Web System Development with Ruby on Rails

Day 11(6/Dec/2012)

File uploading and Image Display

Today's Theme

- Upload image files to the database, and let Memopad store the image file.

- Try some other file types, such as sound file play.

Design Concept of Image (Figure) Attachment

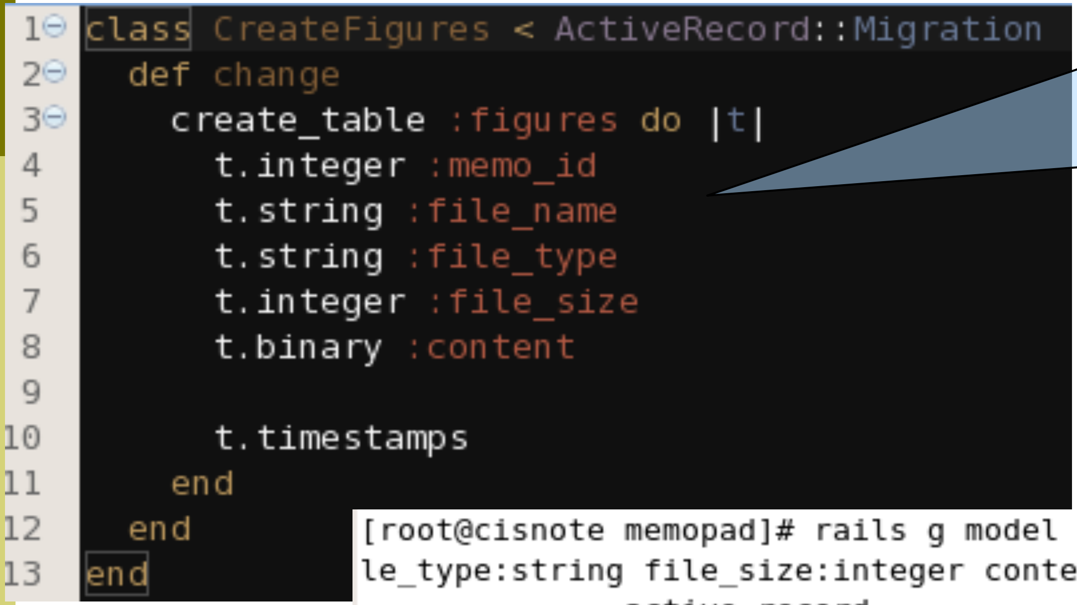
- Table Name: figures
- Model name : figure
- Relationship : (memos : figures) => 1 to many
  - One ’memo' can have many images (figures)
  - One picture belongs to only one memos
  - Memos which do not have figures have no problem without having any figures with it.
- Figures has ’memo_id' field for relation information.

Generate Figure model

- Input the following command to generate the model 'Figure'

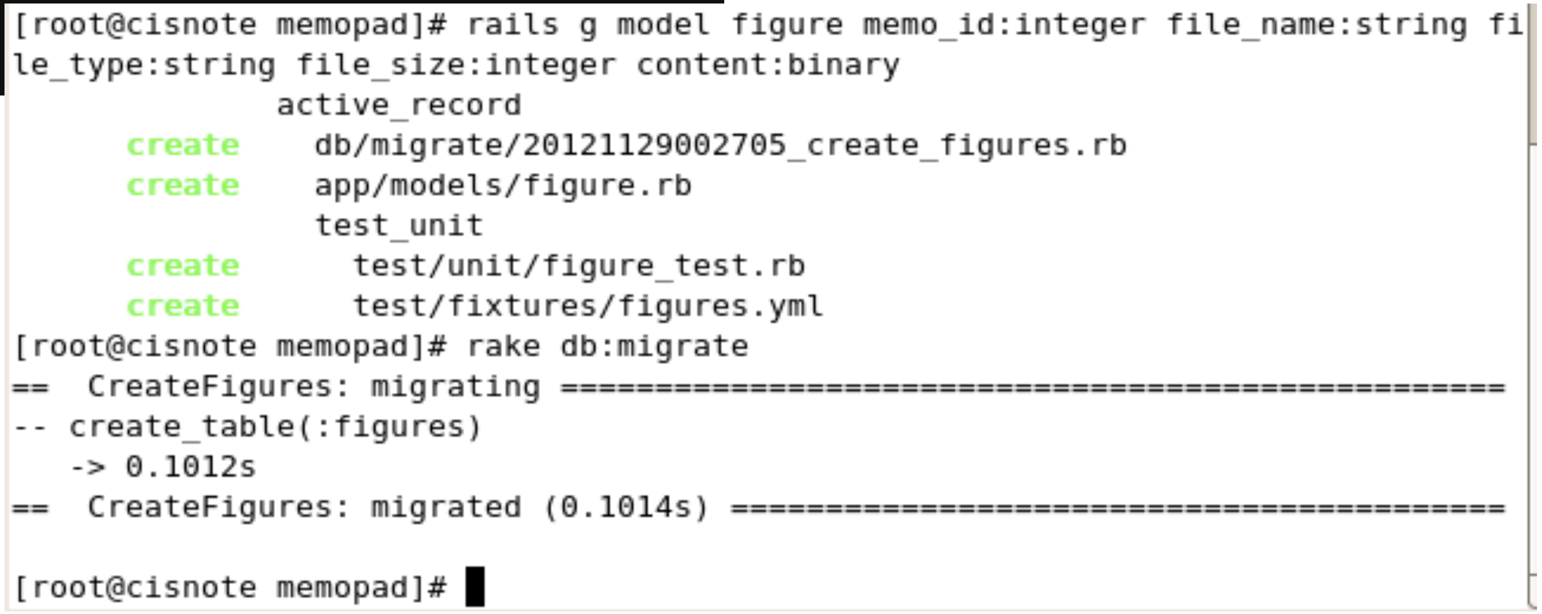
```
rails generate model figure
  memo_id:integer file_name:string
  file_type:string file_size:integer
  content:binary
```

Figure class structure



```ruby
class CreateFigures < ActiveRecord::Migration
  def change
    create_table :figures do |t|
      t.integer :memo_id
      t.string :file_name
      t.string :file_type
      t.integer :file_size
      t.binary :content

      t.timestamps
    end
  end
end
```

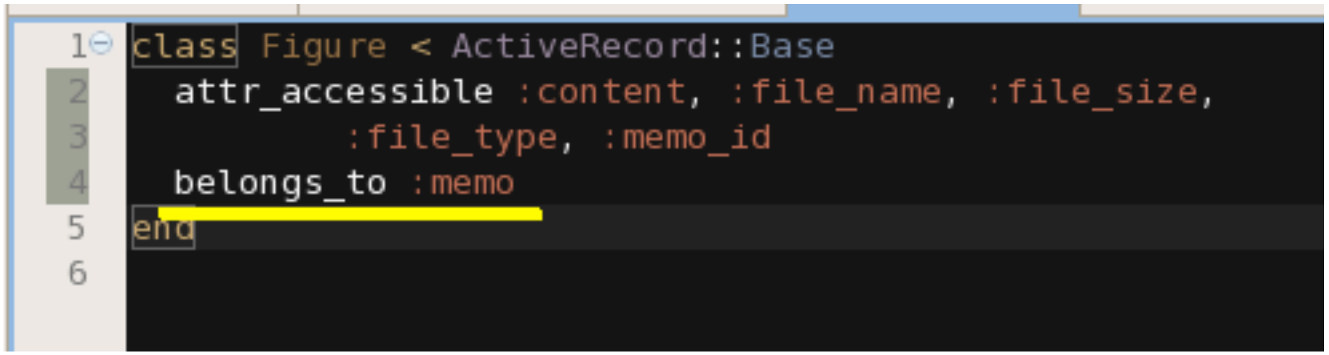
memo_id is the relation id for memos table. 'file_type' is MIME Type name, the kinds of type such as gif, jpg and png. Content contains the image file itself.



```
[root@cisnote memopad]# rails g model figure memo_id:integer file_name:string fi
le_type:string file_size:integer content:binary
              active_record
      create    db/migrate/20121129002705_create_figures.rb
      create    app/models/figure.rb
                test_unit
      create      test/unit/figure_test.rb
      create      test/fixtures/figures.yml
[root@cisnote memopad]# rake db:migrate
==  CreateFigures: migrating ==================================================
-- create_table(:figures)
   -> 0.1012s
==  CreateFigures: migrated (0.1014s) =========================================

[root@cisnote memopad]# 
```

Setting up Relations

- Looking from figures, they only belong to one memo, so the specification of belongs_to :memo is added to figure.rb
- Modify app/models/figure.rb



