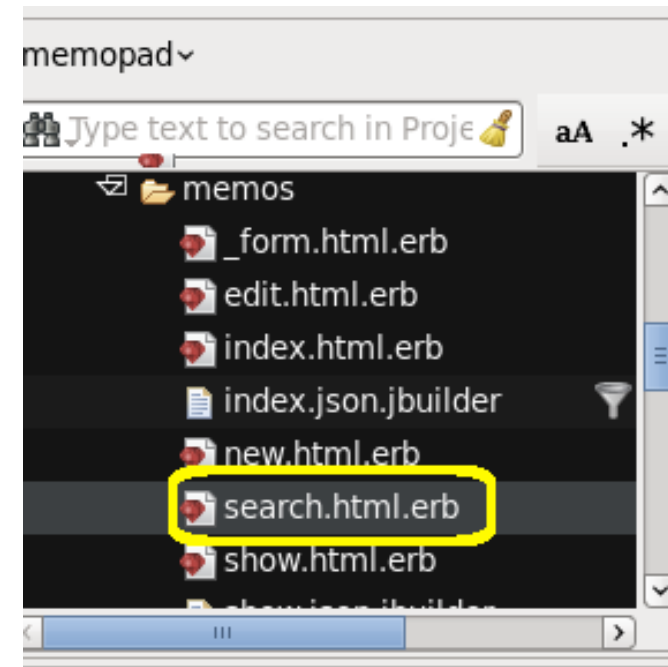
- tableの後、new memoへのリンクの前に、カテゴリ選択のフォームを挿入する。
- form\_tag節

```
<%= form_tag "/memos/search" do %>
  <% @options = options_for_select( Category.all.
    collect{|c| [c.name,c.id]}) %>
  <label>分類</label>
  <%= select_tag 'post[categ_id]', @options %>
  <%= submit_tag '選択' %><br />
<% end %>
<br />
```

```
</table>
<%= form_tag "/memos/search" do %>
  <% @options = options_for_select( Category.all.
    collect{|c| [c.name,c.id]}) %>
  <label>分類</label>
  <%= select_tag 'post[categ_id]', @options %>
  <%= submit_tag '選択' %><br />
<% end %>
<br />
<br>
```

