

# Itch Scratching the ActionMailer API

A brief foray into the mind of a masochist!





# whoami?

Working in Rails & Ruby for 4+ Years

Maintainer of TMail & Author of Mail

DBA and Internet Plumber

Rails Contributor

Entrepreneur

ruby 

No CS Experience



y?

Mail Application

Challenge

TMail



# OMGLOLSPAM

“TMail was a great SPAM checker...  
if it crashed while parsing an email  
then you knew it was SPAM.”

- Koz



# Why not just fix TMail?

Released 10 patch and minor versions

I wrote over 600 lines of Docs

Built on a Strategy Pattern

Parser Written in C

HARD TO FOLLOW

Read design patterns in Ruby



# Why not just fix TMail?

TMail was not very modular and hard to maintain

Wanted to write a major library

How hard could email be?

I wanted a nice DSL

Not as fun



# Mail's DSL is Simple

```
Mail.new({  
  :to => ['mikel@rubyx.com', 'david@rubyx.com'],  
  :from => 'mikel@lindsaar.net',  
  :subject => 'This is our subject',  
  :body => 'This is a simple body'  
})
```



# Mail's DSL is Simple

```
Date: Sun, 06 Jun 2010 22:24:46 -0400
From: mikel@lindsaar.net
To: mikel@rubyx.com,
    david@rubyx.com
Message-ID: <4c0c586ec050d_7cb800db1ac82331@mikel.local.mail>
Subject: This is our subject
Mime-Version: 1.0
Content-Type: text/plain;
    charset=UTF-8
Content-Transfer-Encoding: 7bit

This is a simple body
```



# Mail's DSL is Simple

```
Mail.new do
  to ['mikel@rubyx.com', 'dave@rubyx.com']
  from 'mikel@lindsaar.net'
  subject 'Multipart email'
  text_part do
    body 'This is plain text'
  end
  html_part do
    content_type 'text/html; charset=UTF-8'
    body '<h1>This is HTML</h1>'
  end
end
```

# Mail's DSL is Simple

```
Date: Sun, 06 Jun 2010 22:28:47 -0400
From: mikel@lindsaar.net
To: mikel@rubyx.com,
    dave@rubyx.com
Message-ID: <4c0c595fe98fc_7cb800db1ac828a5@mikel.local.mail>
Subject: Multipart email
Mime-Version: 1.0
Content-Type: multipart/alternative;
    boundary="====_mimepart_4c0c595b4ddf_7cb800db1ac8258b";
    charset=UTF-8
Content-Transfer-Encoding: 7bit

====_mimepart_4c0c595b4ddf_7cb800db1ac8258b
Date: Sun, 06 Jun 2010 22:28:47 -0400
Mime-Version: 1.0
Content-Type: text/plain;
    charset=UTF-8
Content-Transfer-Encoding: 7bit
Content-ID: <4c0c595fe7493_7cb800db1ac826bd@mikel.local.mail>

This is plain text

====_mimepart_4c0c595b4ddf_7cb800db1ac8258b
Date: Sun, 06 Jun 2010 22:28:47 -0400
Mime-Version: 1.0
Content-Type: text/html;
    charset=UTF-8
Content-Transfer-Encoding: 7bit
Content-ID: <4c0c595fe878c_7cb800db1ac827b6@mikel.local.mail>

<h1>This is HTML</h1>

====_mimepart_4c0c595b4ddf_7cb800db1ac8258b--
```





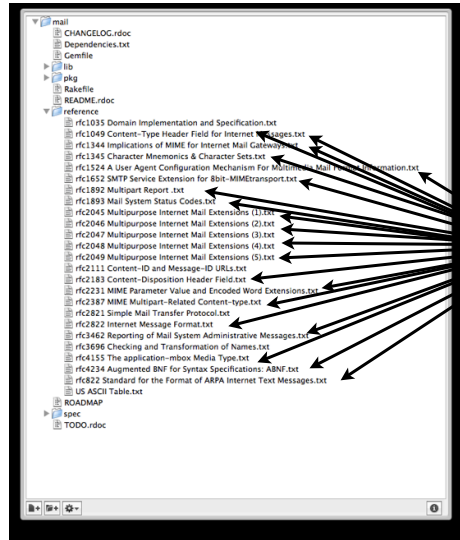
# How Hard Could It Be?