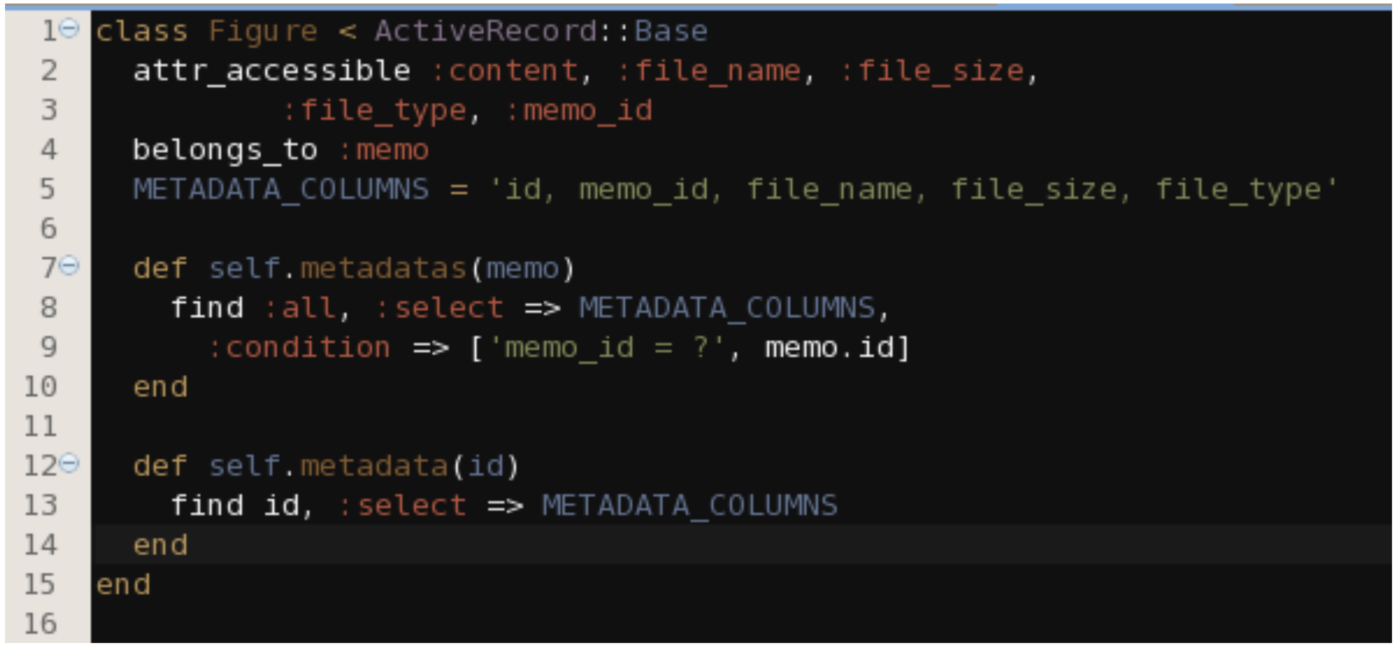
```
class Figure < ActiveRecord::Base
  attr_accessible :content, :file_name, :file_size,
          :file_type, :memo_id
  belongs_to :memo
end
```

Setting up Metadata

- If the image size is small, we may not mind the waiting time to show the image, instead when the file size is great, we may be irritated.
  - (It is not the sole reason.)
- So, make the metadata (data for data) to show the metadata information before we obtain the file.

Metadata Definition (figure.rb)

```
METADATA_COLUMNS = 'id, memo_id, file_name, file_size, file_type'

def self.metadatas(question)
  find :all, :select => METADATA_COLUMNS,
      :condition => ['question_id = ?', question.id]
end

def self.metadata(id)
  find id, :select => METADATA_COLUMNS
end
```

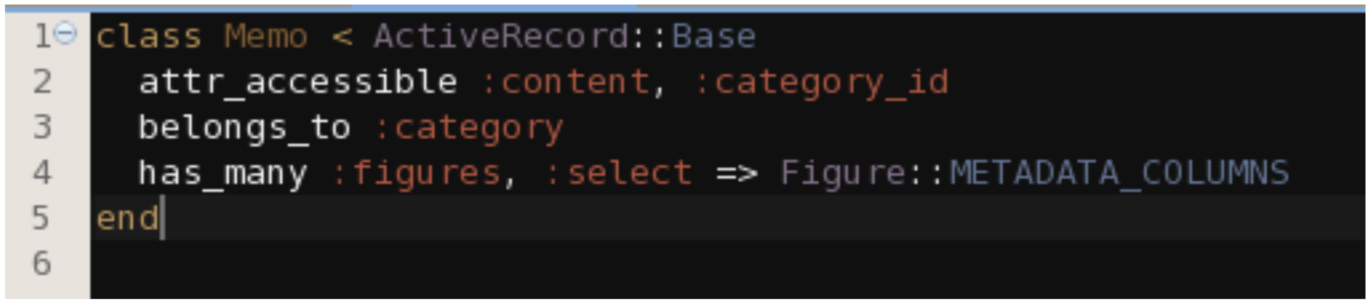


```
class Figure < ActiveRecord::Base
  attr_accessible :content, :file_name, :file_size,
          :file_type, :memo_id
  belongs_to :memo
  METADATA_COLUMNS = 'id, memo_id, file_name, file_size, file_type'

  def self.metadatas(memo)
    find :all, :select => METADATA_COLUMNS,
      :condition => ['memo_id = ?', memo.id]
  end

  def self.metadata(id)
    find id, :select => METADATA_COLUMNS
  end
end

```

Setting up Relations (2)

- Relationship from memos to figures is "has_many"
- Here through the figures attributes of Memo class, directly access to the figures' record, defined with METADATA_COLUMNS

```
has_many :figures, :select => Figure::METADATA_COLUMNS
```



```
class Memo < ActiveRecord::Base
  attr_accessible :content, :category_id
  belongs_to :category
  has_many :figures, :select => Figure::METADATA_COLUMNS
end
```

Add figure files, in memo creation

When a memo is created, figure files should be added to the memo.

Modify memos_controller.rb file at create method.

Method name with '?'

- respond_to?(:symbolName)

  The object is capable of responding to the caller when the “symbolName” method is called, this “respond_to?” method returns “true”. This means, that the object is installed with “symbolName” method.

