PMR3304 - Sistemas de Informação - 09

Aula 09

Prof. Dr. Marcos de Sales Guerra Tsuzuki

25 de Outubro de 2019

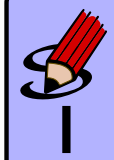
PMR-EPUSP

Enviando e-mails

Enviando e-mails - Introdução

Nesta aula, vamos continuar o desenvolvimento da aplicação de estudantes, awards e courses. Será necessário desenvolver um pouco de infraestrutura, especificando **views** indicando o conteúdo da mensagem e informando o Rails quando a mensagem deve ser enviada. Inicialmente, vamos criar um **mailer**:

```
cd students
rails generate mailer AwardMailer
```



Para você fazer em casa

Execute os comandos descritos.

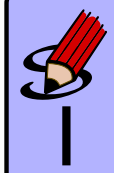
Observe que foi criado um novo diretório: **mailers**. Vamos verificar o arquivo **app/mailers/award_mailer.rb**.

```
class AwardMailer < ApplicationMailer
end
```

Enviando e-mails - Introdução

Vamos configurar valores padrão para os diversos campos: *to*, *subject* e *from*. Podem incluir outros como: *cc*, *cco*.

```
class AwardMailer < ApplicationMailer
  default from: "me@usp.br"
  def award_email(award)
    @award = award
        mail(:to => 'Sansao <sansao@gmail.com>',
          :subject => "Award from Sistemas de Informacao")
  end
end
```

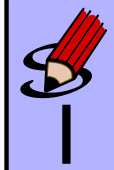


Para você fazer em casa

Modifique o arquivo conforme descrito.

O conteúdo do e-mail pode ser em HTML ou apenas texto. Para este fim, serão criadas duas **views** distintas, uma para cada caso. Vamos criar a **view** para os e-mails de apenas texto. Para isto, vamos criar o arquivo **app/views/award_mailer/award_email.text.erb**.

```
The <%= @award.name %> award for <%= @award.year %> has
      gone to <%= @award.student.name %>.
```



Para você fazer em casa

Crie o arquivo conforme descrito.

Enviando e-mails - Introdução

Vamos criar a **view** para os e-mails com HTML. Para isto, vamos criar o arquivo **app/views/award_mailer/award_email.html.erb**.

```
<!DOCTYPE html><html><head>
<meta content="text/html; charset=UTF-8" http-equiv="
    Content-Type" />
</head><body>
  <h1>Award Notification</h1>
  <p>The <%= @award.name %> award for <%= @award.year
      %> has gone to
  <%= @award.student.name %>.</p>
</body></html>
```

Agora, vamos configurar para que um e-mail seja enviado quando um *student* receber um *award*. Para isto, vamos modificar o método **create** do **controller** *award*, o arquivo é **app/controller/awards_controller.rb**. O seguinte código será inserido:

```
def create
  @award = @student.awards.build(award_params)
  # era @award = Award.new(award_params)

  # Tell the AwardMailer to send a notice Email
  AwardMailer.award_email(@award).deliver
```

Resta agora, informar mais alguns detalhes para que o Rails consiga enviar os e-mails. Qual servidor será utilizado? Estas informações devem ser fornecidas no arquivo **config/environments/development.rb** para o ambiente de desenvolvimento e em **config/environments/production.rb** para o ambiente de produção. É comum que existam configurações distintas para ambos os ambientes, e também para o ambiente de testes. O e-mail poderá ser enviado a um processo sendmail local, ou para um servidor SMTP.

Enviando e-mails - Introdução

Como servidor local, vamos utilizar o *gem mailcatcher*. Inicialmente, devemos instalar a gem:

```
gem install mailcatcher
gem install sendmail
```

Após a instalação, você poderá ver o *mailcatcher em funcionamento*, basta executá-lo:

```
mailcatcher
```

Ele possui uma porta local para smtp `stmp://127.0.0.1:1025` e outra para acesso `http://127.0.0.1:1080`. Tente acessar em um browser o endereço http.

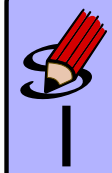
O arquivo **config/environments/production.rb** deverá conter:

```
# Don't care if the mailer can't send.
config.action_mailer.raise_delivery_errors = false
config.action_mailer.delivery_method = :smtp
config.action_mailer.smtp_settings = { :address => "localhost", :port =>
    1025 }
```

Caso seja de seu interesse, o *Gmail* possui configurações específicas [1].

[1] https://guides.rubyonrails.org/action_mailer_basics.html#action-mailer-configuration-for-gmail.

Uma vez com tudo configurado, podemos realizar o teste do envio de e-mail. Abra dois consoles, um para executar o **mailcatcher -b** (isto abrirá o browser automaticamente) e outro para executar o **rails server**. As duas aplicações devem estar rodando simultaneamente. Agora podemos criar um *award* para um *student*.



Para você fazer em casa

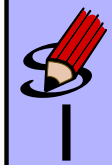
Faça o teste de envio de e-mail.

Veja que o **mailcatcher** indicará que um e-mail foi enviado. No browser, será possível verificar o e-mail enviado.