Read RFCs

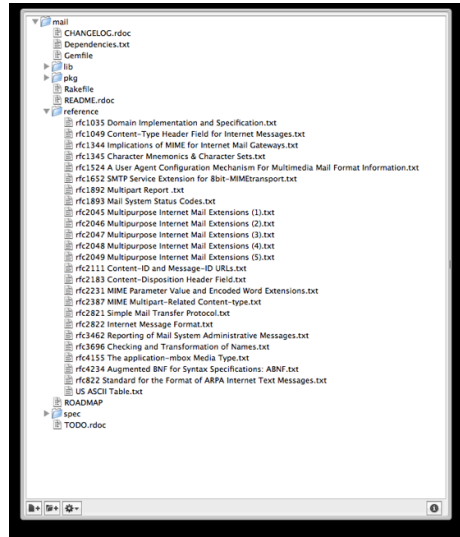
Read more RFCs

Keep reading RFCs...

“email is defined in RFCs, so should be easy..”



RFCs





# RSpec FTW!

I use RSpec because I am lazy

EVERYTHING in Mail is BDD

Regressions Caught

Seriously



# Spec'ing for Libraries

Start Simple

Comments in Spec Files are OK

Use Custom Matchers

Break Down Your Spec Files

Spec in Multiple Rubies



# Start Somewhere

```
def basic_email
  "To: mikel\r\nFrom: bob\r\nSubject: Hello!\r\n\r\nemail message\r\n"
end

describe "initialization" do

  it "should instantiate empty" do
    Mail::Message.new.class.should == Mail::Message
  end

  it "should instantiate with a string" do
    Mail::Message.new(basic_email).class.should == Mail::Message
  end
end
```



# Start Somewhere

```
it "should accept headers and body" do
  message = Mail.new do
    from 'mikel@me.com'
    to 'mikel@you.com'
    subject 'Hello there Mikel'
    body 'This is a body of text'
  end
  message.from.should == ['mikel@me.com']
  message.to.should == ['mikel@you.com']
  message.subject.should == 'Hello there Mikel'
  message.body.to_s.should == 'This is a body of text'
end
```



# Comments are OK

```
require 'spec_helper'
#
# The "In-Reply-To:" field will contain the contents of the "Message-
# ID:" field of the message to which this one is a reply (the "parent
# message"). If there is more than one parent message, then the "In-
# Reply-To:" field will contain the contents of all of the parents'
# "Message-ID:" fields. If there is no "Message-ID:" field in any of
# the parent messages, then the new message will have no "In-Reply-To:"
# field.

describe Mail::InReplyToField do

  describe "initialization" do
    it "should initialize" do
      doing { Mail::InReplyToField.new("<1234@test.lindsaar.net>") }.should_not raise_error
    end

    it "should accept a string with the field name" do
      t = Mail::InReplyToField.new('In-Reply-To: <1234@test.lindsaar.net>')
      t.name.should == 'In-Reply-To'
      t.value.should == '<1234@test.lindsaar.net>'
      t.message_id.should == '1234@test.lindsaar.net'
    end
  end
end
```



# Use Custom Matchers

when appropriate

```
it "should handle |Mikel Lindsaar <mikel@rubyx.com>|" do
  address = Mail::Address.new('Mikel (heh) <mikel@rubyx.com>')
  address.should break_down_to({
    :display_name => 'Mikel',
    :address      => 'mikel@rubyx.com',
    :local        => 'mikel',
    :domain       => 'rubyx.com',
    :format       => 'Mikel <mikel@rubyx.com>',
    :comments     => ['heh'],
    :raw          => 'Mikel <mikel@rubyx.com>'})
end
```