app/controllers/memos_controller.rb

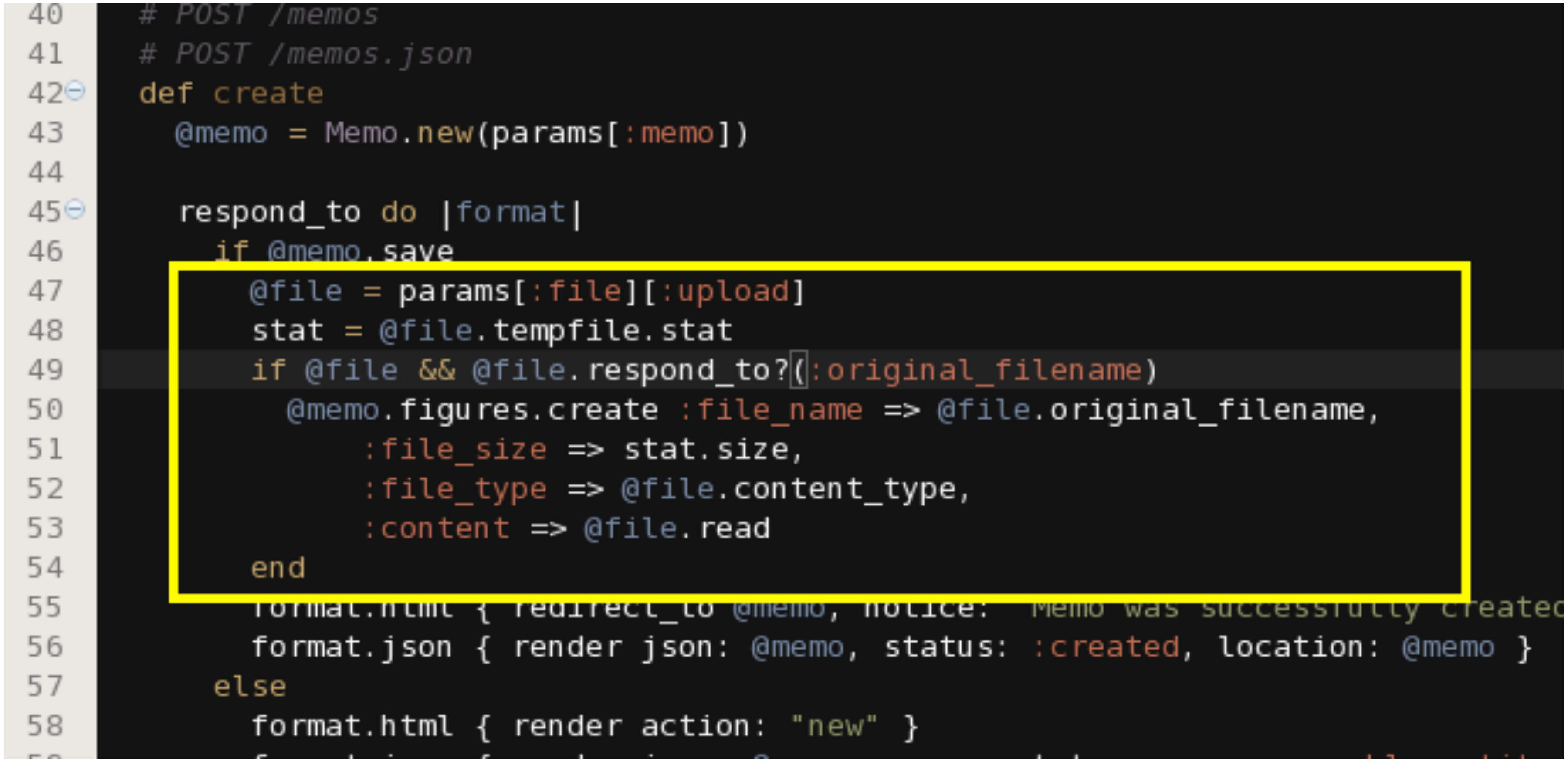
```
# POST /memos
# POST /memos.json
def create
  @memo = Memo.new(params[:memo])

  respond_to do |format|
    if @memo.save
      @file = params[:file][:upload]
      stat = @file.tempfile.stat
     if @file && @file.respond_to?(:original_filename)
        @memo.figures.create :file_name => @file.original_filenam
            :file_size => stat.size,
            :file_type => @file.content_type,
            :content => @file.read
      end
      format.html { redirect_to @memo, notice: 'Memo was successfully created.' }
      format.json { render json: @memo, status: :created, location: @memo }
(The rest is omitted.)
```

Add file generation
to the create method.
It will be executed when
there is an 'original_filename'
property

Note that those
names are specified
in the migration.

create method in memos_controller.rb

- The following lines are to be added.



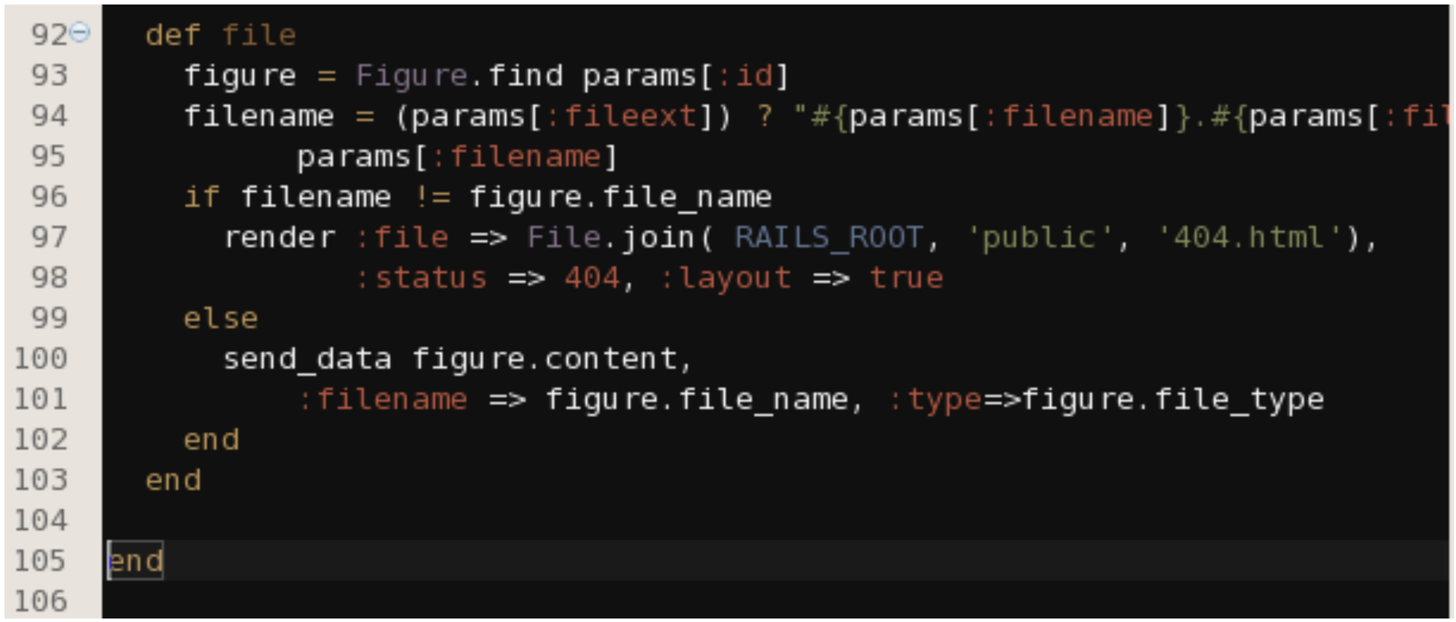
```
# POST /memos
# POST /memos.json
def create
  @memo = Memo.new(params[:memo])

  respond_to do |format|
    if @memo.save
      @file = params[:file][:upload]
      stat = @file.tempfile.stat
      if @file && @file.respond_to?(:original_filename)
        @memo.figures.create :file_name => @file.original_filename,
            :file_size => stat.size,
            :file_type => @file.content_type,
            :content => @file.read
      end
      format.html { redirect_to @memo, notice: 'Memo was successfully created
      format.json { render json: @memo, status: :created, location: @memo }
    else
      format.html { render action: "new" }
```

memos_controller.rb

- Add File method after the destroy method.

```
def file
  figure = Figure.find params[:id]
  filename = (params[:fileext]) ? "#{params[:filename]}.#{params[:fileext]}" :
        params[:filename]
  if filename != figure.file_name
    render :file => File.join( RAILS_ROOT, 'public', '404.html'),
           :status => 404, :layout => true
  else
    send_data figure.content,
        :filename => figure.file_name, :type=>figure.file_type
  end
end
```

File action, which is called for down loading images



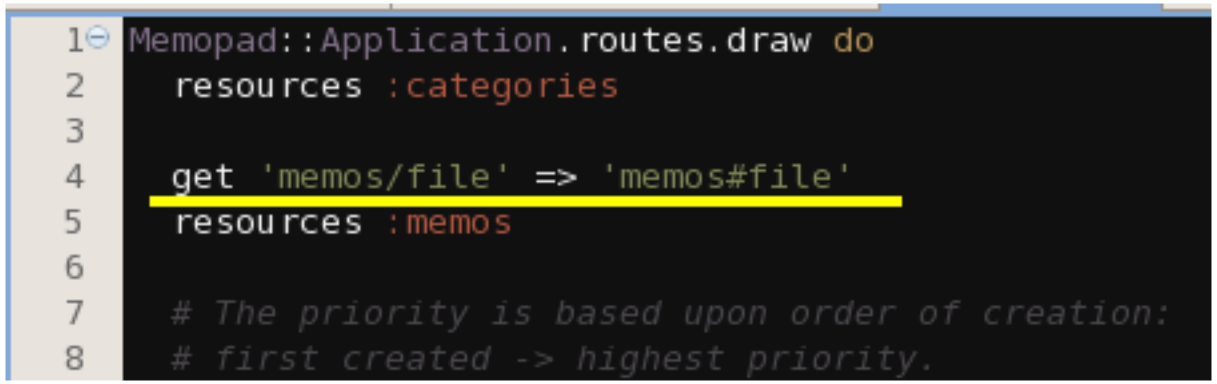
```
  def file
    figure = Figure.find params[:id]
    filename = (params[:fileext]) ? "#{params[:filename]}.#{params[:fi
          params[:filename]
    if filename != figure.file_name
      render :file => File.join( RAILS_ROOT, 'public', '404.html'),
             :status => 404, :layout => true
    else
      send_data figure.content,
          :filename => figure.file_name, :type=>figure.file_type
    end
  end

end
```

