Object Oriented WEB Programming Using Ruby

Day 11: 28/June/2012

Upload and Download Images

Today's Goal

- Upload images as evidences of facts, or the explanation materials of solutions, and such.
- Arrange one to many relations from causes table to images table.

Table Design for Images

Name of the class: CaseAttachment

table name: cause_attachments

One ‘cause_attachments’ record belongs to one cause.

But one cause may have many images, so, relation is ‘one-to-many.’

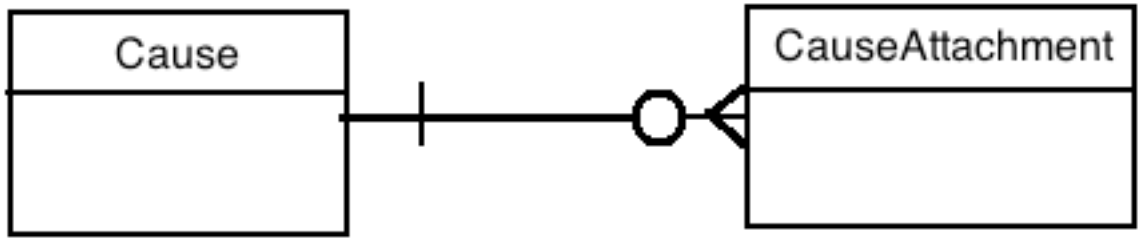
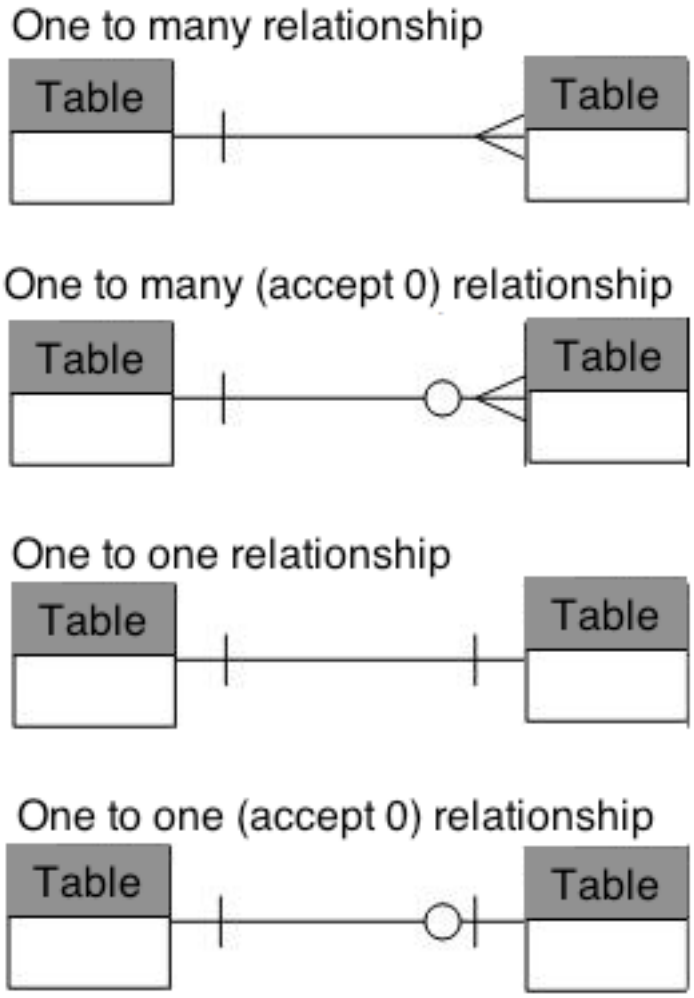