# Use Custom Matchers

when appropriate

```
it "should handle IPete(A wonderful ) chap) <pete(his account)@silly.test(his host)>!" do
  address = Mail::Address.new('Pete(A wonderful \) chap) <pete(his account)@silly.test(his host)>')
  address.should break_down_to({
    :name          => 'Pete',
    :display_name => 'Pete',
    :address       => 'pete(his account)@silly.test',
    :comments      => ['A wonderful \) chap', 'his account', 'his host'],
    :domain       => 'silly.test',
    :local        => 'pete(his account)',
    :format       => 'Pete <pete(his account)@silly.test> (A wonderful \) chap his account his host)',
    :raw         => 'Pete(A wonderful \) chap) <pete(his account)@silly.test(his host)>'})
end
```

# Custom Matchers

```
module CustomMatchers
  class BreakDownTo
    def initialize(expected)
      @expected = expected
    end

    def matches?(target)
      @target = target
      @failed = false
      @expected.each_pair do |k,v|
        @failed = k unless @target.send(k) == @expected[k]
      end
      !@failed
    end

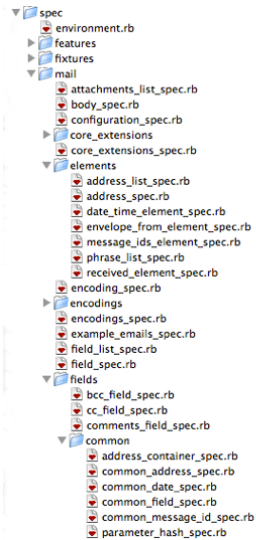
    def failure_message...
    def negative_failure_message...
    end

    # Actual matcher that is exposed.
    def break_down_to(expected)
      BreakDownTo.new(expected)
    end
  end
end
```





# Break Down Spec Files



26 Sub Directories

79 Ruby Spec Files

73 Edge Case Emails

Many Many Lines of Specs

1161 examples

No Failures :)



# Spec in Multiple Rubies

Have to be a good Ruby Citizen

JRuby

REE

Rubinius

(almost)

MRI - 1.8.6, 1.8.7, 1.9.1, 1.9.2



# But What About Real Life?

Mail parses the Enron and Trec Data Sets

```
mikel@baci ~/Code  
$ du -hs corpus/
```

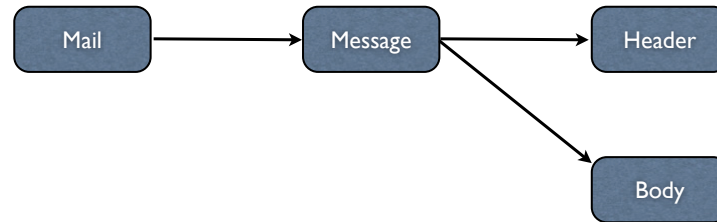
2.4G of Emails

```
mikel@baci ~/Code  
$ find corpus -name "*" -print | wc -l
```