Add route for memos#file

Edit config/routes.rb

Before resources :memos, add

get ‘memos/file’ => ‘memos#file’



```
Memopad::Application.routes.draw do
  resources :categories

  get 'memos/file' => 'memos#file'
  resources :memos

  # The priority is based upon order of creation:
  # first created -> highest priority.
```

views modification

Controllers are modified, and then next few steps are to modify views image display files.

app/views/memos/_form.html.erb

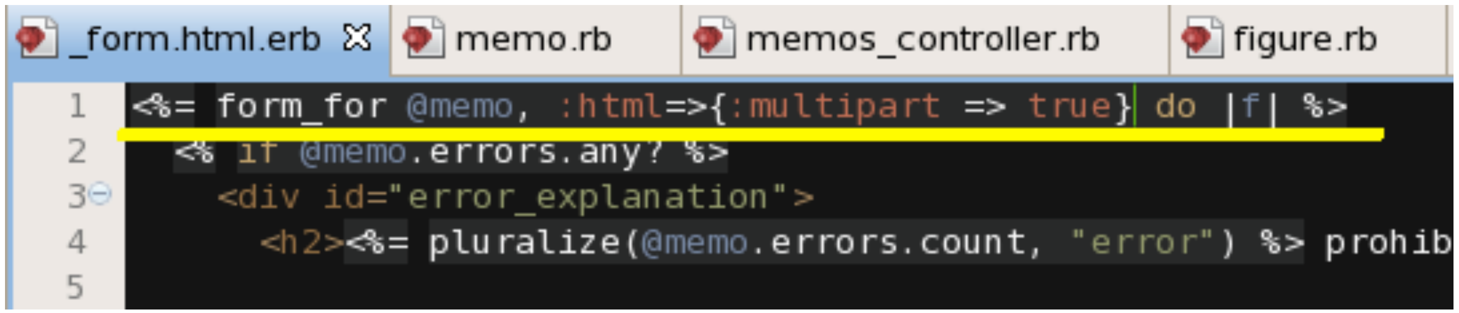
An image file generally has big size of binary information, and cannot be uploaded in only one packet.

So, set up the multi-packet transmission,

```
<%= form_for @memo, :html=>{:multipart => true} do |f| %>
```

where it used to be

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



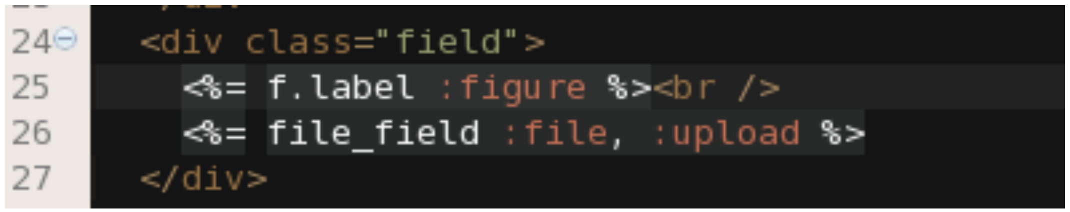
```
<%= form_for @memo, :html=>{:multipart => true} do |f| %>
  <% if @memo.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@memo.errors.count, "error") %> prohib
```

Input for file uploading

app/views/memos/_form.html.erb

```
•     <div class= “field” >
        <%= f. label :figure:</b>
        <%= file_field :file, :upload %>
      </div>
```

Uploaded file can be extracted by params[:file][:upload]



```
  <div class="field">
    <%= f.label :figure %><br />
    <%= file_field :file, :upload %>
  </div>
```

Prepare to show image file

- There are two types of modification.
- [Pattern 1]
  - To show the image itself.
- [Pattern 2]
  - Only show the link to the image file, so that users can download the file. (right-click)

Edit show.html.erb to show image.

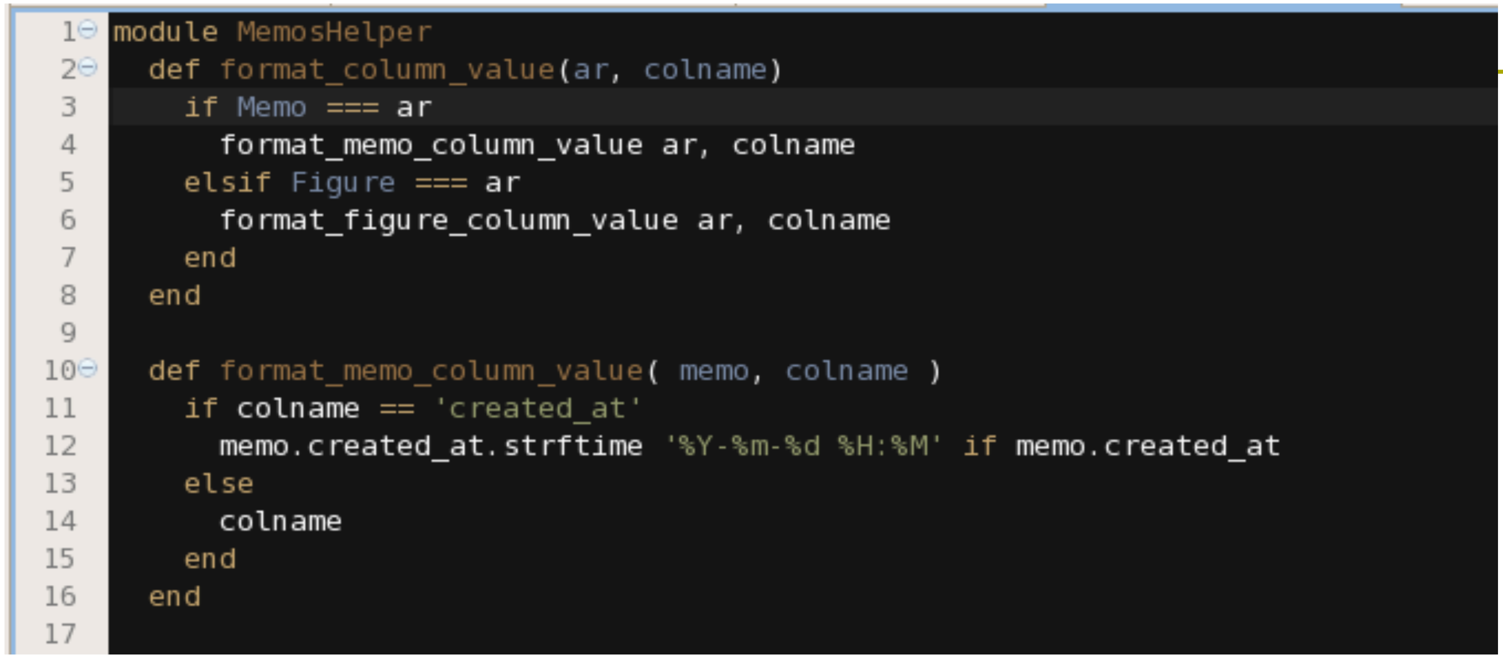
Add the Image file display in show.html.erb

To show the image file, use helper method.

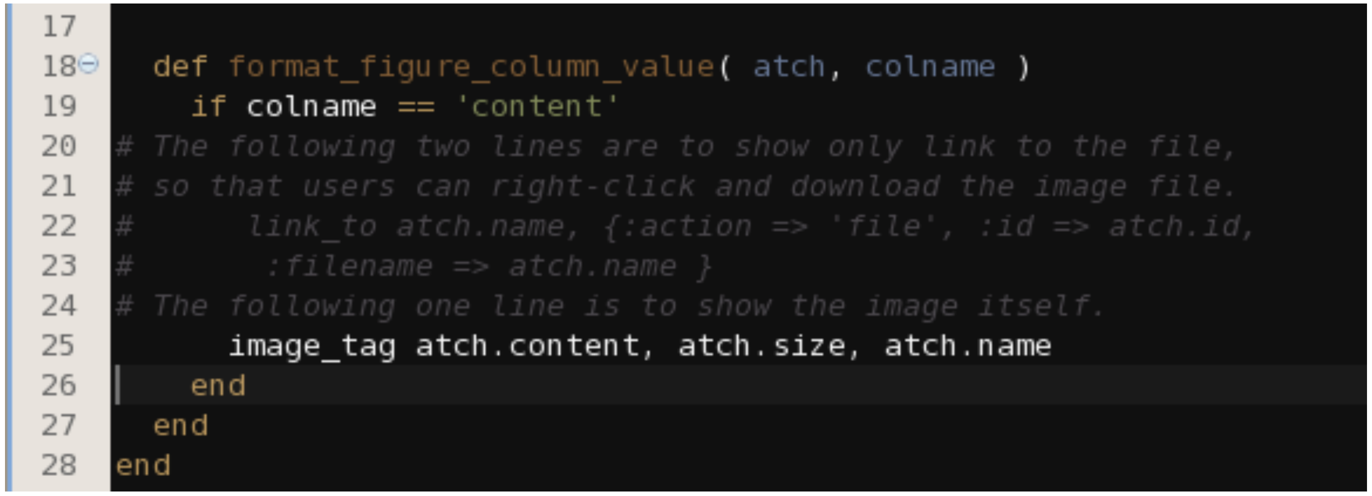
memos_helper.rb

```
module MemosHelper
 def format_column_value(ar, colname)
   if Memo === ar
     format_memo_column_value ar, colname
   elsif Figure === ar
     format_figure_column_value ar, colname
   end
  end

  def format_memo_column_value( memo, colname )
   if colname == 'created_at'
     memo.created_at.strftime '%Y-%m-%d %H:%M' if memo.created_at
   else
     colname
   end
  end

  def format_figure_column_value( atch, colname )
   if colname == 'content'
# The following two lines are to show only link to the file,
# so that users can right-click and download the image file.
#      link_to atch.name, {:action => 'file', :id => atch.id,
#       :filename => atch.name }
# The following one line is to show the image itself.
     image_tag atch.content, atch.size, atch.name
   end
  end
end
```