É possível visualizar o e-mail enviado também pelo Rails. Observe que em modo de desenvolvimento, os e-mails não são enviados. Uma classe para este fim foi criada em **test/mailers/previews/award_mailer_preview.rb**. Para visualizar o **award_email**, vamos acrescentar um método com o mesmo nome que a classe **preview** e acionaremos o método *AwardMailer.award_email*, fornecendo um *award* válido. Neste caso, o primeiro *award* da base de dados.

```
class AwardMailerPreview < ActionMailer::Preview
  def award_email
    AwardMailer.award_email(Award.first)
  end
end
```

Com esta configuração, já podemos visualizar o e-mail, no seu browser selecione a URL `http://localhost:3000/rails/mailers`. Será possível visualizar apenas um e-mail.



Para você fazer em casa

Visualizar o e-mail, conforme descrito.

É possível incluir o visualizador de e-mail utilizando a opção *preview_path*.

Recebendo e-mails - Introdução

Receber e-mails é mais difícil que enviá-los. A configuração de um servidor para receber os e-mails e enviá-los para o Rails não é uma tarefa fácil.

Mas, vamos supor que os e-mails foram recebidos e veremos como utilizá-lo. Considere que o e-mail a seguir foi recebido.

Este e-mail contém um comando que será processado pelo sistema. A nota de um *student* será atualizada.

Enviando e-mails - Introdução

```
Return-Path: <edd@example.org>
X-Original-To: edd@example.org
Delivered-To: edd@mail.example.org
Received: from [192.168.250.137] (foo.example.org [192.168.250.5])
(using TLSv1 with cipher AES128-SHA (128/128 bits))
(No client certificate requested)
by mail.example.org (Postfix) with ESMTP id C01F3140040
Message-Id: <B3E7FCF0-49E0-4C1D-8547-B5459DB72D69@example.org>
From: Edd Dumbill <edd@example.org>
To: Edd Dumbill <edd@example.org>
Content-Type: text/plain; charset=US-ASCII; format=flowed
Content-Transfer-Encoding: 7bit
Mime-Version: 1.0 (Apple Message framework v928.1)
```

Enviando e-mails - Introdução

Subject: Score
Date: Wed, 6 Aug 2012 04:32:54 −0700
X−Mailer: Apple Mail (2.928.1)
Status: RO
Content−Length: 56
Lines: 7

Student: 1
GPA: 3.34

Vamos criar uma base de e-mail para *student*

```
rails generate mailer Student Mailer
```

O arquivo **app/mailers/student_mailer.rb** deverá ser modificado para ficar como o próximo slide.

Enviando e-mails - Introdução

```
class StudentMailer < ApplicationMailer

  def receive(email)
    return unless email.subject =~ /^Score/

    sender = email.from[0]
    user = User.find_by_email(sender)
    unless !user.nil? && user.email == 'edd@example.org'
      logger.error "Recusando mensagens de remetentes desautorizados"
      return
    end
    # Apenas e-mail com titulo 'Score' e enviados pelo administrador

    # Extrair o conteudo da mensagem
    content = email.multipart? ? (email.text_part ? email.text_part.body.decoded :
          nil) : email.body.decoded
```

Enviando e-mails - Introdução

```
# Procura em conteudo linha por linha por student e GPA
content.split(/\r?\n/).each do |l|
  if l =~ /Student:\s*(\d+)/i then
    @student = Student.find_by_id($1.to_i)
  end
  if l =~ /GPA:\s*(\d+\.\d+)/i then
    @gpa = $1.to_f
  end
end
```

```
    # Se o dado foi encontrado facamos a troca
    if @student and @gpa
      @student.update_attribute('grade_point_average', @gpa)
      logger.info "Updated GPA of #{@student.name} to #{@gpa}"
    else
      logger.error "Couldn't interpret scores message"
    end
  end
end
```

Vamos considerar um e-mail desautorizado, como o abaixo:

```
Return-Path: <simonstl@simonstl.com>
X-Original-To: simonstl@simonstl.com
Delivered-To: simonstl@simonstl.com
Received: from SimonMacBook.local (cpe-24-59-184-80.twcny.res.rr.com
        [24.59.184.80])
            by mail.simonstl.com (Postfix) with ESMTPSA id 5384D18C0031
            for <simonstl@simonstl.com>; Sat, 14 Apr 2012 12:48:30 -0400 (EDT)
Message-ID: <4F89AA5A.4060203@simonstl.com>
Date: Sat, 14 Apr 2012 12:48:26 -0400
From: "Simon St.Laurent" <simonstl@simonstl.com>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:11.0) Gecko/20120327
        Thunderbird/11.0.1
MIME-Version: 1.0
```

Enviando e-mails - Introdução

```
To: "Simon St.Laurent" <simonstl@simonstl.com>
Subject: Score
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

Student: 1
GPA: 3.34
```

Esta mensagem deverá ser salva no diretório raiz com o nome **mail1.msgs**. Para executar o recebimento, vamos rodar o servidor em um prompt de comando, e em outro prompt de comando vamos executar o comando abaixo. Ou seja, estamos com dois prompt de comando.

```
cat mail1.msgs | rails runner 'StudentMailer.receive(STDIN.read)'
```

Após a execução, observe o arquivo **log/development.log**. Este é o local onde as mensagens estão escritas. Deve estar presente em algum lugar a mensagem: **Recusando mensagens de remetentes desautorizados**.