328,248 Emails

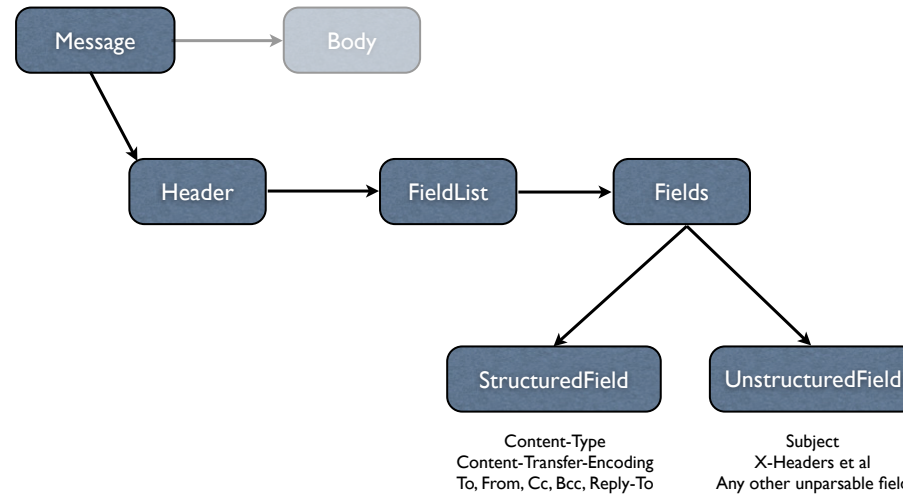


# show me the model!



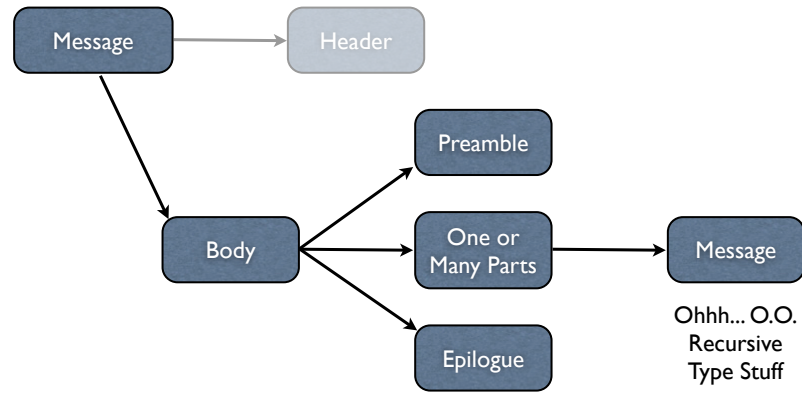


# show me the model!





# show me the model!





# show me the model!

Header Errors  
Parsers  
Body Decoding  
SPAM  
Mbox  
Sendmail Delivery  
Pattern Matching  
Interceptors  
Observers  
Inline Attachments

rfc822  
Delivery Agents  
And That's It!  
PGP Signing  
Simple Hey?  
IMAP  
Bad Emails  
OMGMIME!  
File IO

Legacy Emails  
SMTP  
POP3  
File Types  
Network  
Language Support  
Encoding  
Attachments  
Domain Key Emails  
Status Reports



# OK, ActionMailer?

Much Easier to Use

Nicer Defaults & Config

ActionMailer Methods Return Mail Object

Deprecated (not Decapitated) API

Tutorials



# Much Easier to Use

New Mailer Directory

Can Call “mail” method and pass a hash

```
def enquiry_email(user)
  mail(:to => user.email,
       :subject => "G'day Mate!",
       :from => "enquiries@rubyx.com",
       :sender => "system@rubyx.com")
end
```



# Nicer Defaults

```
class Notifier < ActionMailer::Base
  default :from => "mikel@rubyx.com",
         :reply_to => "enquiries@rubyx.com",
         "X-Time-Code" => Proc.new { Time.now.to_i.to_s }

  def welcome_email(user)
    @name = user.name
    @message = user.message
    mail(:to => user.email,
        :subject => "G'day Mate!",
        :sender => "system@rubyx.com")
  end
end
```



# Nicer Config

```
Blog::Application.configure do
  config.action_mailer.smtp_settings =
    { :address      => "smtp.gmail.com",
      :port        => 587,
      :domain      => 'your.host.and.domain.name',
      :user_name   => '<gmail username>',
      :password    => '<gmail password>',
      :authentication => 'plain',
      :enable_starttls_auto => true }
end
```



# AM Returns Mail

```
mail = Notifier.enquiry_email(user).deliver
mail.class #=> Mail::Message
```

```
mail = Notifier.enquiry_email(user)
mail.class #=> Mail::Message
mail.subject = "Good day sir" if user.british?
mail.deliver
```



