



OBJECT ORIENTED WEB PROGRAMMING USING RUBY

Day 9: 08/June/2017

Table Design & Screen Design



Presentation on your design

Screen proto-types

Database Schema

How many screens do you write?

What kind of data do you need to store?

Think about the screen links.



Sample Application

Memopad

How to set up database table

Scaffold the Entry Screen

The command to scaffold the entry screen is as the following;

```
rails g scaffold ik_memo content:text
```

We can shorten 'generate' to g.

This command create 'ik_memos' table with the text field of content.

```
Running via Spring preloader in process 6227
; rails g scaffold ik_memo content:text
invoke active_record
create db/migrate/20170607081542_create_ik_memos.rb
create app/models/ik_memo.rb
invoke test_unit
create test/models/ik_memo_test.rb
create test/fixtures/ik_memos.yml
invoke resource_route
route resources :ik_memos
invoke scaffold_controller
create app/controllers/ik_memos_controller.rb
invoke erb
create app/views/ik_memos
create app/views/ik_memos/index.html.erb
create app/views/ik_memos/edit.html.erb
create app/views/ik_memos/show.html.erb
create app/views/ik_memos/new.html.erb
```

Database Generation

The command to let Sqlite3 create database is;

`rake db:migrate`

Now we are ready to run the application.

```
[== 20170607081542 CreateIkMemos: migrating =====  
-- create_table(:ik_memos)  
-> 0.1158s  
== 20170607081542 CreateIkMemos: migrated (0.1159s) =====
```

Table name prefix

In order to identify the table groups, for this case, make the project owner clear, I had put prefix `ik_` to memos' table.

Put your own prefix for the tables you had designed, when you install the tables.

How to setup Relation?

Now generate table Category, then
Setup the relationship between memos table
and categories table.

Generate Category table

Type the following command;

```
rails g scaffold ik_category name:string
```

Then, create the database

```
rake db:migrate
```

All most the same procedure with memos.

Migration to Add Column

We generate migration file of adding Column to Memos table by typing the following command;

```
rails generate migration AddCategoryIdToMemos
```

Note that the generated Migration file needs to be modified.

```
[WebDB@cisnote memopad]$ rails g migration AddCategoryIdToMemos
Running via Spring preloader in process 13983
    invoke  active_record
    create  db/migrate/20161026130256_add_category_id_to_memos.rb
[WebDB@cisnote memopad]$
```

Migration Name and the Meaning

Migration name itself have the meaning as a command!

“Add CategoryId to Memos”

“AddFieldnameToTablename” is one of the migration pattern.

Category Id is a relation field to the table categories.

See the document site;

<http://api.rubyonrails.org/classes/ActiveRecord/Migration.html>

Camel Case or Snake Case?

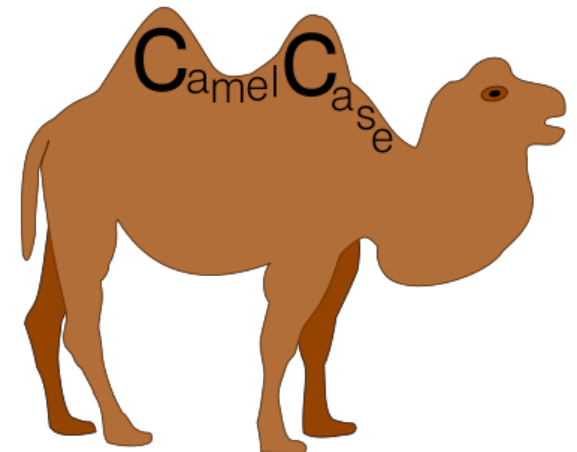
Here the Class names are Upper Camel Case, and file names are snake cases.

We can convert the Camel Case name to/from snake case name.

1: require "active_support/inflector"

2: "snake_case".camelize

3: "CamelCase".underscore



Migration File

Add the following two lines in the

2017mmddhhMMss_add_category_id_to_memos.rb file;

add_column :ik_memos, :ik_category_id, :integer

IkMemo.reset_column_information

In ActiveRecord::Migration, there are
'add_column' and some other methods to
change schema available.

```
1 class AddIkCategoryIdToIkMemos < ActiveRecord::Migration[5.0]
2   def change
3     add_column :ik_memos, :ik_category_id, :integer
4     IkMemo.reset_column_information
5   end
6 end
7
```

Migration and the name

- Insert link_id to ik_categories in IkMemos table;
 - Ik_category_id can be used as a link.
- There is a rule in Ruby

Convention over Configuration

- The name of Relation field is "name_id" where the name should be singular form.
 - It will be `ik_category_id` in this case.

One to Many Relationship

- From memos to categories, choose one.
- From categories, there are many memos in one category.

In memo model (app/models/ik_memo.rb)

`belongs_to :ik_category`

In category model (app/models/ik_category.rb)

`has_many :ik_memos`

Be careful of singular/plural form.

One to One, Many to Many cases

- In case of One to One, either one should become main.
 - Main model `has_one`
 - Sub model `belongs_to`
- In case of Many to Many relationship,
 - both `has_and_belongs_to_many`

Singular form or plural form?

- `belongs_to, has_one(space):(singular)`
- `has_many, has_and_belongs_to_many`
cases, (space):(colon)(plural form)
- Ruby has method to convert singular form into plural form, and plural form into singular form.