# search.html.erb

- app→views→memosを右クリックしてから、「新規」で「ファイル」を挿入する。





# テンプレートは使わない

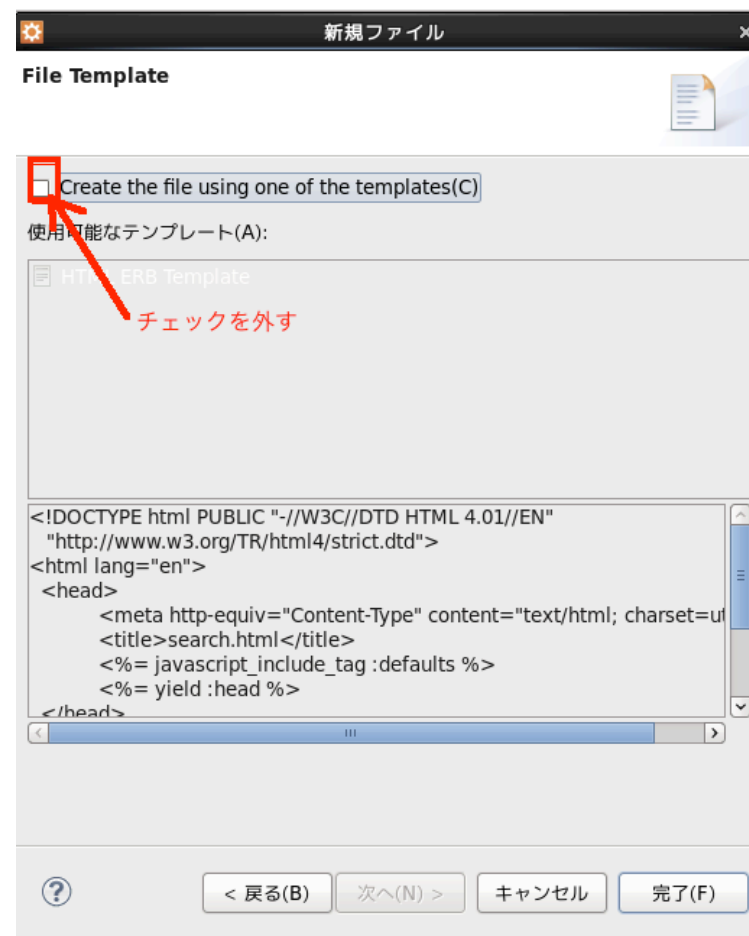
メイン構造は

layout/application.html.erb

で記述されていて、生成される個別  
のhtmlファイルは

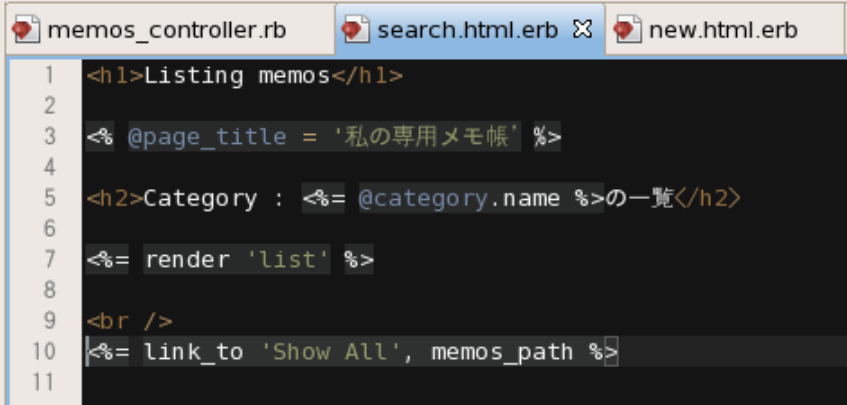
<%= yield %>

の部分に置き換えられるため。



# search.html.erbの内容

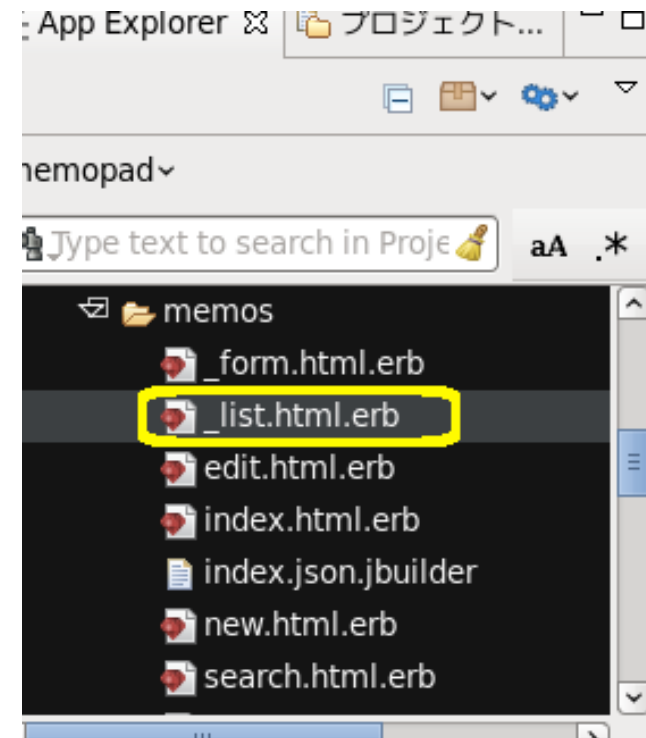
- DRY (Don't Repeat Yourself) の設計コンセプトでやります。
- index.html.erbと共通する部分を、
- \_list.html.erb  
として抽出する。
- index.html.erbにも  
同様の処理を行う。



```
1 <h1>Listing memos</h1>
2
3 <%= @page_title = '私の専用メモ帳' %>
4
5 <h2>Category : <%= @category.name %>の一覧</h2>
6
7 <%= render 'list' %>
8
9 <br />
10 <%= link_to 'Show All', memos_path %>
11
```

# \_list.html.erb

- app→views→memosを右クリックしてから、「新規」で「ファイル」を挿入する。



# \_list.html.erb

元々のindex.html.erb  
の5行目以下を、  
ほとんど全部コピー  
する。

```
<% if @memos.size==0 then %>  
  メモは登録されていません。  
<% end %>
```

```
<table>  
<tr>  
  <th>Title</th>  
  <th>Name</th>  
  <th>Category</th>
```

(中略)

```
<%= form_tag "/memos/search" do %>  
  <% @options = options_for_select( Category.all.  
    collect{|c| [c.name,c.id]}) %>  
  <label>分類</label>  
  <%= select_tag 'post[categ_id]', @options %>  
  <%= submit_tag '選択' %><br />  
<% end %>  
<br />
```

```
<%= link_to 'New Memo', new_memo_path %>
```