# Mail Call Back Hooks

```
class InterceptorAgent
  def self.delivering_email(mail)
    mail['X-Original-To'] = mail.to
    mail['X-Original-Cc'] = mail.cc
    mail['X-Original-Bcc'] = mail.bcc
    mail.to = 'mikel@rubyx.com'
    mail.cc = mail.bcc = nil
  end
end

Mail.register_interceptor(InterceptorAgent)
mail = Mail.new(:to => 'steve@apple.com',
               :cc => 'bill@microsoft.com',
               :bcc => 'ceo@aapt.com')
mail.deliver #=> mail sent _only_ to 'mikel@rubyx.com'
```



# Mail Call Back Hooks

```
class ObserverAgent
  def self.delivered_email(mail)
    puts "Delivered #{mail.message_id}"
  end
end

mail = Mail.new
Mail.register_observer(ObserverAgent)
mail.deliver
#=> "Delivered <4c0c595fe98fc_7cb800db1ac828a5@mikel.local.mail>"
```



# Mail Call Back Hooks

```
class DeliveryAgent
  def self.deliver_mail(mail)
    SpecialDeliveryMethod.write(mail)
  end
end

mail = Mail.new
mail.delivery_handler = DeliveryAgent
mail.deliver
#=> mail message "sent" by SpecialDeliveryMethod
```



# Deprecated API

Decapitation Free Zone

Did not nuke the old API

New API triggers **ONLY** if you call “mail”

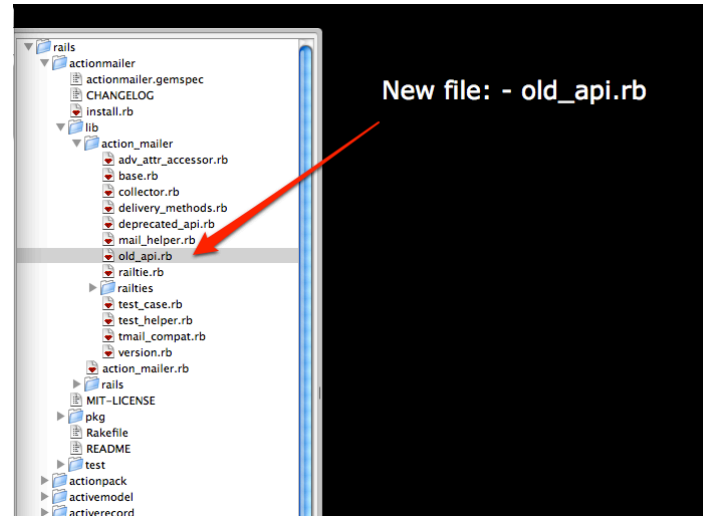
Old API in place if no “mail” call

Will be Removed in 3.1



# Deprecated API

Decapitation Free Zone





# Deprecated API

Decapitation Free Zone

```
def mail(headers={}, &block)
  # Guard flag to prevent both the old and the
  # new API from firing. To be removed when
  # old API is removed
  @mail_was_called = true
  m = @_message
```



# Deprecated API

Decapitation Free Zone

```
module ActionMailer
  module OldApi #:nodoc:
    def process(method_name, *args)
      initialize_defaults(method_name)
      super
      unless @mail_was_called
        create_parts
        create_mail
      end
      @_message
    end
  end
end
```



# HELP!!!

Replace treetop with Ragel or RACC

Implement full POP3 and IMAP Support

Domain Key Implementation

PGP Signing

More Documentation



# Tutorials

[lindsaar.net](http://lindsaar.net)

[www.RailsDispatch.com](http://www.RailsDispatch.com)

[www.RailsCasts.com](http://www.RailsCasts.com)



# KTHXBAI

Questions?

@raasdnil

mikel@rubyx.com

lindsaar.net

ruby 