Vamos executar a mensagem descrita nos slides 14 e 15 que deverá ser salva no diretório raiz com o nome **mail2.msg**. Para este fim, será necessário cadastrar o e-mail *edd@example.org* como usuário do sistema. Podemos executar o comando abaixo, como descrito no slide anterior.

```
cat mail2.msgs | rails runner 'StudentMailer.receive(STDIN.read)'
```

Verifique a nota do estudante 1, ela deve ter sido modificada.

Expressões Regulares

Três casos simples de expressões regulares:

```
validates_format_of :secret, :with => /[0–9]/,
  :message => "precisa conter pelo menos um numero"
```

```
validates_format_of :secret, :with => /[A–Z]/,
  :message => "precisa conter pelo menos um caracter maiusculo"
```

```
validates_format_of :secret, :with => /[a–z]/,
  :message => "precisa conter pelo menos um caracter minusculo"
```

Um teste simples pode ser feito, pela execução em modo console. No prompt de comando tecle **irb**. Veja a seguinte execução:

```
C:\Sites\students>irb
irb(main):001:0> secret = "password"
=> "password"
irb(main):002:0> if secret =~ /[0–9]/ then puts "sim, esta ai" else "nao, nao esta"
        end
=> "nao, nao esta"
irb(main):003:0> secret = "password1"
=> "password1"
irb(main):004:0> if secret =~ /[0–9]/ then puts "sim, esta ai" else "nao, nao esta"
        end
=> "sim, esta ai"
irb(main):005:0>
```

Um outro exemplo onde se procura por uma string completa. O retorno indica a primeira aparição da string.

```
irb(main):007:0> sentence = "Ruby is the best Ruby–like programming language."
=> "Ruby is the best Ruby–like programming language."
irb(main):008:0> sentence =~ /Ruby/
=> 0
irb(main):009:0> sentence =~ /the/
=> 8
irb(main):010:0>
```

Um outro exemplo onde se procura por uma string completa. O retorno indica a primeira aparição da string.

- Podemos criar a própria lista de caracteres como: */[aeiou]/*;
- Uma outra possibilidade é: */[aeiouAEIOU]/*;
- Ignorando se é maiúscula ou minúscula: */[aeiou]/i*.;
- Podemos combinar: */[Rr][aeiou]by/*. Isto encontrará Ruby, ruby, raby, roby;
- Podemos negar a existência: */^[aeiou]/*.

Caracteres especiais:

- **.**: Casa com qualquer caracter;
- \ **d**: Casa com qualquer dígito;
- \ **D** : Casa com qualquer não dígito;
- \ **s**: Casa com caracteres de espaço como tab, CR, LF, FF;
- \ **S**: Casa com caracteres que não são do tipo de espaço;
- \ **w**: Casa com caracteres: A-Z, a-z, e 0-9;
- \ **W**: Casa com caracteres opostos aos indicados acima.

Expressões Regulares - Introdução

Um exemplo:

```
C:\Sites\students>irb
irb(main):001:0> secret = "password"
=> "password"
irb(main):002:0> if secret =~ /[\d]/ then puts "sim, esta ai" else "nao, nao esta" end
=> "nao, nao esta"
irb(main):003:0> secret = "password1"
=> "password1"
irb(main):004:0> if secret =~ /[\d]/ then puts "sim, esta ai" else "nao, nao esta" end
=> "sim, esta ai"
irb(main):005:0>
```

Mais Caracteres especiais:

- \ **f**: Form Feed
- \ **n**: Newline
- \ **r**: Carriage return
- \ **t**: Tab

Modificadores:

- **i**: ignorar maiúscula e minúscula
- **m**: casamento de multiplas linha, o caracter **.** casa com o caracter \ **n**;

Âncoras:

- ^: Quando no início da expressão, ou seja casará apenas com o início. Se for de múltiplas linhas, será para o início de cada linha;
- **$**: Ao final da expressão. Se forem múltiplas linhas, será o final de cada linha;
- \ **A**: Apenas no início da expressão e não considera início de linhas;
- \ **Z**: Apenas ao final da expressão e não considera final de linhas;
- \ **b**: Marca o contorno entre duas palavras;
- \ **b**: Marca o contorno entre duas palavras, incluíndo espaços em branco;
- \ **B**: Marca aquilo que não é o contorno entre duas palavras.

Expressões Regulares - Introdução

Um exemplo:

```
C:\Sites\students>irb
irb(main):001:0> secret = "sempre ele"
=> "sempre ele"
irb(main):002:0> if secret =~ /^sempre/ then puts "sim, esta ai" else "nao, nao esta"
        end
=> sim, esta ai
irb(main):003:0> if secret =~ /ele$/ then puts "sim, esta ai" else "nao, nao esta" end
=> sim, esta ai
```

The End!