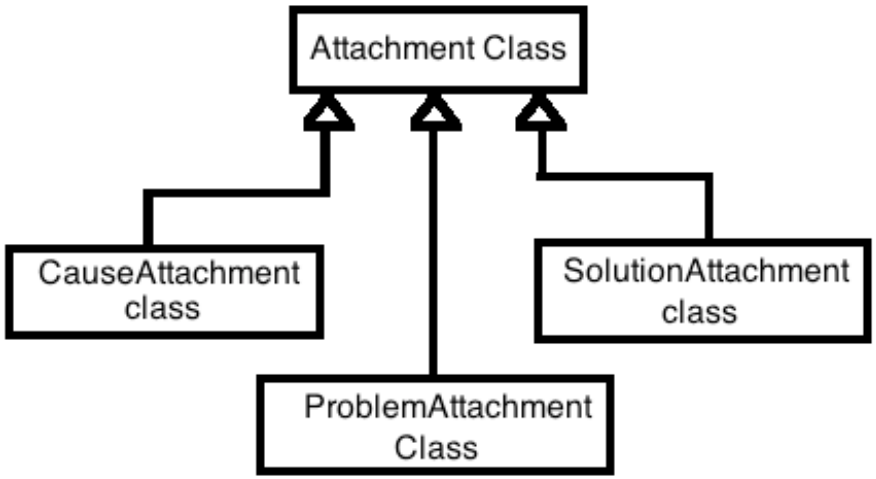
Table Design diagrams

When we use UML(Unified Modeling Language,) relationship is described as shown in the right.



Attachment Class and sub classes

Behavior of Attachment class instances should be defined in only one place, but relations are linked to different tables. In order to let it be the 'DRY' design, we let each 'attachments' table inherit one Attachment class. (Initial design…)



Not DRY yet…

Because of coding problems… I gave up designing the inheritance from Attachment class.

Structure is not DRY yet, but we use CauseAttachment directly. Because of this design problem, if we need to attach images to a Solution table, we may repeat the same kind of coding to SolutionAttachment table.

Generate CauseAttachment Class

CauseAttachment Class should have

cause_id, integer (relation index to Cause)

name, string, (filename, size 255)

size, integer (image file size in bytes)

type, string (MIME type name, size 32)

content, blob (content of image file)

BLOB: Binary Large OBject. Field for image, sound and such ‘raw’ binary data.

MIME Type

What is MIME?

Multipurpose Internet Mail Extension

Such as:

text/css

image/jpeg

image/png

application/x-internet-signup

(and more than 571 types are used..)

http://www.geocities.co.jp/Hollywood/9752/mime.html

Data Type of SQlite3

Type	Contents
NULL	Null Value
INTEGER	Signed Integer Value in 1, 2, 3, 4, 6, or 8 bytes
REAL	Floating point numerical value, in 8 bytes
TEXT	Text string in UTF-8, UTF-16BE, or UTF-16-LE
BLOB	Raw image of binary large object

Such data type as ‘String’ is finally mapped to Text type of Sqlite3, but for convenience and/or the compatibility, Sqlite 3 accepts other type descriptions like ‘String’ and such.

Generate CauseAttachment class

Type the following command (in one line):

rails generate model CauseAttachment cause_id:integer name:string size:integer content_type:string content:binary

And then type:

rake db:migrate

```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ rails generate model CauseAttachment cause_id:intege
r name:string size:integer content_type:string content:binary
      invoke  active_record
      create    db/migrate/20120628003129_create_cause_attachments.rb
      create    app/models/cause_attachment.rb
      invoke    test_unit
      create      test/unit/cause_attachment_test.rb
      create      test/fixtures/cause_attachments.yml
kobayashi-ikuo-no-MacBook:spielberg kobayashi$
```

One of the error cause

First I had put the column name ‘type’ instead of ‘content_type,’ because the name is shorter. It had caused the following error.

Just for your information.

ActiveRecord::SubclassNotFound in Causes#show

Showing */Users/kobayashi/Aptana3Work/spielberg/app/views/causes/show.html.erb* where line **#18** raised:

```
The single-table inheritance mechanism failed to locate the subclass: 'image/png'. This error is raised because the column 'type' is
reserved for storing the class in case of inheritance. Please rename this column if you didn't intend it to be used for storing the
inheritance class or overwrite CauseAttachment.inheritance_column to use another column for that information.
```

Set Relationship

Set

belongs_to :cause

in the cause_attachment.rb file, and set

has_many :cause_attachments

in the cause.rb file.

```ruby
class CauseAttachment < ActiveRecord::Base
  attr_accessible :cause_id, :content, :content_type, :name, :size
  belongs_to :cause
end
```

```ruby
class Cause < ActiveRecord::Base
  attr_accessible :cons, :fact, :pros, :problem, :problem_id
  belongs_to :problem
  has_many :votes
  has_many :cause_attachments
end
```

_form.html.erb File (1)

Now we add the uploading area to 'Cause' data create screen, in _form.html.erb.