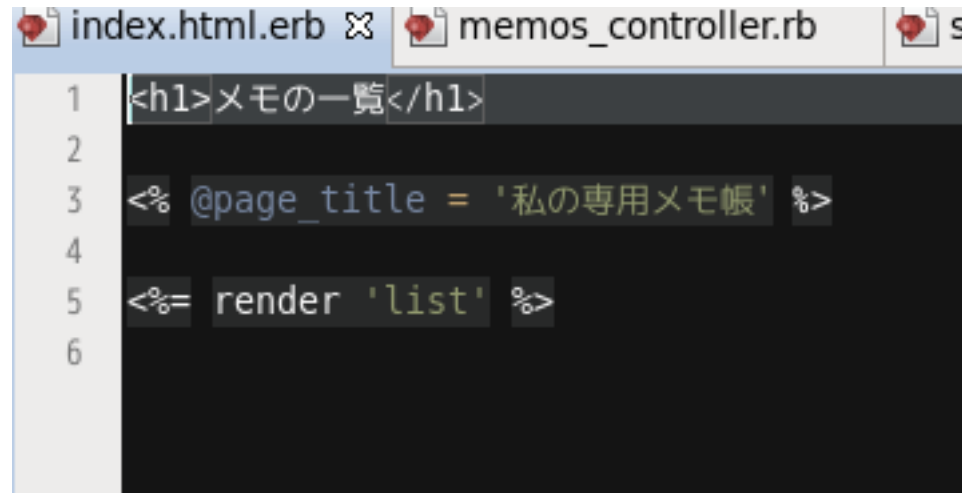
# index.html.erb

```
<p id="notice"><%= notice %></p>
```

```
<h1>メモの一覧</h1>
```

```
<%= render 'list' %>
```

中身の大半は  
\_list.html.erb  
に引っ越し済みなので、  
3行だけ残った。



```
index.html.erb  memos_controller.rb  s
1 <h1>メモの一覧</h1>
2
3 <%= @page_title = '私の専用メモ帳' %>
4
5 <%= render 'list' %>
6
```

# search.html.erb

```
<p id="notice"><%= notice %></p>
```

```
<h1>メモの一覧</h1>
```

```
<h2>Category : <%= @category.name %>の一覧</h2>
```

```
<%= render 'list' %>
```

```
<br />
```

```
<%= link_to '全部の表示', memos_path %>
```

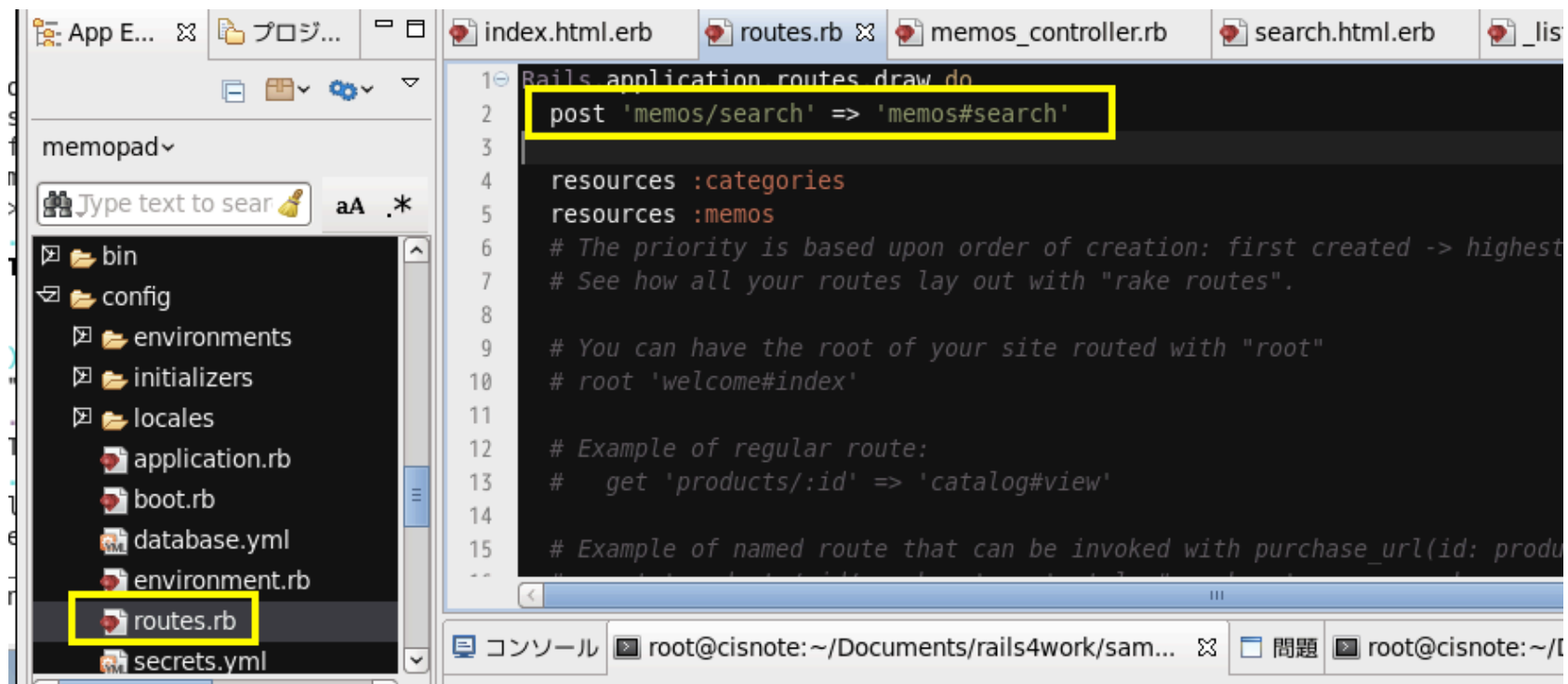
A screenshot of a code editor window showing the content of a file named 'search.html.erb'. The editor has three tabs at the top: 'index.html.erb', 'routes.rb', and 'memos\_controller.rb'. The main editing area shows the following code:

```
1 <p id="notice"><%= notice %></p>
2
3 <h1>私の独り言的メモです</h1>
4
5 <h2>Category : <%= @category.name %>の一覧</h2>
6
7 <%= render 'list' %>
8 <br />
9 <%= link_to '全部の表示', memos_path %>
10
```

