Object Oriented WEB Programming Using Ruby

Day 13: 11/July/2012

Message and Mail Delivery

Plan on Last Week

We discuss the 'extension' of our project, "Problem Solving Engine."

If we allow the participants of the system to debate on certain topics, what kind of design could be possible?

If we allow the chairman of certain topics to control the discussion, what kind of design and screen could be possible?

Let us think about all of these WEB system design, based on Object Oriented Characteristics.

To do first…

We will install the mail transmission.

And then, think about the extension of the Problem solving engine.

Today's Goal

- We can send mails to certain users.
- Then finish the design of the system.

ActiveMailer

Use ActionMailer: Modify

config/environments/development.rb

Replace false with true at line #17, then Add ActionMailer settings;

```
# Don't care if the mailer can't send
config.action_mailer.raise_delivery_errors = true

ActionMailer::Base.delivery_method = :smtp
ActionMailer::Base.smtp_settings = {
    :address => "mail.smtp.server",
    :port => 587,
    :authentication => :login,
    :user_name => "mailaddress@domain.jp",
    :password => "password",
    :domain => "domain.jp"
}
```

config/environments/development.rb

Some settings will be explained in the class room.

development.rb

```
15
16  # Don't care if the mailer can't send
17  config.action_mailer.raise_delivery_errors = true
18
19  ActionMailer::Base.delivery_method = :smtp
20  ActionMailer::Base.smtp_settings = {
21    :address => "[redacted].hosei.ac.jp",
22    :port => [redacted],
23    :authentication => :login,
24    :user_name => "ikuo.kobayash[redacted]osei.ac.jp",
25    :password => "[redacted]",
26    :domain => "hosei.ac.jp"
27  }
28
29  # Print deprecation notices to the Rails logger
30  config.active_support.deprecation = :log
31
```

Generation of Mailer Class

Let us use Msend class for mail transmission. The method name should be simple_send for this sample.

Type the following command;

rails generate mailer msend simple_send

```
kobayashi-ikuo-no-MacBook:chirpy kobayashi$ rails generate mailer msend simple_send
      create  app/mailers/msend.rb
      invoke  erb
      create    app/views/msend
      create    app/views/msend/simple_send.text.erb
      invoke  test_unit
      create    test/functional/msend_test.rb
kobayashi-ikuo-no-MacBook:chirpy kobayashi$
```

Msend Class

```
class Msend < ActionMailer::Base
  default :from => ”ikuo.kobayashi.XXXX@example.com",
          :return_path => “kobayashi@XXXXXX.co.jp”

  # Subject can be set in your I18n file at config/locales/en.yml
  # with the following lookup:
  #
  #   en.msend.simple_send.subject
  #
  def simple_send( recipient )
    @greeting = "Hi”

    # mail( :to => recipient, :subject => “Hello Mailer” )
    mail(:to => recipient ) do |format|
      format.text
      format.html
    end
  end
end
```

See ActionMail Document

It is best to check it at api.rubyonrails.org.

If we want to explicitly render only certain templates, pass a block:

```
mail(:to => user.email) do |format|
  format.text
  format.html
end
```

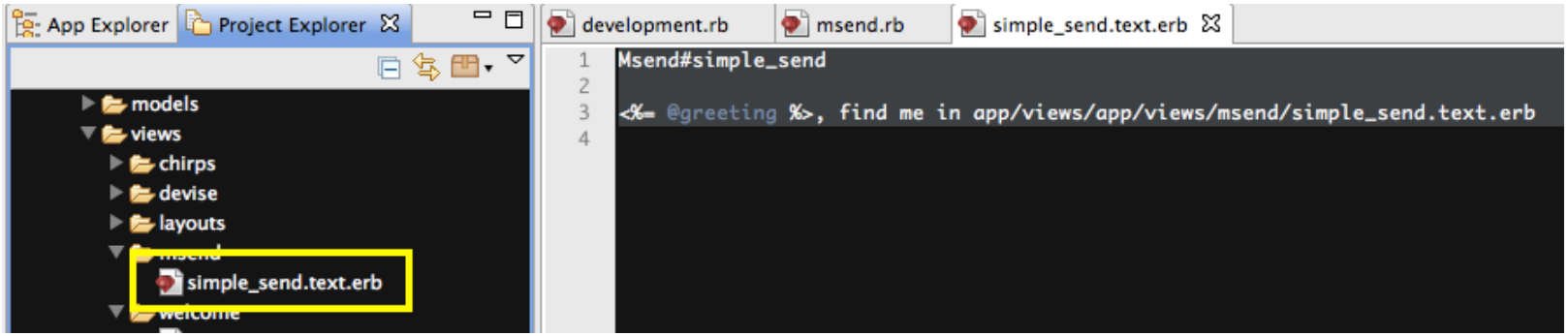
And now, simple_mail.text.erb is generated.

App/views/msend/ simple_send.text.erb

Msend#simple_send

<%= @greeting %>, find me in app/views/app/views/msend/ simple_send.text.erb

By arranging the content of this file, we can generate the content of the mail, just like in Web screen.



Sending mail

We have defined the simple_send method in Msend Class, with a recipient as an argument.

We can send mail in two ways,

```
Msend.simple_send( "userAddr" ).deliver
```

or

```
mail = Msend.simple_send( "userAddr" )
mail.deliver
```

Which screen to use ‘simple_send’?

That is what I have not designed yet.

Would you arrange the mail transmission screen in this PSE sight?

Problem registrant to chair?

It all depends on the system design.

I think, (if it were my system,) I will let the person who registered the problem should be a chair person of the discussion, and give the person to control the debate.

To keep the clarity of the system, I would show the counter of the removal of ‘causes’ and ‘solutions’ by the chair person.

All the visitors can see the counter.

Owner of the record

To realize the discussion above, we should explicitly introduce the concept of 'the owner of the record.'

In OR mapping, a "record" in the database is an instance in the object model. And then, only the owner and the chair person should be able to remove and modify the record.

Of course, this is one of the idea of the system.

Prepare for the Next Week

The Final lecture. The plan is 'the summary of the semester.'

What I think now is to let all students think about the 'System Design' keys of Object Oriented WEB and DB system. In other words, what is(/are) the best approach(es) to make the most of the Object Oriented characteristics of the ruby language environment.

Very vague? Exactly!