Because the image file is too big, we can not send the contents all together in only one transmission. So, we allow HTML to send in multi packets transmission.

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

```
1  <%= form_for @cause, :html => {:multipart => true} do |f| %>
2    <% if @cause.errors.any? %>
3      <div id="error_explanation">
4        <h2><%= pluralize(@cause.errors.count, "error") %> prohibited thi
5
6        <ul>
```

_form.html.erb File (2)

To upload the file, add the following in _form.html.erb file.

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

```
14  <div class="field">
15    <%= f.label :fact %><br />
16    <%= f.text_area :fact, :size => "60x10" %>
17  </div>
18  <div class="field">
19    <%= f.label :attachment_file %><br />
20    <%= file_field :file, :upload %>
21  </div>
22  <%= f.hidden_field :problem_id, :value => @cause.problem.id %>
23  <div class="actions">
24    <%= f.submit %>
25  </div>
26<% end %>
27
```

create method in cause_controller.rb

We add the following lines to cause_controller.rb in create method;

```
if params[:file]
  @file = params[:file][:upload]
  if @file && @file.respond_to?(:original_filename)
    stat = @file.tempfile.stat
    @cause.cause_attachments.create :cause_id => @cause.id,
        :name => @file.original_filename,
        :size => stat.size,
        :content_type => @file.content_type,
        :content => @file.read
  end
end
```

This is only performed only when It responds to original_filename property.

Create method of Cause action

```
# POST /causes
 # POST /causes.json
 def create
   @cause = Cause.new(params[:cause])
   @cause.pros = 0
   @cause.cons = 0
   @problem = Problem.find(params[:cause][:problem_id])
   @cause.problem_id = @problem.id

  respond_to do |format|
    if @cause.save
      if params[:file]
        @file = params[:file][:upload]
        if @file && @file.respond_to?(:original_filename)
          stat = @file.tempfile.stat
          @cause.cause_attachments.create :cause_id => @cause.id,
              :name => @file.original_filename,
              :size => stat.size,
              :content_type => @file.content_type,
              :content => @file.read
        end
      end
      format.html { redirect_to @cause, :notice => 'Cause was successfully created.' }
      format.json { render :json => @cause, :status => :created, :location => @cause }
    else
      format.html { render :action => "new" }
      format.json { render :json => @cause.errors, :status => :unprocessable_entity }
    end
  end
 end
```

Create method of Cause Action

```
74  # POST /causes
75  # POST /causes.json
76  def create
77    @cause = Cause.new(params[:cause])
78    @cause.pros = 0
79    @cause.cons = 0
80    @problem = Problem.find(params[:cause][:problem_id])
81    @cause.problem_id = @problem.id
82
83    respond_to do |format|
84      if @cause.save
85        if params[:file]
86          @file = params[:file][:upload]
87          if @file && @file.respond_to?(:original_filename)
88            stat = @file.tempfile.stat
89            @cause.cause_attachments.create :cause_id => @cause.id,
90                :name => @file.original_filename,
91                :size => stat.size,
92                :content_type => @file.content_type,
93                :content => @file.read
94          end
95        end
96        format.html { redirect_to @cause, :notice => 'Cause was successfully created.' }
97        format.json { render :json => @cause, :status => :created, :location => @cause }
98      else
99        format.html { render :action => "new" }
100       format.json { render :json => @cause.errors, :status => :unprocessable_entity }
101     end
102   end
103 end
```

Method Name “respond_to?”

‘?’ is a part of method name ‘respond_to?’

There are also the method names which have ‘!’ in the name.

Most methods with ‘?’ at the end of the name, respond boolean value; true or false, such as ‘exist?’, and ‘matched?’

Most methods with ‘!’ implies the meaning that they performs even if there are slight problems in the executing environment.

One bug from the last time

When we click ‘back’ at the following screen…

ActiveRecord Error occurred,

At the ‘back’ link

ActiveRecord::RecordNotFound in CausesController#index

```
Couldn't find Problem without an ID
```

Rails.root: /Users/kobayashi/Aptana3Work/spielberg

Application Trace | Framework Trace | Full Trace

```
app/controllers/causes_controller.rb:6:in `index'
```

Request

Parameters:

The reason for this error…

Causes all belong to one problem, so, all the view screen requires the problem_id to show (select) the related problem. (Thus we had modified.)