# config/routes.rbの設定

先頭部分に以下の1行を追加します。

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



# ここまでの改修でどうなったか？

- 一覧表示の下に、分類コードの選択ボタンが現れた。
- ここで、「分類」を選ぶと・・・  
選択されたメモだけが表示される。



The screenshot shows the Memopad application interface. At the top, there is a browser address bar with the URL "127.0.0.1:3000/memos". Below the browser, there is a header area with a "My Memopad" logo and a navigation menu. The main content area is titled "メモの一覧" (Memo List) and contains a table of memos. A dropdown menu is open over the "分類" (Category) column, showing options: "至急" (Urgent), "要連絡" (Need Contact), "提出物" (Submission), and "来客あり" (Guest Present). The "至急" option is highlighted with a red box. A red arrow points from the "至急" option in the dropdown menu to the "メモの一覧" section on the right.

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 Destr
新宿の事務所訪問	山田さん	至急	Show Edit Destr
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
東京農大チーム、合同練習打ち合わせ	坂本	来客あり	Show Edit Destroy

Copyright (C) by Ikuro Kobayashi

## メモの一覧 Category: 至急の一覧

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

分類 至急 選択

New Memo  
Show All



# 暗黙の約束事(1)

form\_tagからは、POSTメソッドでREQUESTが送出される。

パラメータは、params[:post][:hoge]で抽出する。

送る側は、post[hoge]を文字列で指定する。

```
32 | <% @options = options_for_select( Category.find(:all).
33 | <label>分類</label>
34 | <%= select_tag 'post[categ_id]', @options %>
35 | <%= submit_tag '選択' %><br />
36 | <% end %>
37 | <br />
38 |
```

```
def search
  @memos = Memo.where( category_id: params[:post][:categ_id])
  @category = Category.find_by_id(params[:post][:categ_id])
end
```

## 暗黙の約束事(2)

ControllerのActionのメソッド名からは、  
同名のviewのhtml.erbファイルが呼ばれる。

```
Started POST "/memos/search" for 127.0.0.1 at 2014-06-06 00:04:55 +0900
Processing by MemosController#search as HTML
  Parameters: {"utf8"=>"✓", "authenticity_token"=>"nomAxrw/jD/N4jtKct9opa9UMPH//
ItpMXlPx5Ijmfg=", "post"=>{"categ_id"=>"1"}, "commit"=>"選択"}
DEPRECATION WARNING: This dynamic method is deprecated. Please use e.g. Post.where(...)
.all instead. (called from search at /root/Documents/rails4work/memopad/app/controllers/memos_controller.rb:13)
Memo Load (0.2ms) SELECT "memos".* FROM "memos" WHERE "memos"."category_id" =
1
Category Load (0.1ms) SELECT "categories".* FROM "categories" WHERE "categories"."id" = 1 LIMIT 1
Category Load (0.1ms) SELECT "categories".* FROM "categories" WHERE "categories"."id" = ? LIMIT 1 [["id", 1]]
CACHE (0.0ms) SELECT "categories".* FROM "categories" WHERE "categories"."id"
```