memos_helper.rb



```ruby
module MemosHelper
  def format_column_value(ar, colname)
    if Memo === ar
      format_memo_column_value ar, colname
    elsif Figure === ar
      format_figure_column_value ar, colname
    end
  end

  def format_memo_column_value( memo, colname )
    if colname == 'created_at'
      memo.created_at.strftime '%Y-%m-%d %H:%M' if memo.created_at
    else
      colname
    end
  end

```



```ruby

  def format_figure_column_value( atch, colname )
    if colname == 'content'
# The following two lines are to show only link to the file,
# so that users can right-click and download the image file.
#       link_to atch.name, {:action => 'file', :id => atch.id,
#         :filename => atch.name }
# The following one line is to show the image itself.
      image_tag atch.content, atch.size, atch.name
    end
  end
end
```

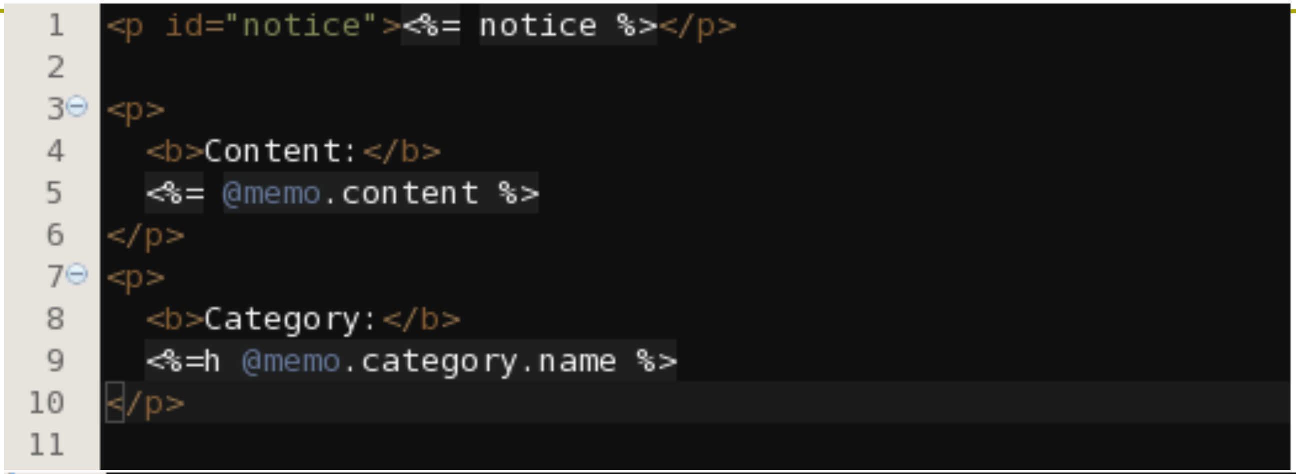
=== operator in memos_helper.rb

- In this helper method, look up the attribute of each column, and choose the method to show the content.
- The operator ‘===‘ is not common in other languages. If Figure === ar means “if the object ‘ar’ is an instance of Figure Class.”

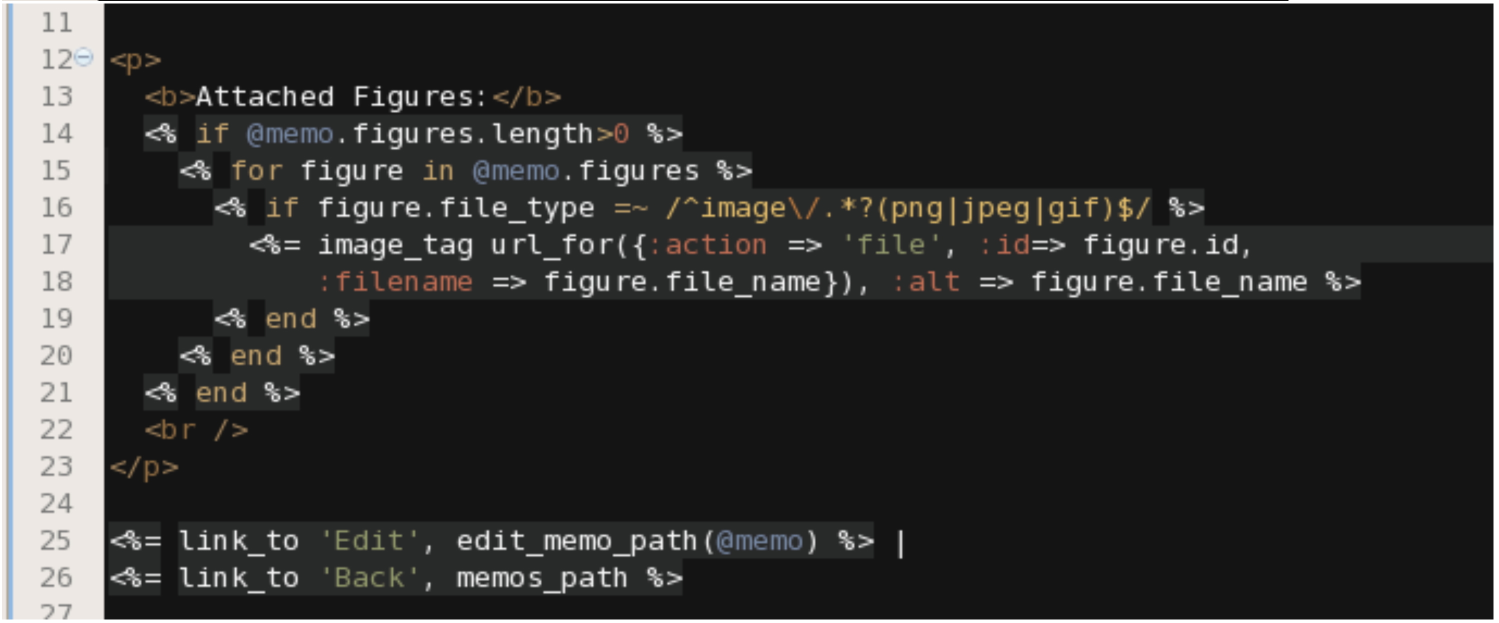
Edit show.html.erb

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

<p>
  <b>Content:</b>
  <%= @memo.content %>
</p>
<p>
  <b>Category:</b>
  <%=h @memo.category.name %>
</p>

<p>
  <b>Attached Figures:</b>
  <% if @memo.figures.length>0 %>
    <% for figure in @memo.figures %>
      <% if figure.file_type =~ /^image¥/.*?(png|jpeg|gif)$/ %>
        <%= image_tag url_for({:action => 'file', :id=> figure.id,
            :filename => figure.file_name}), :alt => figure.file_name %>
      <% end %>
    <% end %>
  <% end %>
  <br />
</p>

