

# Email on Rails



Allen Fair  
BIGLIST Inc.

Philly on Rails  
August 6, 2007

# Email in Rails

## ■ ActionMailer

- Creates Email Messages
- Invokes SMTP or Sendmail for delivery
- Receives Email Messages for processing
- Includes updated TMail library

## ■ TMail

- Constructs action messages (outgoing)
- Parses messages (incoming)
- Accesses, traverses parts/attachments, headers, etc.
- Supports most RFC822 and MIME standards

# Email Review

- Mail User Agent (MUA)
  - Reading and Composition
- Mail Transport Agent (MTA)
  - Queuing & Delivery
- Email
  - Envelope – Of Interest to your MTA
    - Return Path – Specifies source/bounce address
    - Recipients – List of addresses to receive the email
  - Message – Of interest to your MUA
    - Headers
    - Simple Body or MIME Multipart Container
    - Attachments and Body Parts

# Email Headers

- From: User Name <user1@example.com>
- To: User Name <user2@example.com>
- Reply-To: <user3@example.com>
- Subject: [PhillyOnRails] Email on Rails Talk
- Date: Sat, 04 Aug 2007 14:43:01 -0400
- Content-Type: text/plain
- Received: from smtp.x.com by mail.y.com ...
- X-\*:

# Three Ways Home

- From: [allen@example.com](mailto:allen@example.com) "Allen"
  - Default "reply to" address
  - Matches address book for "safe list"
  - Displayed to user
- Reply-To: [fair@example.com](mailto:fair@example.com)
  - Overrides "from" address
- Return Path: [bounce-user=x.com@y.com](mailto:bounce-user=x.com@y.com)
  - Used by MTA to authenticate sender and returned bounced emails
  - Not used or seen by most mail clients
  - Variable Envelope Return Path (VERP)-Friendly

# Simple text/plain email

From: <sender@example.com>

To: <recipient@example.com>

Subject: Text is plain

Date: Tue, 5 Jun 2007 05:58:03 -0400

This is a plain text email.

# MIME Types



- text/plain
- text/html
- multipart/alternative
- image/jpeg
- multipart/mixed
- multipart/related
- Stir to combine...

# Complex Email Message

- **multipart/mixed** – Body and attachments
  - **multipart/alternative** – Versions for viewing
    - **text/plain** – Text version of email body
    - **multipart/related** – References positioned images (within HTML body)
      - **text/html** – HTML version of email body
      - **image/jpeg** – Image embedded inside the HTML body part
  - **application/pdf** – Some attachment

# Components of Rails Email

- Configuration in `environments.rb` or `environment/*.rb`
- Email Out
  - Mailer model (`ActionMailer::Base`)
  - Message body Templates
- Email In
  - SMTP or IMAP checker
  - Email Controller for direct delivery (`procmail`, `.qmail`)
  - `Mailer#receive` instance method
- Plugins

# ActionMailer Configurations

- Set in config/environment\*
- ActionMailer::Base.xxxxxx = value
  - smtp\_settings = { }
  - sendmail\_settings = { }
  - raise\_delivery\_errors = false
  - delivery\_method = :smtp
  - perform\_deliveries = true
  - default\_charset = "utf-8"
  - See rdoc for full list.
- ActionMailer::Base is well documented!

# ActionMailer::Base Configs

- `config.action_mailer.raise_delivery_errors = true`
- `ActionMailer::Base.delivery_method =`
  - `:smtp` (default, needs `smtp_settings` defined)
  - `:sendmail` (\*nix, optional `sendmail_settings`)
- `ActionMailer::Base.smtp_settings = {`
  - `:address => "smtp.example.com",`
  - `:port => 25,`
  - `:domain => 'www.example.com',`
  - `:user_name => "user@example.com",`
  - `:password => 'mypass', # no TLS support`
  - `:authentication => :login }`

# Configuring Sendmail

- default:
  - `sendmail -i -t [recipients....] < msg`
- In `./config/environment*`
  - `sendmail_settings = { :location=> '/usr/sbin/sendmail', :arguments=> '-i -t' }`
- `-t`: read messages for recipient
  - not good multi-recipient messages
- `-i`: ignore dots on lines
  - SMTP signals `.` as EOF
- `-f` return path (useful for VERP)