But in some screen, we have not programmed to hand ‘problem_id’ yet, so this error message appears.

By assigning @problem in every method, this error should dispear…

After destroying the record too..

The same reason causes another error.

One of the point to fix:

I forgot to put these two lines at the point of creation, place

```
@problem = Problem.find(params[:cause][:problem_id])
@cause.problem_id = @problem.id
```

It might be defiant to say, though

Well, the 'perfect material' will easily lead you to the result fruits, but rob you of the chance to learn how to trouble-shoot!

So, I will give you chances to trouble-shoot!

At the initialization, assign the parameter.

At the reference, read and hand the parameter.

Link from ‘show’ to ‘index’

Please modify like this:

```
<%= link_to ('Back', { :controller=>"causes", :action=>"index",
    :problem_id => @cause.problem.id} )  %>
```

```
12
13  <p>
14    <b>Cons:</b>
15    <%= @cause.cons %>
16  </p>
17
18
19  <%= link_to 'Edit', edit_cause_path(@cause) %> |
20  <%= link_to ('Back', { :controller=>"causes", :action=>"index",
21          :problem_id => @cause.problem.id} )  %>
22
```

Now it is almost the time of the class…

This week again, time is almost up again.

The following slides are not completed yet, by running on Rails 3.2 environment, and for this semesters' topics; Problem Solving Engine.

Some slides are untouched from the program for the environment of last year, Rails 2.

I think you may need to rewrite/ reform the codes.

file method in cause_controller.rb

Add the following lines to cause_controller.rb as file method.

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

File action is used when images are downloaded

causes_helper.rb

```
module CausesHelper
  def format_column_value(ar, colname)
    if Cause === ar
      format_cause_column_value ar, colname
    elsif CauseAttachment === ar
      format_attachment_column_value ar, colname
    end
  end

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

  def format_attachment_column_value( atch, colname )
    if colname == 'content'
# 以下の2行は、showの画面では画像は表示せず、ダウンロードする形式
#       link_to atch.name, {:action => 'file', :id => atch.id,
#         :filename => atch.name }
# 以下の1行は、showの画面で画像を表示する形式
      image_tag atch.content, atch.size, atch.name
    else
      atch.send( colname )
    end
  end
end
```

Edit show.html.erb

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

<h3>As a Cause of the problem: <%= @cause.problem.title %>

<p>
  <b>Fact:</b>
  <%= @cause.fact %>
</p>

<p>
  <b>Pros:</b>
  <%= @cause.pros %>
  <b>Cons:</b>
  <%= @cause.cons %>
</p>

<p>
  <% if @cause.cause_attachments.length>0 %>
    <b>Attachments</b>
    <table border="1">
      <tr>
          <% for column in @cause.cause_attachments.content_columns %>
          <th><%= column.human_name %></th>
        <% end %>
      </tr>
      <% for attachment in @cause.cause_attachments %>
        <tr>
              <% for column in @cause.cause_attachments.content_columns %>
          <% if column.name == 'content' &&
                attachment.content_type =~ /^image¥/.*?(png|jpeg|gif)$/ %>
            <td><%= image_tag url_for({:action => 'file', :id=> attachment.id,
                  :filename => attachment.name}), :alt => attachment.name %></td>
          <% else %>
            <td><%= format_column_value attachment, column.name %></td>
          <% end %>
        <% end %>
        </tr>
      <% end %>
    </table>
  <% end %>
</p>

<%= link_to 'Edit', edit_cause_path(@cause) %> |
<%= link_to ('Back', { :controller=>"causes", :action=>"index",
            :problem_id => @cause.problem.id} )  %>
```