<%= link_to 'Edit', edit_memo_path(@memo) %> |
<%= link_to 'Back', memos_path %>
```

Show.html.erb file



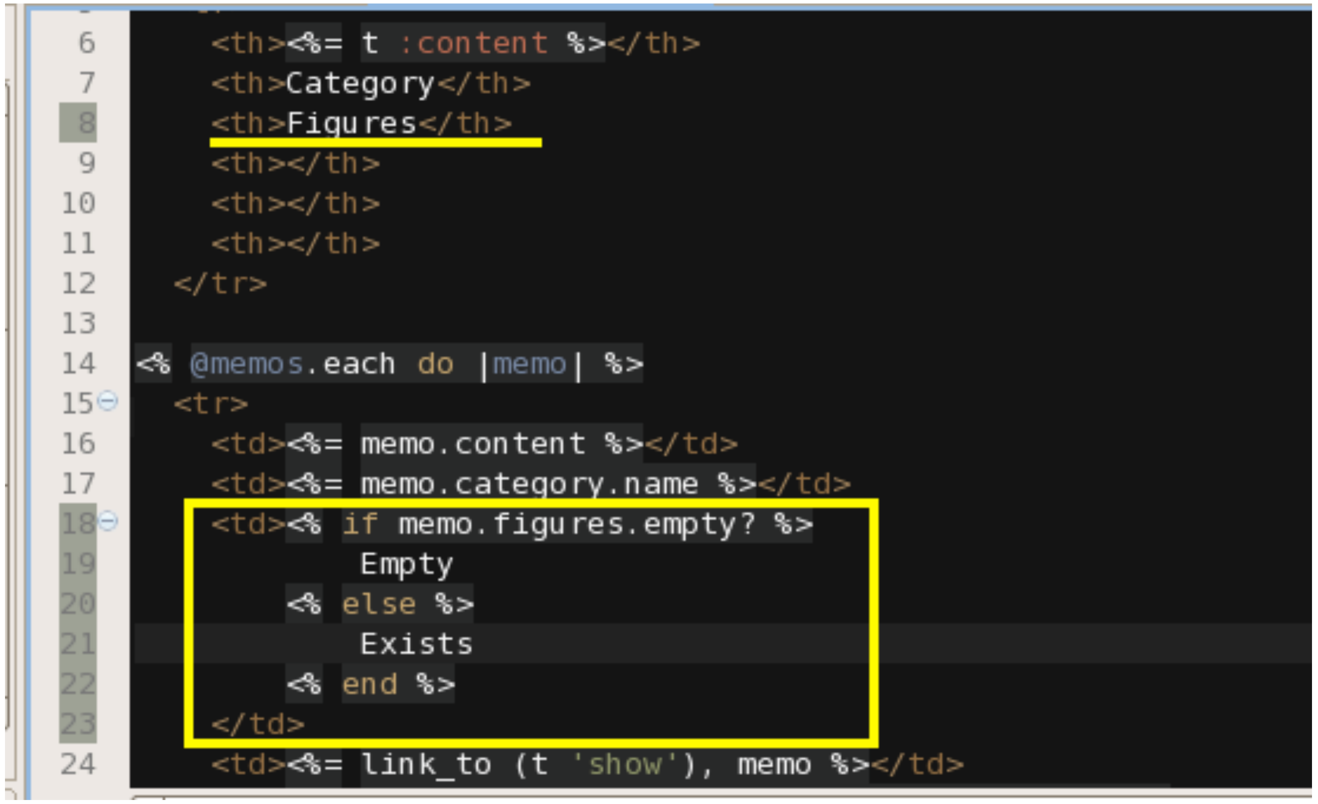
```
<p id="notice"><%= notice %></p>

<p>
  <b>Content:</b>
  <%= @memo.content %>
</p>
<p>
  <b>Category:</b>
  <%=h @memo.category.name %>
</p>

```



```

<p>
  <b>Attached Figures:</b>
  <% if @memo.figures.length>0 %>
    <% for figure in @memo.figures %>
      <% if figure.file_type =~ /^image\/.*?(png|jpeg|gif)$/ %>
        <%= image_tag url_for({:action => 'file', :id=> figure.id,
            :filename => figure.file_name}), :alt => figure.file_name %>
      <% end %>
    <% end %>
  <% end %>
  <br />
</p>

<%= link_to 'Edit', edit_memo_path(@memo) %> |
<%= link_to 'Back', memos_path %>
```

index.html.erb

- For debugging purpose, in index.html.erb, let the program shows only figures are attached or not.



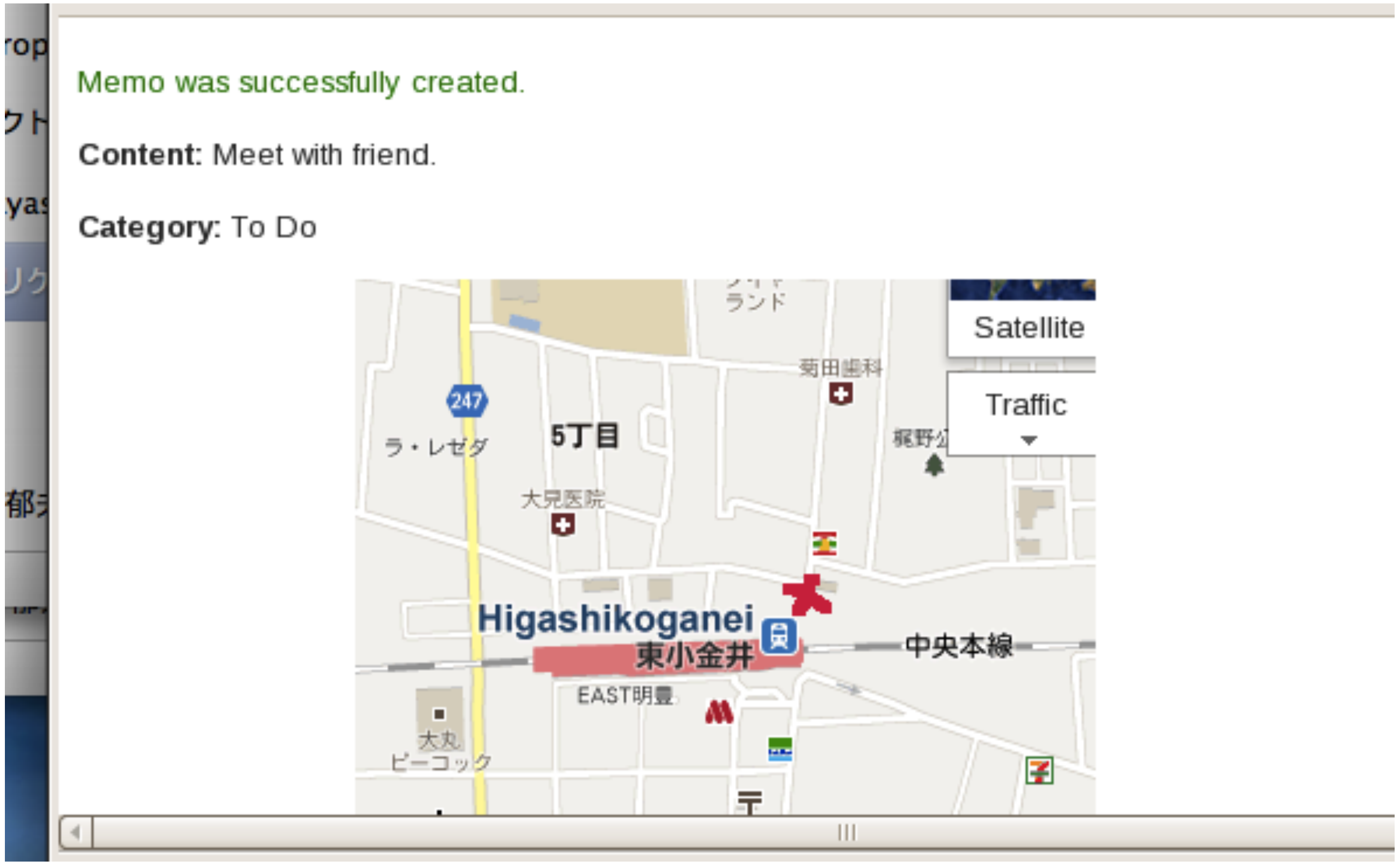
```
    <th><%= t :content %></th>
    <th>Category</th>
    <th>Figures</th>
    <th></th>
    <th></th>
    <th></th>
  </tr>

<% @memos.each do |memo| %>
  <tr>
    <td><%= memo.content %></td>
    <td><%= memo.category.name %></td>
    <td><% if memo.figures.empty? %>
            Empty
        <% else %>
            Exists
        <% end %>
    </td>
    <td><%= link_to (t 'show'), memo %></td>
```

(Table part) index.html.erb

```
<table border="1">
  <tr>
    <th><%= t :content %></th>
    <th>Category</th>
    <th>Figures</th>
    <th></th>
    <th></th>
    <th></th>
  </tr>