# Add Mailer Class to Application

```
$ script/generate mailer UserMailer register  
password ...
```

```
exists app/models/  
create app/views/user_mailer  
exists test/unit/  
create test/fixtures/user_mailer  
create app/models/user_mailer.rb  
create test/unit/user_mailer_test.rb  
create app/views/user_mailer/register.rhtml  
create test/fixtures/user_mailer/register  
create app/views/user_mailer/password.rhtml  
create test/fixtures/user_mailer/password
```

# Building the Message

```
# app/models/user_mailer.rb
class UserMailer < ActionMailer::Base

  def register(user, sent_at = Time.now)
    subject    'Thank you for registering'
    body       :user=>user #instance vars in rhtml
    recipients user.email
    from       'admin@example.com'
    sent_on    sent_at
    headers    'reply-to'=>'support@example.com'
  end

  ...
end
```

# Message Body Template

```
<%# app/views/user_mailer/register.rhtml %>
```

```
<%= @user.name %> ,
```

Thanks for registering. Please click the following link to validate your registration and use our application!

```
<%= url_for(:host => "example.com",  
           :controller => "user", :action => "confirm",  
           :id=>@user.id,  
           :code=>@user.confirm_code ) %>
```

Sincerely,  
mydomain.com

# Sending the email

```
# app/models/user.rb
def self.register(info)
  user = User.create(info)
  UserMailer.deliver_register(user)
end
```

```
# app/controllers/user_controller.rb
def forgot_password
  @user = User.find_by_email(params[:email])
  # Alternate method to create, then deliver
  the mail
  tmail = UserMailer.create_password(@user)
  UserMailer.deliver(tmail)
end
```

# Creating Email in HTML

```
class UserMailer < ActionMailer::Base

  def register(user, sent_at = Time.now)
    subject    "Thanks for registering"
    body       :user=>user # instance vars in rhtml
    recipients user.email
    from       'admin@example.com'
    sent_on    sent_at
    headers   'reply-to'=>'support@example.com'
    content_type "text/html"
  end

  ...
end
```

# The HTML Mail Template

```
<%# app/views/user_mailer/register.rhtml %>  
<p><%= @user.name %>,</p>
```

```
<p>Thanks for registering. Please click the  
following link to validate your registration and  
use our application!</p>
```

```
<div>
```

```
<%= url_for(:host => "example.com",  
           :controller => "user", :action => "confirm",  
           :id=>@user.id,  
           :code=>@user.confirm_code ) %>
```

```
</div>
```

```
<p>Sincerely,<br />
```

```
mydomain.com
```

```
</p>
```

# Attachments

```
def signup_notification(recipient)
  recipients recipient.email_address_with_name
  subject    "New account information"
  from       "admin@example.com"

  attachment :content_type => "image/jpeg",
    :body => File.read("an-image.jpg")

  attachment "application/pdf" do |a|
    a.body = generate_your_pdf_here()
  end
end
```

# Multiple Content Types

- Implicit with special view name formats:
  - register.text.plain.rhtml (\*.text.plain.erb)
  - register.text.html.rhtml (\*.text.html.erb)
- Explicit with attaching each part

```
part :content_type => "text/html",
:body => render_message("register-as-html",
:account => recipient)

part "text/plain" do |p|
  p.body = render_message("register-as-plain",
:account => recipient)
  p.transfer_encoding = "base64"
end
```

# Custom Mail with TMail

```
headers = { :subject => "Hello", ... }
text_part = new_body_part(txt, 'text/plain')
html_part = new_body_part(html, 'text/html')
mail = TMail::Mail.new
headers.each { |f,v| mail[f.to_s] = v }
mail.mime_encode_multipart(true)
mail.content_type = 'multipart/alternative'
mail.parts << text_part << html_part
UserMailer.deliver(mail)
```

```
def self.new_body_part(data, ct='text/plain')
  part = TMail::Mail.new
  part.body = data
  part.content_type = ct
  part
end
```