```
14    <%= @cause.cons %>
15  </p>
16
17  <p>
18    <% if @cause.cause_attachments.length>0 %>
19      <b>Attachments</b>
20      <table border="1">
21        <tr>
22          <% for column in @cause.cause_attachments.content_columns %>
23            <th><%= column.human_name %></th>
24          <% end %>
25        </tr>
26        <% for attachment in @cause.cause_attachments %>
27          <tr>
28            <% for column in @cause.cause_attachments.content_columns %>
29              <% if column.name == 'content' &&
30                      attachment.content_type =~ /^image\/.*?(png|jpeg|gif)$/ %>
31                <td><%= image_tag url_for({:action => 'file', :id=> attachment.id,
32                        :filename => attachment.name}), :alt => attachment.name %></td>
33              <% else %>
34                <td><%= format_column_value attachment, column.name %></td>
35              <% end %>
36            <% end %>
37          </tr>
38        <% end %>
39      </table>
40    <% end %>
41  </p>
42
```

causes_helper.rb

- According to the column (attribute) type, switch the display format in ‘show’ causes view.
- Unusual (in other languages) operator === returns true, when Cause === ar, and “object ar is an instance of Cause class.”
- There are two sample codes displayed in the previous slide, to show images in the view or to show only links to image display.
  - Commented out with # letter.

Screen image at this point

Now we have reached to show the attachment contents from the causes link.

TOP | Register new Problem | ruby Official Site | My Twitter(Not Read

As a Cause of the problem: I cannot remove this bug.

Fact: File is attached in this record.

Pros: 0 Cons: 0

Attachments

Name	Size	Content type	Content	Created at	Updated at
gazou4.png	66982	image/png	gazou4.png	2012-06-28 00:56:04 UTC	2012-06-28 00:56:04 UTC

Edit | **Back**

Our expectation

- The screen shots below is the previous year result with Rails 2.x.
- But we failed with Rails 3.x environment.
- Why?

Assets directory

Asset directory is the default directory where images are stored.

It seems rails try to find the image file with ‘img_tag’ description.

```
Started GET "/assets?action=file&controller=causes&filename=gazou4.png&id=4" for 127.0.0.1
un 28 09:58:25 +0900 2012
Served asset  - 404 Not Found (5ms)

ActionController::RoutingError (No route matches [GET] "/assets"):
  actionpack (3.2.3) lib/action_dispatch/middleware/debug_exceptions.rb:21:in `call'
  actionpack (3.2.3) lib/action_dispatch/middleware/show_exceptions.rb:56:in `call'
  railties (3.2.3) lib/rails/rack/logger.rb:26:in `call_app'
  railties (3.2.3) lib/rails/rack/logger.rb:16:in `call'
  actionpack (3.2.3) lib/action_dispatch/middleware/request_id.rb:22:in `call'
  rack (1.4.1) lib/rack/methodoverride.rb:21:in `call'
  rack (1.4.1) lib/rack/runtime.rb:17:in `call'
  activesupport (3.2.3) lib/active_support/cache/strategy/local_cache.rb:72:in `call'
  rack (1.4.1) lib/rack/lock.rb:15:in `call'
```

Here is a hint from the last year

- Modify config\routes.rb

- The following the indication for Rails 2.x.
- We have not checked yet, but apparently, we face with the routing problem.

```
18   map.connect ':controller/service.wsdl', :action => 'wsdl'
19
20   # Install the default route as the lowest priority.
21   map.connect ':controller/:action/:id.:filename.:fileext'
22   map.connect ':controller/:action/:id.:filename'
23   map.connect ':controller/:action/:id.:format'
24   map.connect ':controller/:action/:id'
25 end
```

Now it is completely time up.

We have almost reached to today's planed goal, but we failed at the very few steps before the goal.

Please go forward for the last step by yourself.

I am not sure if it were a proper word, but, "Good Luck!"

Today's Report Topics

Yukihiro Matsumoto says that 'multiple-inheritance' is not always bad. In the following Japanese article.

http://itpro.nikkeibp.co.jp/article/COLUMN/20070828/280575/

You can write today's report either in Japanese or in English, paying respect to the Ruby Language Inventor.

Please discuss the design I have shown today to install Attachments Class for three classes, Causes, Solutions, and Problems.

Please discuss the 'should-be' design regarding the inheritance on the practical standpoint.

View Point of today's Topics

If we use 'inheritance' in a 'good manner,' it will absolutely reduce codes(, and our labor) dramatically.

But if once we miss design the characteristics of the language, and the installation environment, 'Inheritance' may be the cause of complexity, and increase our labor cost to maintain the source program.

I was wondering if I should have used the Attachment table as the parent of CauseAttachment, SolutionAttachment, and such.

And once again, as the result of 'time-out' I decided to use CauseAttachment class without inheriting Attachment common class. But I was not certain that this was the best answer of the installation. So, please you students judge!

Prepare for the Next Week

The lecture plan for the next week is Session Management.