<% @memos.each do |memo| %>
  <tr>
    <td><%= memo.content %></td>
    <td><%= memo.category.name %></td>
    <td><% if memo.figures.empty? %>
          Empty
        <% else %>
          Exists
        <% end %>
    </td>
    <td><%= link_to (t 'show'), memo %></td>
    <td><%= link_to (t 'edit'), edit_memo_path(memo) %></td>
    <td><%= link_to (t 'destroy'), memo, method: :delete, data: { confirm: 'Are you sure?' } %></td>
  </tr>
<% end %>
</table>
```

Test run

Confirm that we can upload image.



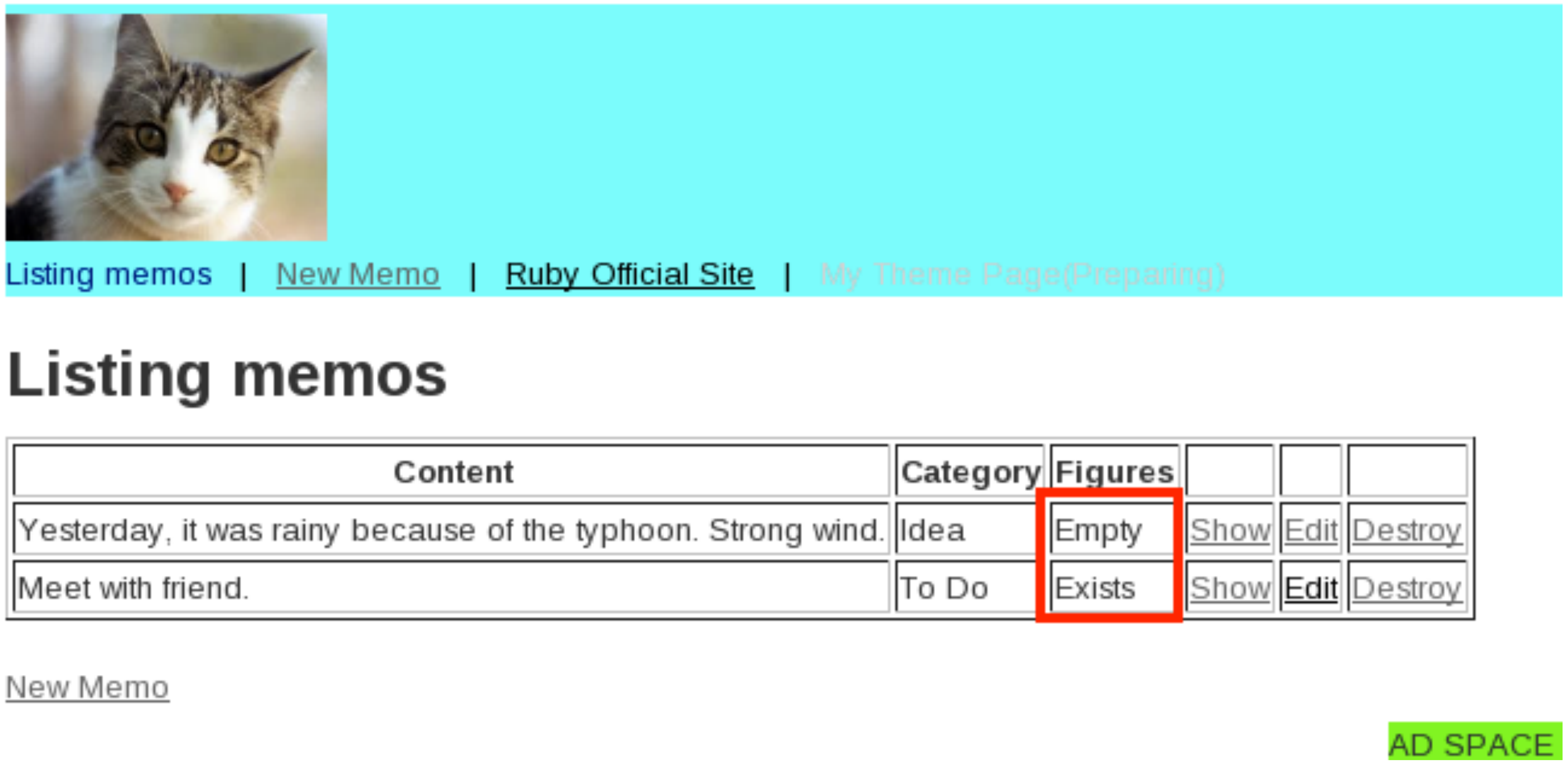
Memo was successfully created.

Content: Meet with friend.

Category: To Do

Check the list screen

- Confirm that we can see if an attached file is empty or exists.



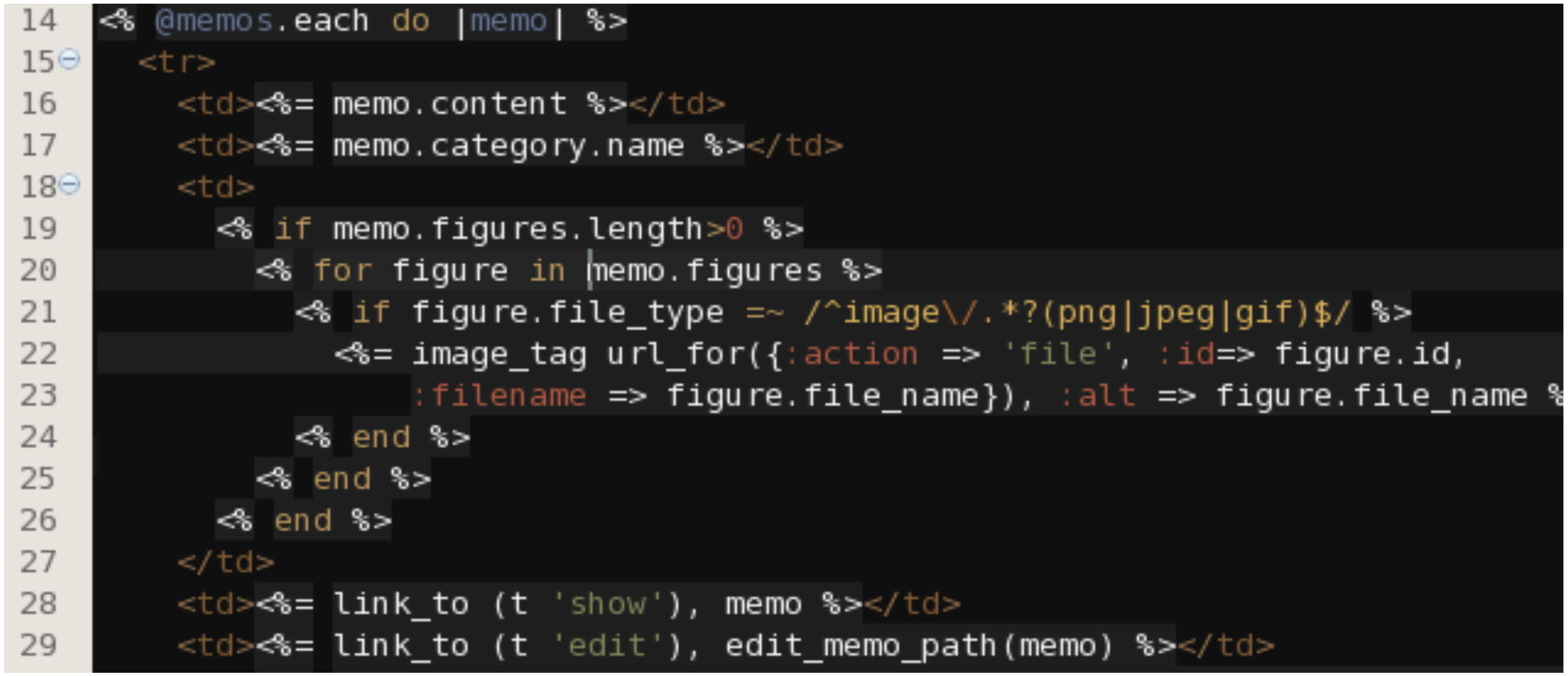
Listing memos | New Memo | Ruby Official Site | My Theme Page(Preparing)

Listing memos

Content	Category	Figures			
Yesterday, it was rainy because of the typhoon. Strong wind.	Idea	Empty	Show	Edit	Destroy
Meet with friend.	To Do	Exists	Show	Edit	Destroy

Final modification for index.html.erb

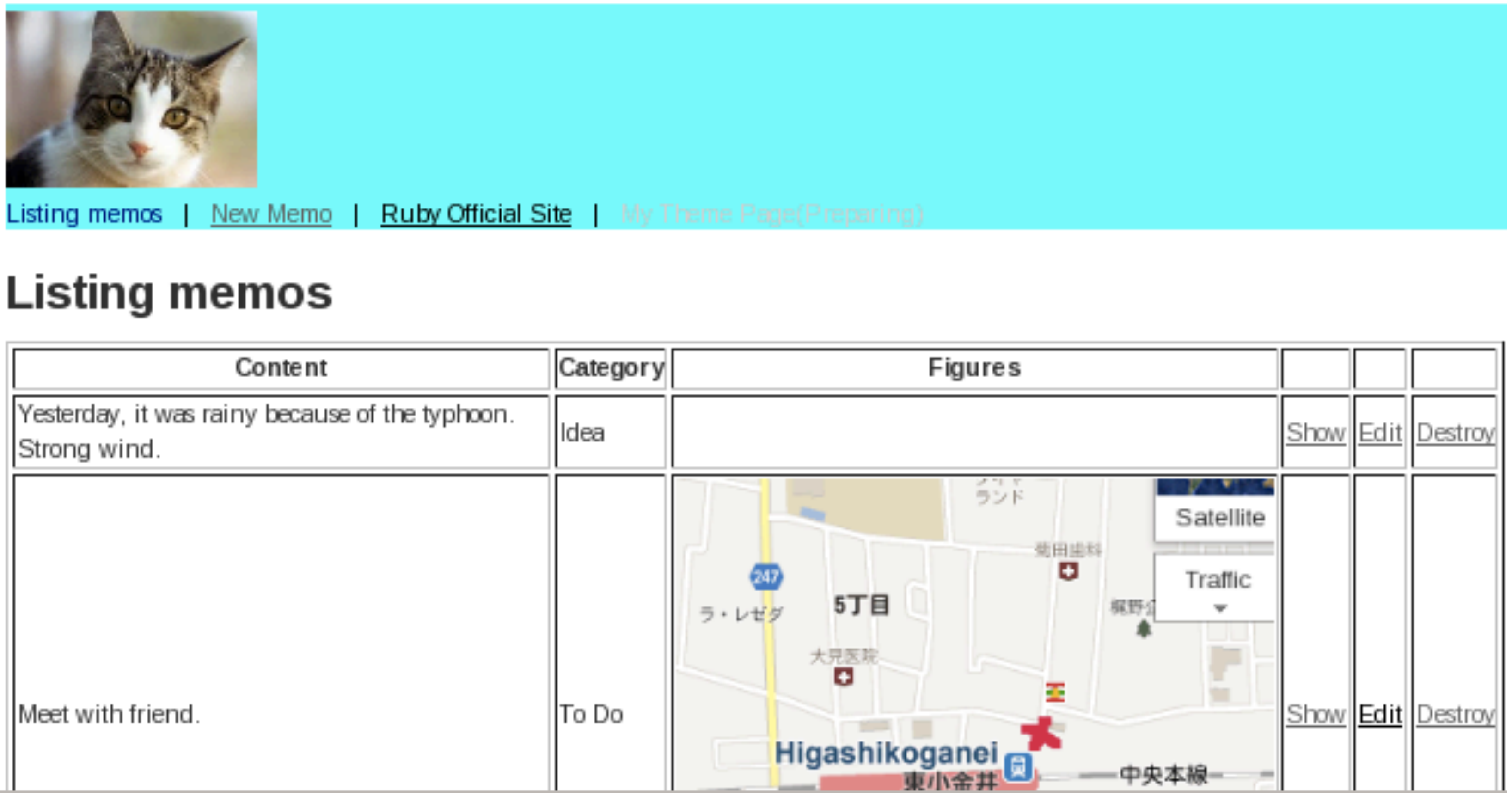
Replace the “figures empty/exist” text part with the following image display.