# Gotchas

- Recipients from email headers (sendmail)
- Use `deliver_register()`, not `register()`
- Not good for mass mailings
- models with views!
- Helpers are supported, but not automatically created  
(`app/helpers/user_mailer_helper.rb`)
  - `app/helpers/application_helper.rb`
- Use `script/console` for debugging
- Errors/exceptions are silenced by default

# Receiving Email

- POP3 or IMAP
  - Trigger or Cron process
- Local Delivery
  - Procmail
  - qmail-local (dot-qmail)
  - sendmail/postfix
- Sending to Rails
  - script/runner (not recommended in production)
  - curl to /controller/action

# Receive Email via POP3/IMAP

```
require 'net/pop'  
def self.check_pop # in user_mailer.rb or plugin  
  Net::POP3.start(@@pop_settings[:server],  
    @@pop_settings[:port],@@pop_settings[:user_name],  
    @@pop_settings[:password]) do |pop|  
    if pop.mails.empty?  
      logger.info "NO MAIL"  
    else  
      pop.mails.each do |email|  
        begin  
          new.receive(TMai::Mail.parse(email.pop))  
          email.delete  
        rescue Exception => e  
          logger.error ...  
        end  
      end  
    end  
  end  
end  
end
```

# Running the POP checks

```
# Place in ./config/environment*
class ActionMailer::Base
  @@pop_settings = {:server=>"mail.comcast.net",
    :port=> 110, :user_name=>'username',
    :password=>'secret', }
end
# model/user_mailer.rb
def self.receive_loop(seconds=300, iterations=0)
  iteration = 1
  while true do
    logger.info "Running Mail Importer..."
    check_pop
    logger.info "Finished Mail Importer."
    return if iteration==iterations
    iteration += 1
    sleep seconds
  end
end
end
```

# Directing email to HTTP

```
class EmailController < ApplicationController

  def receive
    if params[:mail]
      UserMailer.receive(params[:mail])
      render "Ok", :status => HTTP::Status::OK
    else
      render "Error: This action is intended to
        transfer email into the application",
        :status => HTTP::Status::BAD_REQUEST
    end
  end
end

end
```

# Running the /email/receive action

```
$ curl -F "mail=<msgfile" \  
  http://localhost/email/receive
```

## Security:

- Can be run from separate email server
- Lock down this url to public
- Scan and drop incoming virus messages
- Scan and drop incoming Spam messages
- Prevent unexpected addresses from being accepted by SMTP (SMTP-Level Rejection)

# Mass Mailings

- Send to entire subscriber/registration list
  - Hide Individual email addresses (To: list@domain)
  - SMTP and sendmail-command recipient limitations
- Monitor bounced emails
  - Remove bounced addresses
  - Variable Envelope Return Path (VERP)

# VERP: Variable Envelope Return Path

- Best for automating results from multiple deliveries or mailing lists
- adds the recipient email address to return path dynamically
- Supported by many major MTA's
- Return Path (RP) In: `msg-123-@example.com`
- Qmail VERP Ret. Path: `user-@host-@[]`
- RP Out: `msg-123-user=x.com@example.com`

# Running your own Mail Server?

- Anti-Virus – ClamAV, Anti-Spam – Spam Assassin
- Whitelists – AOL, Yahoo, MSN
- Feedback Loops – AOL, MSN, Juno, etc.
- Concurrency Issues – Comcast, BellSouth
- Certified Email:
  - Sender Score Certified – MSN, Hotmail
  - Goodmail – AOL, Yahoo
- Domain Keys, DK/IM, SPF, DNS-Lookup
- RBL, SpamCop, Blacklists
- External SMTP/Delivery Service

# Recap

- Set up configurations for outgoing and incoming email
- `script/generate_mailer MyMailer` templates...
- `MyMailer.deliver_xxxx()`
- Send email carefully, be a good netizen
- Process Incoming email via SMTP or HTTP
- Prepare for Virus, Spam, and other Attacks
- Thank you!