# 余談ですが・・・

サンプルプログラムをネットなどで掲示する中で、「意味のない部分」を明示するため、よく使われる変数名が、

foo, bar, zot

などで、日本語のサイトでは、

hoge, moge, fuga, moga

などが使われます。

「サンプルだから、意味のある単語に置き換えて使ってね」という気持ちが込められています。この部分には意味のある単語が入る事になります。

「部下がJavaのプログラムにhoge, mogemogeなどを多用していたので、呆れて絶句した」というシニアSEの方がいましたがホゲホゲ、もげもげ、フガフガ、もがもがは商用のソースコードには残さないようにしましょう。

# \_list.html.erb

共通部分を抜き出します。

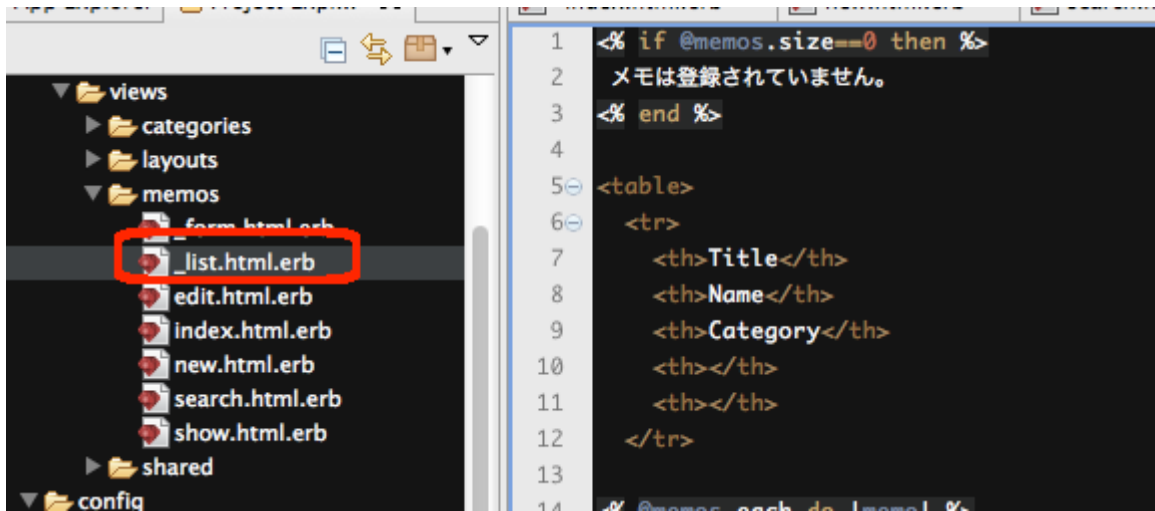
```
<% if @memos.size==0 then %>  
  メモは登録されていません。  
<% end %>
```

から始まり、form\_tagのブロックの<% end %>

```
<%= select_tag 'post[categ_id]', @options %>  
<%= submit_tag '選択' %><br />  
<% end %>
```

までを、\_list.html.erbに書き写します。

# \_list.html.erb



```
<%= form_tag "/memos/search" do %>
| <% @options = options_for_select( Category.all.collect{|c| [c.name,c.id]}) %>
  <label>分類</label>
  <%= select_tag 'post[categ_id]', @options %>
  <%= submit_tag '選択' %><br />
<% end %>
<br />
```

# ここまでの修正での問題点

- メモが存在しない場合でも、見出しが表示される。
  - メモが存在しない場合はメッセージだけで一覧表示はしないようにする。
  - というプログラムは、if / elseの基本です。
  - 皆さんが修正して下さい。→[授業課題](#)

# レポート課題

- 今日は、レポート課題はありません。

# 今日の授業を休んだ人は・・・

分類コードで画面検索できるようにし、そのプログラム修正内容をスクリーンショットとともに報告して下さい。



[一覧表示](#) | [新規追加](#) | [ruby 公式サイト](#) | [自分の課題](#)

## メモの一覧

Category : 至急の一覧

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](#)

分類

[New Memo](#)

[Show All](#)

Copyright (