```
<% @memos.each do |memo| %>
  <tr>
    <td><%= memo.content %></td>
    <td><%= memo.category.name %></td>
    <td>
      <% if memo.figures.length>0 %>
        <% for figure in memo.figures %>
          <% if figure.file_type =~ /^image\/.*?(png|jpeg|gif)$/ %>
            <%= image_tag url_for({:action => 'file', :id=> figure.id,
                :filename => figure.file_name}), :alt => figure.file_name %
          <% end %>
        <% end %>
      <% end %>
    </td>
    <td><%= link_to (t 'show'), memo %></td>
    <td><%= link_to (t 'edit'), edit_memo_path(memo) %></td>
```

Today's final screen.

Finally, we can see figures with memo.



Listing memos | New Memo | Ruby Official Site | My Theme Page(Preparing)

Listing memos

Content	Category	Figures			
Yesterday, it was rainy because of the typhoon. Strong wind.	Idea		Show	Edit	Destroy
Meet with friend.	To Do	Satellite Traffic 247 5丁目 ラ・レゼダ Higashikoganei 東小金井 中央本線	Show	Edit	Destroy

Practice

No report is requested, however, try to fix the following problems.

(1) When we destroy a memo record, the linked figures left undestroyed. Add some program to the destroy method in the memos controller.

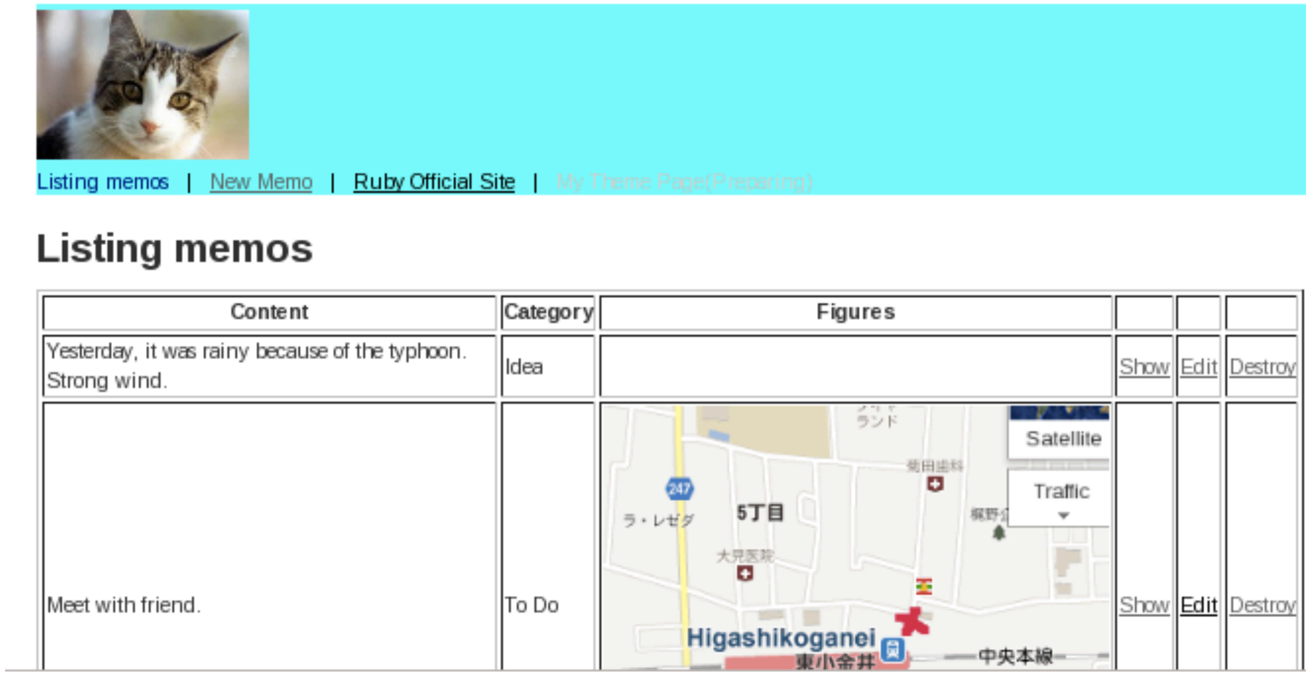
(2) The relationship between memo and figures is one-to-many. But, we do not have a control logic to add/remove attached figures. Try this.

The answer for today’s practice

- Next week, in the session control lecture, I will show my program as one answer for the problems listed in the previous slide.
- But, as graduate school students, I hope you could solve this problem by yourselves.

Absence Report for Today

- Submit the report of screenshots, to show that you could upload image file to memo, just like the following.



Listing memos | New Memo | Ruby Official Site | My Theme Page(Preparing)

Listing memos

Content	Category	Figures			
Yesterday, it was rainy because of the typhoon. Strong wind.	Idea		Show	Edit	Destroy
Meet with friend.	To Do		Show	Edit	Destroy