Type of relations between tables

- One to one
 - Student number $\leq \Rightarrow$ Student Name
- One to many
 - Name in address book $\leq \Rightarrow$ Phone numbers
- Many to many
 - Guest in restaurant $\leq \Rightarrow$ ordered dishes


Preparation for the last modification of relationship

When we miss the link, it will cause an error in tracing the link from ik_memos to ik_categories.

So, make sure that we have 'ik_category_id' field values in all memos.

Make sure we have categories list

Like memos listing screen, we have whole set of screens to add, edit, show and remove the categories.



ScapsOwl Project
Web+DB Hosei 2017 Project

[Top Page](#) [Chat Sample](#) [ruby Official Site](#)

Ik Categories

| Name | |
|------------------|---|
| Idea | Show Edit Destroy |
| Meet with people | Show Edit Destroy |

[New Ik Category](#)

links

[TOP](#)
[Chat Sample](#)
[Lecturers](#)

AD space for RENT

Copyright (C) 2017, by Web+DB Hosei Project Team

Ik_category_id value first!

In order to activate the link from memos table to categories table, to use

`<%= ik_memos.ik_category.name %>`

if category_id is not given yet, the above sentence will cause method error.

(method call from `nil` object.)

Make sure that you can show the value by

`<%= ik_memos.ik_category_id %>`

app/views/ik_memos/ index.html.erb

```
<table>
<thead>
  <tr>
    <th>Content</th>
    <th>Category Id</th>
    <th colspan="3"></th>
  </tr>
</thead>

<tbody>
  <% @ik_memos.each do |ik_memo| %>
    <tr>
      <td><%= ik_memo.content %></td>
      <td><%= ik_memo.ik_category_id %></td>
      <td><%= link_to 'Show', ik_memo %></td>
      <td><%= link_to 'Edit', edit_ik_memo_path(ik_memo) %></td>
      <td><%= link_to 'Destroy', ik_memo, method: :delete, data: { confirm: 'Are you sure?' } %></td>
    </tr>
  <% end %>
</tbody>
</table>
```

views/ik_categories/ index.html.erb

```
<table>
  <thead>
    <tr>
      <th>Id</th>
      <th>Name</th>
      <th colspan="3"></th>
    </tr>
  </thead>
```

Add the display of ID field , to confirm which 'category' has what ID value.

```
<tbody>
  <% @categories.each do |category| %>
    <tr>
      <td><%= category.id %></td>
      <td><%= category.name %></td>
      <td><%= link_to 'Show', category %></td>
      <td><%= link_to 'Edit', edit_category_path(category) %></td>
      <td><%= link_to 'Destroy', category, method: :delete, data: { confirm: 'Are you sure?' } %></td>
    </tr>
  <% end %>
</tbody>
</table>
```

Add Ik_category_id input in ik_memos _form

Edit `app/views/ik_memos/_form.html.erb`

Add the following four lines;

```
<div class="field">  
  <%= f.label :ik_category_id %><br />  
  <%= f.text_field :ik_category_id %>  
</div>
```

```
3  
4  <div class="field">  
5    <%= f.label :content %>  
6    <%= f.text_area :content %>  
7  </div>  
8  
9  <div class="field">  
10    <%= f.label :ik_category_id %><br />  
11    <%= f.text_field :ik_category_id %>  
12  </div>  
13  
14  <div class="actions">  
15    <%= f.submit %>
```

Gem: Strong Parameters

This gem is taken in from Rails 4 into the core function.

We maintain the white list of parameters, in controllers.

Edit `app/controllers/ik_memos_controller.rb`

Add `ik_category_id` into the permitted list.

```
59     format.html { redirect_to ik_memos_url, notice: "ik memo was successfully destroyed." }
60     format.json { head :no_content }
61   end
62 end
63
64 private
65   # Use callbacks to share common setup or constraints between actions.
66   def set_ik_memo
67     @ik_memo = IkMemo.find(params[:id])
68   end
69
70   # Never trust parameters from the scary internet, only allow the white list through.
71   def ik_memo_params
72     params.require(:ik_memo).permit(:content, :ik_category_id)
73   end
74 end
75
```


Drop Down list

Modify app/views/memos/_form.html.erb

Now coment out f.text_field :category_id,
and then add the following line.

```
<%= f.select :category_id, Category.all  
      .collect{|c|[c.name,c.id]} %>
```

```
13  
14   <div class="field">  
15     <%= f.label :content %>  
16     <%= f.text_area :content %>  
17   </div>  
18  
19   <div class="field">  
20     <%= f.label :ik_category_id %><br />  
21     <%= f.select :ik_category_id, Ik.Category.all.collect{|c|[c.name,c.id]} %>  
22   </div>  
23  
24   <div class="actions">
```

Now you can trace the relation

In views/ik_memos/index.html.erb

```
<td><%= ik_memo.ik_category_id %></td>
```

Can be now:

```
<td><%= ik_memo.ik_category.name %></td>
```

The above `ik_category_id` is just a field value of the `ik_memos` table, however, now we have relation from `ik_memo` to `ik_category`, so we can use any value like `name` in the `ik_categories` table.



Installation of Relationship

To realize the 3rd order normalization, make the most of relational table design!

Enjoy!

Next week: Understanding Test Script Writing

Now we learn the TDD (Test Driven Development) and/or BDD (Behavior Driven Development).

Next week, we will introduce this concept.